Personalized Opinion-based Recommendation

Ruihai Dong, Barry Smyth

Insight Centre for Data Analytics
School of Computer Science and Informatics
University College Dublin, Ireland

Abstract. E-commerce recommender systems seek out matches between customers and items in order to help customers discover more relevant and satisfying products and to increase the conversion rate of browsers to buyers. To do this, a recommender system must learn about the likes and dislikes of customers/users as well as the advantages and disadvantages (pros and cons) of products. Recently, the explosion of user-generated content, especially customer reviews, and other forms of opinionated expression, has provided a new source of user and product insights. The interests of a user can be mined from the reviews that they write and the pros and cons of products can be mined from the reviews written about them. In this paper, we build on recent work in this area to generate user and product profiles from user-generated reviews. We further describe how this information can be used in various recommendation tasks to suggest high-quality and relevant items to users based on either an explicit query or their profile. We evaluate these ideas using a large dataset of TripAdvisor reviews. The results show the benefits of combining sentiment and similarity in both query-based and user-based recommendation scenarios, and also disclose the effect of the number of reviews written by a user on recommendation performance.

Keywords: recommender systems, opinion mining, sentiment analysis, personalization

1 Introduction

Recommender systems help to provide users with the right information at the right time. They do this by profiling a user's interests and preferences over time and use these profiles to select and/or rank items for presentation by preferring those that are similar to the ones the user has liked in the past. In an e-commerce setting, recommender systems endeavour to learn from our past purchasing habits in order to identify new products that we may wish to purchase in the future. Getting this right can mean improved conversion rates for the retailer and improved satisfaction levels for the shopper. More satisfied customers are more loyal customers which improves the likelihood of repeat business, a win-win for customer and retailer alike.

Different types of recommendation techniques rely on different types of data. For example, collaborative filtering [1], either neighbourhood methods [2, 3] or latent factor models [4], relies on product ratings. These ratings may be explicitly provided by users, or they may be inferred from their behaviour. And patterns in these ratings are used to identify similar users and relevant products for recommendations.

Content-Based recommendation [5] is another important approach for building recommender systems, based on the availability of specific item knowledge. For example, product meta-data or catalog-data may be used to characterise products in terms of specific features (type, price, size, weight etc). When this type of product data is available the preferences of users can be expressed in terms of the products that they have purchased and the features of these products. Recommendations can then be based on various forms of feature similarity, by recommending products that are *similar* to the user's profile. This approach obviously shares many commonalities with case-based reasoning methods and in fact many case-based recommendation techniques have been proposed based on representation, similarity, and retrieval ideas from the CBR community [6, 7].

In the past, a big challenge when building recommender systems has been ensuring the availability of and access to these different sources of data. Sometimes product data is hard to come by, limiting the efficacy of content/case-based methods. And the type of rating data used by collaborative filtering approaches can be notoriously sparse. This has led researchers to develop hybrid models that make the best of both worlds [8] and also led to the exploit of auxiliary information; see [9, 10].

In recent years, a new alternative data source has emerged with the ubiquity of user-generated reviews. The intuition is that these reviews contain important product knowledge and valuable customer preferences and insights for use in recommendation; see [11, 12]. But the information contained within user-generated reviews can be noisy and unstructured. Nevertheless, opinion mining and natural language processing techniques (for example, see [13]) are now robust enough for researchers to use user-generated review content as an alternative (or complementary) source of recommendation knowledge [14].

User-generated reviews are plentiful and they contain rich product feature information including sentient information. For example, *The Thai red curry was delicious, ..., but price at $22, quite expensive*, tells us that the restaurant in question serves a very tasty Thai red curry (positive sentiment) that costs $22 but it is expensive (negative sentiment). This combination of traditional feature-value information and sentiment means that we can generate recommendations that are not only *similar* to those a user has liked in the past, but that are also *better* based on features that matter to the user; see [15, 16].

The features mentioned by a user in her reviews may reflect her preferences. Intuitively, if a feature is mentioned many times by a user, it may indicate that it is an important one. Musat et al. [17] built a user interest profile for each user based on the topics mentioned in their reviews and used these profiles to produce personalized product rankings. Liu et al. [18] built user preferences based on the assumption that a user may have a higher requirement for a feature if she frequently gives a lower score for the feature compared to other users; see also the work of [19] on estimating reviewer weights at the aspect (feature) level.

In this paper, we focus on a pair of use-cases in the context of a hotel recommendation site such as TripAdvisor. Specifically, we imagine a traveller attempting to book a suitable hotel for an upcoming trip to a new city. In the first use-case (*more-like-this*) we consider the common scenario where the traveler has a previous hotel that they have liked but in a different city and they wish to look for something similar in their target city. In this use-case, the previous hotel becomes a target query and is used to select and rank hotels from the target city. We describe a *query-based* technique to show how the combination of similarity and sentiment can be used to recommend better hotels than similarity alone. In our second use-case (*personalization*) we use the user's profile as an implicit query in order to recommend hotels from the target city, again looking at the benefits of using similarity and sentiment; we refer to this as a *user-based* approach.

2 Mining User Profiles & Item Cases

This paper extends recent work on mining opinions from user reviews to generate user profiles and item descriptions. The work of [20, 15, 21] is especially relevant and describes how shallow NLP, opinion mining, and sentiment analysis can be used to extract rich feature-based product descriptions (product *cases*) based on the features that users mention in their reviews, and also the polarity of their opinions. In order to generate user profiles and hotel descriptions, we follow four basic steps:

1. Mine hotel features from user-generated reviews (extending the techniques described in [15]) to generate a set of *basic* hotel features and sentiment.
2. Apply clustering techniques to group related basic features together and use these clusters as *high-level* features. We refer to these as *clustered* features.
3. Aggregate clustered features and sentiment mined from the reviews of a hotel to generate *hotel cases.*
4. Aggregate clustered features for users to generate user profiles; that is, the features mined from the reviews of a given user form the basis of this user's profile.

This is summarized in Figure 1 and produces a set of hotel cases and user profiles/cases which are used as the basic recommendation data for our recommender system as shown. In the next section we will describe the details of the recommendation process but first, in what follows, we will provide additional details about this feature mining and case generation process.

2.1 Features & Sentiment

The feature extraction and sentiment analysis approach adopted in this work are based closely on the approach taken by [15]. As such we mine *bi-gram* (adjective-

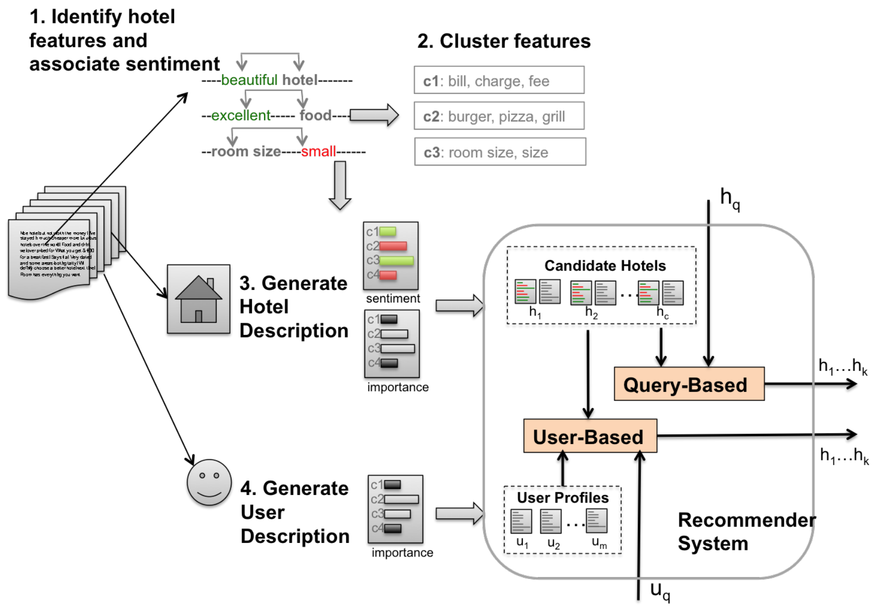


Fig. 1. High-level opinion mining architecture: basic features are extracted from user generated reviews; these features are clusterd into related sets; these clustered features are used as the basis for hotel/item cases and user profiles/cases.

noun) and *single-noun* features using a combination of shallow NLP and statistical methods.

In the current work we extend this approach to also consider additional *tri-gram* features by using the patterns described in the work of [22]. This allows us to find features like *member of staff*, *bottle of water*.

These single-noun, bi-gram, and tri-gram features are filtered using a number of statistical techniques as described previously in [15]. For example, we eliminate certain bi-gram features if the adjective is a common sentiment word; this way we can eliminate false bi-gram features such as *great location*, *excellent service*, which are really single-noun features.

Next, for each extracted (basic) feature, f_i, mentioned in some sentence s_j of review r_k, we must evaluate its sentiment. We do this by finding the closest *sentiment word* w_{min} to f_i in s_j; sentiment words are available from a sentiment lexicon and for the purpose of this work we rely on the common Bing Lui sentiment lexicon [23]. Note, if no sentiment words are present in s_j then the feature f_i is labeled as *neutral*.

In order to further filter the basic features, we use the opinion pattern mining technique described by [15]. To do this we identify the part-of-speech (POS) tags for w_{min}, f_i, and any words that occur between them. These POS tags constitute an abstract opinion pattern. After a complete pass over all features, the frequency of occurrence of all of these opinion patterns is computed. Patterns that occur more than k times are considered valid, on the grounds that less frequent patterns are likely to correspond to unusual or convoluted expressions of opinion that cannot be reliably understood. In this case, we set $k = 1$ and for each valid feature, f_i (that is, a feature which corresponds to a valid opinion pattern) we assign it a sentiment label based on the sentiment of its w_{min}. Finally, we reverse the sentiment of f_i if s_j contains a negation term within a 4-word distance of w_{min}.

The end result of this process is, for each review, a set of valid features (mentioned in the review) and their sentiment labels. These reviews are associated with specific hotels and specific users and so their features and sentiment contribute to the descriptions of these hotels and the profiles of the users; we will return to this later.

2.2 Clustering Features

One of the problems with the above approach is that it tends to produce a proliferation of features. In reviews, people may refer to the same types of things in a variety of ways. For example, in hotel reviews, some reviewers will comment on the *elevators* while others will talk about *lifts*. One reviewer might comment on the quality of the *pillows* while another may comment on the comfort of the *bed* or the softness of the *matress*. Rather than treating these as separate and independent features it is more natural and useful to recognize that they are referring to the same aspect of a hotel.

```
Cluster 1  - location, walking distance, distance, minutes work, ...
Cluster 2  - bill, charge, fee
Cluster 3  - burger,pizza,grill
Cluster 4  - chip, chicken, salad, fish, steak, meat
Cluster 5  - egg, bacon, waffle, pancake, sausage
Cluster 6  - cake, afternoon tea, sandwich
Cluster 7  - bar, beer, drink, cocktail
Cluster 8  - bed, pillow, mattress
Cluster 9  - bathroom, tube, toilet, shower, bath
Cluster 10 - stair, lift, elevator
Cluster 11 - sofa, king-bed, couch, living room
Cluster 12 - desk staff, hotel staff
```

Fig. 2. Examples for Clustered Features

To do this we apply standard clustering techniques to group related features together based on their similarities. Firstly, we associate each basic feature with a term-vector made up of the set of words extracted from the sentences that refer to this feature; these words are converted to lowercase, stemmed, and stop words are removed. Thus, each feature f_i is associated with a set of terms and the value of each term is a normalized term frequency weight. Next, we apply a standard clustering algorithm empirically setting the target number of clusters to be 35% of the total number of features; we used CLUTO[1]. The result is a set of feature clusters such as the examples in Figure 2. These clusters act as high-level features and they are used as the basis for generating hotel and user cases as we shall describe in the next sections.

2.3 Generating Hotel Cases

Each item/hotel (h_i) is associated with a set of customer reviews $reviews(h_i) = \{r_1, \ldots, r_n\}$ and the above processes extract a set of features and, ultimately, clusters, $c_1, \ldots, c_m$, from these reviews. Each cluster is comprised of a set of basic features and in effect, acts as a type of abstract feature, a *clustered feature*, such that the basic features it contains are related in some way. For example, in Figure 2 we can see that the features of *Cluster 1* are all related to the location of the hotel. The features in clusters 4, 5, and 6 are all related to food but each is separately related to *dinner*, *lunch*, or *breakfast* meals.

Effectively each clustered feature is labelled with the cluster id, c_j, and it is assigned an *importance* score and a *sentiment* score as per Equations 2 and 3. Then a hotel/item case is made up of the clustered features associated with its reviews and their corresponding importance and sentiment scores; see Equation 1. Note that $c_j \in reviews(h_i)$ means that any of the basic features in c_j are present in $reviews(h_i)$.

$$item(h_i) = \{(c_j, s(c_j, h_i), imp(c_j, h_i)) : c_j \in reviews(h_i)\} \tag{1}$$

The importance score of c_j, $imp(c_j, h_i)$, is the relative number of times that c_j (or rather its basic features) is mentioned in the reviews of hotel h_i.

$$imp(c_j, h_i) = \frac{\sum_{f_k \in c_j} count(f_k, h_i)}{|reviews(h_i)|} \tag{2}$$

The sentiment score of c_j, $s(c_j, h_i)$, is the degree to which c_j (that is, its basic features) is mentioned positively or negatively in $reviews(h_i)$. Note, $pos(f_j, h_i)$ and $neg(f_j, h_i)$ denote the number of mentions of the basic feature f_j labeled as positive or negative during the sentiment analysis phase.

$$s(c_j, h_i) = \frac{\sum_{f_k \in c_j} pos(f_k, h_i) - \sum_{f_k \in c_j} neg(f_k, h_i)}{\sum_{f_k \in c_j} pos(f_k, h_i) + \sum_{f_k \in c_j} neg(f_k, h_i)} \tag{3}$$

2.4 Generating User Profile Cases

Just as we generate hotel cases from the reviews written about a specific hotel we can generate user profile cases using the reviews written by a specific user, u_q. Equation 4 defines each user as a set of clustered features and each feature is

[1] http://glaros.dtc.umn.edu/gkhome/views/cluto

associated with an importance score based on how frequently that user refers to that feature in her reviews; this is exactly analogous to the hotel cases. However, unlike the hotel cases, we do not associate any sentiment with the user profile features. The reason for this is that the sentiment that a user expresses for a given feature is a property of the feature in the context of a specific hotel, rather than a general property of the feature; we will revisit this in future work.

$$user(u_q) = \{(c_j, imp(c_j, u_q)) : c_j \in reviews(u_q)\} \tag{4}$$

3 Ranking & Recommendation

Unlike traditional content-based recommenders — which tend to rely exclusively on similarity in order to rank products with respect to some user profile or query — our approach leverages feature sentiment *and* similarity, during recommendation. The starting point is the work of [15] which uses a linear combination of similarity and sentiment during recommendation ranking. Briefly, when it comes to recommending some candidate item i we can compute a recommendation score based on how similar the item is to the query q, and based on the sentiment of the item. And as per Equation 5 we can adjust the relative influence of similarity and sentiment using the parameter w.

$$Score(q, i) = (1 - w) \times Sim(q, i) + w \times Sent(q, i) \tag{5}$$

In the present work we use clustered features as the basis of our item descriptions, rather than the basic features used by [15]; we will usually refer to these clustered features as just *features* in what follows. We will also describe a personalized version of ranking, which harnesses user profiles that are *better* with respect to features that matter to the query user.

3.1 Query-Based Recommendation

To begin with we will implement a standard non-personalized ranking approach, similar to [15], albeit based on clustered features. We imagine an user u_q has a hotel in mind, h_q. Perhaps it is a hotel he has stayed in before and he is looking for something similar in a new city. We use h_q as a query and we compare it to candidate items h_c, computing the similarity and sentiment values to score each h_c for ranking and recommendation to u_q. For the purpose of similarity assessment we use a standard cosine metric; see [5]. Equation 6 demonstrates this for h_q and h_c. Note that we use the importance scores of shared features as the feature values.

$$Sim_h(h_q, h_c) = \frac{\sum\limits_{c_i \epsilon C(h_q) \cap C(h_c)} imp(c_i, h_q) \times imp(c_i, h_c)}{\sqrt{\sum\limits_{c_i \epsilon C(h_q)} imp(c_i, h_q)^2} \times \sqrt{\sum\limits_{c_i \epsilon C(h_c)} imp(c_i, h_c)^2}} \tag{6}$$

Next, we need to calculate the sentiment score for h_c. As mentioned earlier, sentiment information is unusual in a recommendation context but it makes it possible to consider not only how similar h_c is to h_q but also whether it enjoys better sentiment; it seems reasonable to recommend items that are not only similar to h_q but have also been more positively reviewed. We do this based on a feature-by-feature sentiment comparison as per Equation 7. We can say that c_i is *better* in h_c than h_q ($better(c_i, h_q, h_c) > 0$) if c_i in h_c has a higher sentiment score than it does in h_q.

$$better(c_i, h_q, h_c) = s(c_i, h_c) - s(c_i, h_q) \tag{7}$$

We calculate the sentiment score, $Sent(h_q, h_c)$ from the sum of these better scores for the features that are common to h_q and h_c as per Equation 8; we use $C(i)$ to denote the clustered features of item i.

$$Sent(h_q, h_c) = \frac{\sum_{c_i \in C(h_q) \cap C(h_c)} better(c_i, h_q, h_c) \times imp(c_i, h_c)}{|C(h_q) \cap C(h_c)|} \tag{8}$$

Accordingly we can implement a non-personalized scoring function based on the above by combining Sim_h and $Sent$ as per Equation 9.

$$Score_{QB}(h_q, h_c) = (1 - w) \times Sim_h(h_q, h_c) + w \times Sent(h_q, h_c) \tag{9}$$

3.2 User-Based Recommendation

Rather than using a specific item (h_q) as a query to trigger recommendations, another common recommendation use-case, is to use the user profile u_q as a query. To do this we need to implement a personalized version of the approach above, by introducing user preference information into both the similarity and the sentiment calculations.

Regarding similarity, we implement a version in which we use the importance weights from the query user u_q instead of the weights from h_q during similarity assessment as per Equation 10. In this way, features that are more important to u_q and h_c play a greater role in the similarity computation.

$$Sim_u(u_q, h_c) = \frac{\sum\limits_{c_i \epsilon C(u_q) \cap C(h_c)} imp(c_i, u_q) \times imp(c_i, h_c)}{\sqrt{\sum\limits_{c_i \epsilon C(u_q)} imp(c_i, u_q)^2} \times \sqrt{\sum\limits_{c_i \epsilon C(h_c)} imp(c_i, h_c)^2}} \tag{10}$$

Regarding sentiment, we propose Equation 11, which calculates a sentiment score based on the average sentiment of all hotels visited by u_q with h_c. We use $H(u_q)$ to denote the hotels visited by u_q. Thus, the sentiment score of h_c is influenced by u_q's history. Obviously, this is just one way that we might make the sentiment calculation more personalized for u_q. In this work it serves as a useful test-case and future work will consider alternative approaches.

$$Sent_u(u_q, h_c) = \frac{\sum_{h_q \in H(u_q)} Sent(h_q, h_c)}{|H(u_q)|} \quad (11)$$

For now, this means we can implement Equation 12 as the recommendation scoring function for generating personalized recommendations.

$$Score_{UB}(u_q, h_c) = (1 - w) \times Sim_u(u_q, h_c) + w \times Sent_u(u_q, h_c) \quad (12)$$

4 Evaluation

In this section, we will describe an initial evaluation of these query-based and user-based recommendation techniques using real-user data.

4.1 Datset & Setup

The dataset used in this work is based on the TripAdvisor dataset shared by the authors of [15]. This dataset covers 148,575 users for 1,701 hotels. We collected the member pages of these users from TripAdvisor website, we found that these users totally have written 1,008,585 hotel reviews until July, 2014; approximately 7 reviews per user, although some users have authored significantly more. For the purpose of this work, we focus on a subset of 1,000 users who have written at least 5 hotel reviews. This provides a test dataset of 11,993 reviews for 10,162 hotels. Finally, for each of these hotels, we collected up to its top 100 reviews for a total of 867,644 reviews; some hotels do not have 100 reviews.

For each of these users and hotels, we apply opinion mining to generate our feature-based descriptions. On average our test users have written 12 reviews resulting in profiles containing an average of 91 different clustered features. Likewise, the hotels are associated with an average of 89 reviews resulting in 189 clustered features per hotel on average. Clearly the opinion mining is capable of generating rich item descriptions. In what follows we will evaluate the query-based and user-based recommendation techniques using a standard leave-one-out style approach in which we attempt to recommend a specific target hotel given a query hotel or a user profile.

4.2 Evaluating Query-Based Recommendation

To evaluate the non-personalized, query-based recommendation strategy we produce a set of *test triples* of the form (u_q, h_q, h_t) corresponding to a query user u_q, a query hotel from u_q's profile and a target hotel, h_t, that is also in the user's profile. h_q and h_t are in different cities to simulate the use-case of the user looking for a hotel in some new city but based on a familiar hotel in a different city. For the purpose of this test we are careful to choose h_t from a set of 8 cities in our dataset which have sufficient candidate hotels that are also in our dataset(> 80). In each triple h_t is chosen only if it has been rated as 5-star by u_q; we assume the user is looking for a hotel they will like.

Furthermore, we note the user's rating for each of the query hotels in our test triples and distinguish between those query hotels that have a low (2-star or 3-star) rating by the user and those that have a much higher (4-star or 5-star) rating; we will refer to these as *23-queries* and *45-queries*, respectively. This allows us to compare the performance of the ranking based on how well the query user liked the query hotel; we might expect that it is easier to identify h_t if we are starting from a 45-query than a 23-query. In this sense, the 23-queries correspond to more a challenging query-based recommendation task than the 45-queries.

This provides us with 888 test triples; 705 have 45-queries and the remaining 183 have 23-queries. Each triple is a recommendation test, the objective of which is to locate h_t based on a ranking of hotels (from the same city as h_t) using h_q as a query. In other words, for each triple we use h_q as a query hotel and rank-order the other hotels in the same city as h_q using $Score_{QB}$, varying w to adjust the mix of similarity and sentiment.

The results are presented in Figure 3(a) as a graph of the top-20 hit-rate – the percentage of times the target hotel is within the top-20 recommendations – versus w. As we increase w (that is, increase the influence of sentiment) the hit-rate of the query-based algorithm improves for both 45- and 23-query groups. For example, at $w = 0$ (pure similarity-based scoring) we can see that the hit-rate is 0.20 for the 23-queries and 0.27 for the 45-queries. In other words h_t is in among the top-20 recommendations between 20% and 27% of the time. As expected, we can more successfully recommend the target hotel when using a 45-query than a 23-query.

As we increase w up to about 0.6-0.8 then this hit-rate increases to 0.35. That is, as we introduce more sentiment we improve hit-rates, from 27% (45-queries) to 35%, a relative increase of 30% compared to the similarity-only setting at $w = 0$. Indeed, the improvement is even more striking for the more challenging 23-queries: we see a relative improvement of 75% as hit-rate climbs from 20% to as high as 35% (at $w = 0.8$). Even when the query is a relatively poor starting point, as a 23-query is, the introduction of sentiment helps to produce a hit-rate that is comparable to the top hit-rate achieved by the 45-queries (with sentiment). Figure 3(a) also shows the average results across all queries (combining 23/45 query groups) where hit-rate climbs from 26% ($w = 0$) to 35% ($w = 0.6$) for a relative increase of about 35%.

There is a point after which more sentiment causes a disimprovement in hit-rate as the influence of similarity is no longer felt and sentiment tends to dominate. This point is $w = 0.6$ for 45-queries (and, on average, for all queries). It occurs later ($w = 0.8$) for the 23-queries. This suggests that sentiment should be relied on more when the query hotel is not a strong reflection of a user's true interests, as is the case with 23-queries. This makes sense: if these query hotels are not a good reflection of a user's true interests then it may not be appropriate to rely on similarity, it would be much better to focus on recommendations that are better than the query hotels by as much as possible.

For comparison, in Figure 3(b) we show the relative ranking of the target hotels in the recommendation lists. In this case, a *lower* relative ranking score means that the target hotel appears *higher* in the recommendation rankings. For example, across all queries we see that the relative ranking of the target hotel falls from being in the top 37% of recommendations to the top 30% of recommendations as w is increased up to 0.6. These relative ranking results are entirely consistent with the previous hit-rate results. They show an improvement in recommendation performance with increasing w up to $w = 0.6 - 0.8$ depending on the query set. Once again we see improved recommendation performance for the 45-queries compared to the tougher 23-queries, but the introduction of sentiment dramatically improves recommendation even for these 23-queries.

So far we have been looking for a single target hotel. That is not unusual in these types of offline, leave-one-out tests, but the results remain silent when it comes to the quality of the other recommendations. For example, what is the average rating of the top-20 recommended hotels? And how does this change with w? This is important because it tells us about the overall quality of the recommendation lists produced. This information is shown in Figure 3(c) as the average TripAdvisor rating for the top-20 recommendations as we increase w.

Once again we can see that there is a benefit when it comes to introducing sentiment: as we increase w the average rating of the recommended hotels increases from about 4-stars to over 4.5-stars; at $w = 0.6$, the turning point for hit-rate and relative ranking, the average rating is almost exactly 4.5-stars. As expected, we can see that the average rating for the 'easier' 45-queries is slightly higher than the average rating for the 23-queries. Note that there is no turning point on this average rating graph, as there was previously. The reason for this is that as we increase w we are guiding recommendation towards hotels with more and more positive features and so we can expect their average ratings to increase accordingly. And as they do, the average ratings is up to a maximum of about 4.6-stars, which presumably is at, or close to, the maximum possible rating for a list of 20 hotels given the TripAdvisor rating distribution. Obviously, this does not mean that we should turn-up w to its maximum level because if we do then there is a cost when it comes to retrieving a particular target case. And these results suggest a balancing point of $w = 0.6$ is close to optimal for this type of query-based recommendation, at least in this domain.

4.3 Evaluating User-Based Recommendation

We follow a similar approach to evaluate the personalized, user-based recommendation strategy. This time we use a set of user-item *test pairs*, each containing a user profile u_q, as a query, and a target item h_t. In each case h_t is one of the hotels that u_q has previously rated as 5-stars. This time our recommendation test will be to use u_q as a query to recommend h_t.

There are 665 of these test pairs. Earlier we distinguished between hard and easy queries, we do similar here by dividing user profiles into *small*, *medium*, and *large* based on their number of reviews; see Table 1. For example, small profiles have up to 10 reviews and there are 274 pairs from these profiles involving

249 unique hotels and 211 unique users. The intuition is that small profiles will represent a tougher user-based recommendation test than medium or large profiles.

Table 1. Testing Pairs for User-Based Recommendation

	Pairs	Hotels	Users
All	665	526	461
Small ($<= 10$)	274	249	211
Medium ($11 - 20$)	238	215	166
Large (> 20)	153	142	84

Each test pair defines a recommendation test in which u_q is used to rank and recommend hotels from the same city as h_t using $Score_{UB}$. And in this test, for each test pair, we re-generate u_q without reviews from target hotel.

We calculate the top-20 hit-rate for different values of w, and the results are presented in Figure 4 (a). As before we can see how increasing w tends to improve hit-rate. For example, at $w = 0$ the hit-rate for large profiles is about 30% and this grows to 40% (at $w = 0.4$), a relative improvement of about 33%. A similar effect is noted for medium and small profiles but, as expected, the size of the effect is reduced.

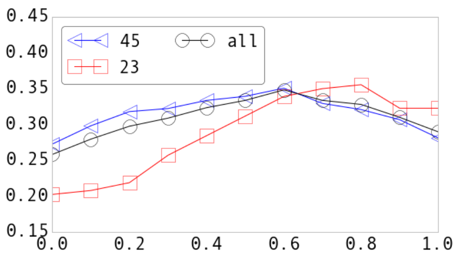
Once again there is an optimal w in range of $w = 0.4 - 0.5$, that seems to deliver an optimum hit-rate. Beyond this, as sentiment dominates, the hit-rate falls sharply, eventually dropping below the hit-rate achieved at $w = 0$ (similarity only).

Relative ranking results are also presented in Figure 4 (b). They are consistent with the hit-rate results, showing a benefit to increasing w up to about 0.4-0.5 and with the best relative ranking accruing to the larger profiles.

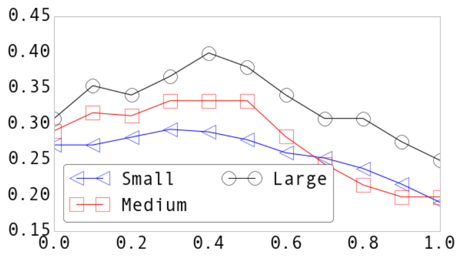
For completeness, we also show the average TripAdvisor rating for the full set of 20 recommendations as w varies. Once again we see a gradual increase in recommendation quality for increasing w; the average rating of recommendation lists increases from 4-stars to about 4.6-stars.

4.4 Discussion

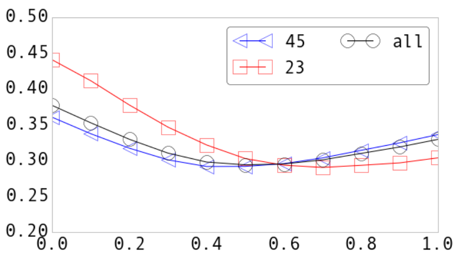
We have evaluated the performance of our query-based and user-based recommendation approaches on real-user data from TripAdvisor. In each case, we have explored the benefit of using sentiment during recommendation for queries of different difficulty levels. The results have been very consistent across a number of evaluation metrics. In each case, increasing sentiment can significantly improve recommendation performance to a point. Although it is likely that the best level of sentiment to include will likely be domain and task dependent, it is equally likely, we believe, that selecting a reasonable value for w, such as $w = 0.5$, should deliver significant recommendation improvements over more conventional



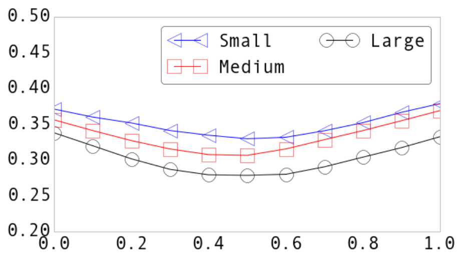
(a) Top 20 Hit-Rate.



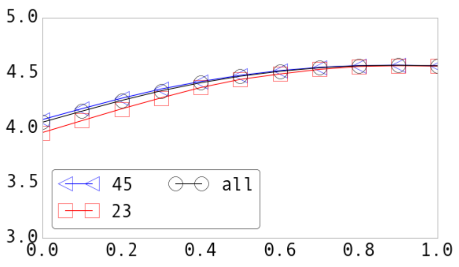
(a) Top 20 Hit-Rate.



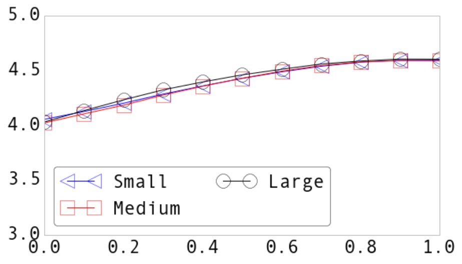
(b) Relative Ranking.



(b) Relative Ranking.



(c) Top 20 Ave. Rating.



(c) Top 20 Ave. Rating.

Fig. 3. Query-based recommendation.

Fig. 4. User-based Recommendations.

similarity-based techniques. Fine-tuning might improve this further but it may not be necessary or particularly worthwhile in many settings.

5 Conclusions

This paper builds on recent work using product reviews as a novel source of recommendation knowledge; see [15]. Our main contribution has been to propose and evaluate a new, personalized, user-based recommendation approach that is capable of generating proactive recommendations for a user based on their mined preferences. In the process, we have modified the work of [15] by introducing tri-gram features and a feature-clustering step during opinion mining in order to better capture the relationship between basic features.

We have evaluated these approaches in an offline study using real-user evaluation data from TripAdvisor. The results show the benefits of mixing sentiment and similarity during recommendation in both query-based and user-based recommendation scenarios. Profile size is important and the size of the effect on recommendation performance is closely connected to the number of reviews that have contributed to a profile.

Extending the evaluation to larger datasets and/or new domains is one priority for future work. We also plan to explore new approaches to opinion mining by using topic-modeling techniques to better understand hotel preferences and trip purposes. Our intuition is that the probability distribution of topics in a review is a mixture of the distributions associated with the user interests, travel purpose, and the hotel features. We are currently exploring ways to learn these topic models from review texts and incorporate this into recommendations.

6 Acknowledgments

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

References

1. J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, “Collaborative filtering recommender systems,” in *The Adaptive Web* (P. Brusilovsky, A. Kobsa, and W. Nejdl, eds.), vol. 4321 of *Lecture Notes in Computer Science*, pp. 291–324, Springer Berlin Heidelberg, 2007.
2. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “Grouplens: An open architecture for collaborative filtering of netnews,” in *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW ’94, (New York, NY, USA), pp. 175–186, ACM, 1994.
3. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Item-based collaborative filtering recommendation algorithms,” in *Proceedings of the 10th International Conference on World Wide Web*, WWW ’01, (New York, NY, USA), pp. 285–295, ACM, 2001.
4. Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, no. 8, pp. 30–37, 2009.
5. M. J. Pazzani and D. Billsus, “Content-based recommendation systems,” in *The adaptive web*, pp. 325–341, Springer, 2007.
6. D. Bridge, M. H. Göker, L. McGinty, and B. Smyth, “Case-based recommender systems,” *Knowl. Eng. Rev.*, vol. 20, pp. 315–320, Sept. 2005.
7. B. Smyth, “Case-based recommendation,” in *The Adaptive Web, Methods and Strategies of Web Personalization*, pp. 342–376, 2007.
8. R. Burke, “Hybrid recommender systems: Survey and experiments,” *User modeling and user-adapted interaction*, vol. 12, no. 4, pp. 331–370, 2002.
9. H. Wang, B. Chen, and W.-J. Li, “Collaborative topic regression with social regularization for tag recommendation.,” in *IJCAI*, 2013.
10. H. Wang and W.-J. Li, “Relational collaborative topic regression for recommender systems,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 27, no. 5, pp. 1343–1355, 2015.

11. Š. Pero and T. Horváth, “Opinion-driven matrix factorization for rating prediction,” in *User Modeling, Adaptation, and Personalization*, pp. 1–13, Springer, 2013.
12. G. Ganu, N. Elhadad, and A. Marian, “Beyond the stars: Improving rating predictions using review text content.,” in *WebDB*, vol. 9, pp. 1–6, Citeseer, 2009.
13. B. Liu, “Sentiment analysis and opinion mining,” *Synthesis Lectures on Human Language Technologies*, vol. 5, no. 1, pp. 1–167, 2012.
14. L. Chen, G. Chen, and F. Wang, “Recommender systems based on user reviews: the state of the art,” *User Modeling and User-Adapted Interaction*, vol. 25, no. 2, pp. 99–154, 2015.
15. R. Dong, M. P. O’Mahony, and B. Smyth, “Further Experiments in Opinionated Product Recommendation,” in *Proceedings of The 22nd International Conference on Case-Based Reasoning*, (Cork, Ireland), pp. 110–124, 2014.
16. S. Aciar, D. Zhang, S. Simoff, and J. Debenham, “Informed recommender: Basing recommendations on consumer product reviews,” *Intelligent Systems, IEEE*, vol. 22, no. 3, pp. 39–47, 2007.
17. C.-C. Musat, Y. Liang, and B. Faltings, “Recommendation using textual opinions,” in *IJCAI International Joint Conference on Artificial Intelligence*, IJCAI ’13, pp. 2684–2690, AAAI Press, 2013.
18. H. Liu, J. He, T. Wang, W. Song, and X. Du, “Combining user preferences and user opinions for accurate recommendation,” *Electronic Commerce Research and Applications*, vol. 12, no. 1, pp. 14–23, 2013.
19. F. Wang and L. Chen, “Review mining for estimating users ratings and weights for product aspects,”
20. R. Dong, M. Schaal, M. P. O’Mahony, and B. Smyth, “Topic Extraction from Online Reviews for Classification and Recommendation,” in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, IJCAI ’13, pp. 1310–1316, AAAI Press, 2013.
21. R. Dong, M. P. O’Mahony, M. Schaal, K. McCarthy, and B. Smyth, “Combining similarity and sentiment in opinion mining for product recommendation,” *Journal of Intelligent Information Systems*, pp. 1–28, 2015.
22. J. S. Justeson and S. M. Katz, “Technical terminology: some linguistic properties and an algorithm for identification in text,” *Natural language engineering*, vol. 1, no. 01, pp. 9–27, 1995.
23. M. Hu and B. Liu, “Mining opinion features in customer reviews,” in *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI ’04, pp. 755–760, AAAI Press, 2004.