User-based Opinion-based Recommendation*

Ruihai Dong, Barry Smyth
Insight Centre for Data Analytics
School of Computer Science, University College Dublin, Ireland
{firstname.lastname}@ucd.ie

Abstract

User-generated reviews are a plentiful source of user opinions and interests and can play an important role in a range of artificial intelligence contexts, particularly when it comes to recommender systems. In this paper, we describe how natural language processing and opinion mining techniques can be used to automatically mine useful recommendation knowledge from user generated reviews and how this information can be used by recommender systems in a number of classical settings.

1 Introduction

Intelligent web-based systems have, for many years, taken advantage of AI techniques in a variety of online settings. Recommender systems are a case in point, bringing together ideas from knowledge representation, machine learning, and intelligent user interfaces to help match the right user with the right information at the right time.

In this paper, we explore the benefits of a recent source of recommendation knowledge – the opinions of users expressed in user generated reviews – for the purpose of recommendation. Historically, different types of recommendation techniques rely on different types of information and different sources of data. For example, collaborative filtering [Schafer *et al.*, 2007; Resnick *et al.*, 1994; Sarwar *et al.*, 2001; Koren *et al.*, 2009], rely on product ratings. whereas content and case-based techniques [Pazzani and Billsus, 2007; Bridge *et al.*, 2005; Smyth, 2007] rely on product meta-data and features (type, price, size etc). However, product data can be hard to come by while rating data is notoriously sparse. This has led researchers to develop hybrid techniques [Burke, 2002] and approaches to harness alternative auxiliary information [Wang and Li, 2015].

The recent proliferation of user generated reviews serve as a new source of recommendation information. Reviews contain important product knowledge and valuable customer insights for use in recommendation [Pero and Horváth, 2013; Ganu *et al.*, 2009]. But reviews are unstructured, and noisy, and they can even be biased. Nevertheless, opinion mining and natural language processing techniques [Liu, 2012] are now sufficiently mature for such content to act as an alternative (or complementary) source of recommendation knowledge [Chen *et al.*, 2015]. User-generated reviews are plentiful and they contain rich product feature information including sentiment information (e.g. *The desserts were beautiful, ..., but the wine was over-priced*).This makes it possible to generate recommendations that are *similar* to those a user has liked in the past, but also *better* based on features that matter to the user; see [Dong and Smyth, 2016; Dong *et al.*, 2014; Aciar *et al.*, 2007]. Musat et al. [Musat *et al.*, 2013] 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. [Liu *et al.*, 2013] 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 [Wang and Chen, 2015].

In this work, we consider two classical recommendation scenarios in a hotel recommendation setting. The work is an abridged version of a more detailed recent article by [Dong and Smyth, 2016].

2 Mining Profiles & Products

Extending the work of [Dong *et al.*, 2013; 2014; 2015] we create user profiles and item/product descriptions from reviews as summarized below.

2.1 Extracting Features & Sentiment

The feature mining and sentiment analysis approaches used in this work are based on the techniques described by [Dong *et al.*, 2014]. We mine *bi-gram* (adjective-noun) and *single-noun* features using shallow NLP methods. In the present work we also consider additional *tri-gram* features, as described in the work of [Justeson and Katz, 1995], to allow us to find features like *member of staff*, *bottle of water*.

For each extracted (basic) feature, f_i, from some sentence s_j in review r_k, we evaluate its sentiment by finding the closest *sentiment word* w_{min} to f_i in s_j using a sentiment lexicon [Hu and Liu, 2004]. Thus for each review we generate a set of features and their sentiment labels.

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

```
Cluster 1 - bill, charge, fee
Cluster 2 - burger,pizza,grill
Cluster 3 - cake, afternoon tea, ...
Cluster 4 - bar, beer, drink, cocktail
Cluster 5 - bed, pillow, mattress
Cluster 6 - stair, lift, elevator
```

Figure 1: Examples for Clustered Features.

2.2 Feature Clusters

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*. This tends to produce a proliferation of features. Rather than treating these as separate features, it is more natural to recognize that they are referring to the same aspect of a hotel.

To do this we apply clustering techniques to group related features 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 1. 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 Item/Product Cases

Each item/hotel (h_i) is associated with a set of reviews $rs(h_i) = \{r_1, \ldots, r_n\}$ and the above processes extract a set of features and clusters, $c_1, \ldots, c_m$, from these reviews. Each cluster is comprised of a set of basic features and acts as a type of abstract feature, a *clustered feature*, with the basic features it contains related in some way. For example, in Figure 1 we can see that the features in clusters 2, and 3 are related to food as *dinner*, *lunch* respectively. 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 $c_j \in rs(h_i)$ means that any of the basic features in c_j are present in $rs(h_i)$.

$$item(h_i) = \{(c_j, s(c_j, h_i), i(c_j, h_i)) : c_j \in rs(h_i)\} \quad (1)$$

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

$$i(c_j, h_i) = \frac{\sum_{f_k \in c_j} count(f_k, h_i)}{|rs(h_i)|} \quad (2)$$

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

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 $rs(h_i)$. Note, $pos(f_k, h_i)$ and $neg(f_k, h_i)$ denote the number of mentions of the basic feature f_k labeled as positive or negative during the sentiment analysis phase.

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

2.4 Generating User Profiles

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 as in Equation 4.

$$user(u_q) = \{(c_j, i(c_j, u_q)) : c_j \in rs(u_q)\} \quad (4)$$

3 Ranking & Recommendation

In this section, we introduce how these user and product profiles can be used in various recommendation tasks by combining sentiment and similarity in both query-based and user-based recommendation scenarios.

3.1 Query-Based Recommendation

To begin we implement a standard non-personalized ranking approach, similar to [Dong *et al.*, 2014], albeit based on clustered features. Consider a user u_q with a hotel in mind, h_q. Perhaps a hotel she has stayed in previously and she wants something similar in a new city. Let h_q be the query and 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 [Pazzani and Billsus, 2007]l; see Equation 5 and note that we use the importance scores of shared features as the feature values.

$$Sim_h(h_q, h_c) = \frac{\sum_{c_i \epsilon C(h_q) \cap C(h_c)} i(c_i, h_q) \times i(c_i, h_c)}{\sqrt{\sum_{c_i \epsilon C(h_q)} i(c_i, h_q)^2} \times \sqrt{\sum_{c_i \epsilon C(h_c)} i(c_i, h_c)^2}} \quad (5)$$

Next, we calculate the sentiment score for h_c. Sentiment information makes it feasible to consider not only how similar h_c is to h_q but also whether it enjoys better sentiment; then, we can 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 6. 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.

$$better(c_i, h_q, h_c) = s(c_i, h_c) - s(c_i, h_q) \quad (6)$$

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 7; we use $C(i)$ to denote the clustered features of item i.

$$Sent(h_q, h_c) = \frac{\sum_{c_i \in C(h_q) \cap C(h_c)} better(c_i, h_q, h_c) \times i(c_i, h_c)}{|C(h_q) \cap C(h_c)|} \quad (7)$$

Accordingly, we can implement a non-personalized scoring function based on the above by combining Sim_h and $Sent$ as per Equation 8.

$$Score_{QB}(h_q, h_c) = (1-w) \times Sim_h(h_q, h_c) + w \times Sent(h_q, h_c) \quad (8)$$

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 9. In this way, features that are more important to u_q and h_c play a greater role in the similarity computation.

$$Sim_u(u_q, h_c) = \frac{\sum\limits_{c_i \epsilon C(u_q) \cap C(h_c)} i(c_i, u_q) \times i(c_i, h_c)}{\sqrt{\sum\limits_{c_i \epsilon C(u_q)} i(c_i, u_q)^2} \times \sqrt{\sum\limits_{c_i \epsilon C(h_c)} i(c_i, h_c)^2}} \quad (9)$$

Regarding sentiment, we propose Equation 10, 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.

$$Sent_u(u_q, h_c) = \frac{\sum_{h_q \in H(u_q)} Sent(h_q, h_c)}{|H(u_q)|} \quad (10)$$

For now, this means we can implement Equation 11 as the recommendation scoring function for generating personalized recommendations.

$$Score_{UB}(u_q, h_c) = (1-w) \times Sim_u(u_q, h_c) + w \times Sent_u(u_q, h_c) \quad (11)$$

4 Evaluation

The dataset used in this work is based on the TripAdvisor dataset [Dong *et al.*, 2014]. 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 with 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 if it has for a total of 867,644 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. We will evaluate the *query-based* and *user-based* recommendation by using a standard leave-one-out style approach.

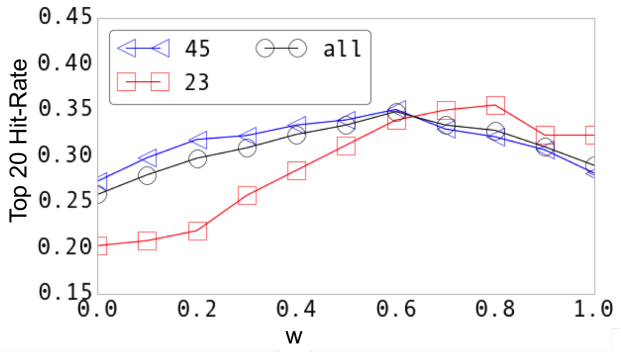
4.1 Evaluating Query-Based Recommendation

To evaluate the non-personalized, query-based recommendation strategy we use a set of *test triples* of the form (u_q, h_q, h_t) corresponding to a query user u_q, a query hotel, h_q, and a target hotel, h_t. h_q and h_t are in the u_q's profile, but in different cities to simulate the user looking for a hotel in some new city but based on a familiar hotel. For the purpose of this test, we 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.

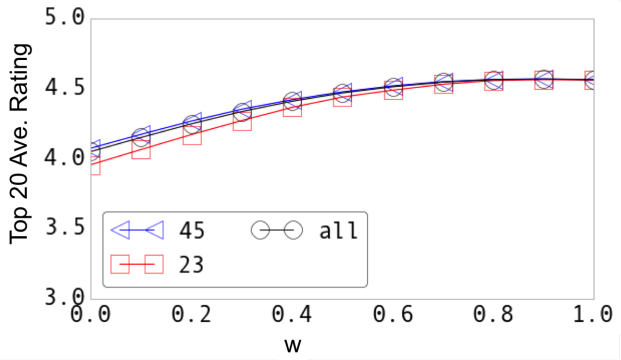
We note the user's rating for each query hotel in our test triples and distinguish between those with a low (2 or 3-star) rating and those that have higher (4 or 5-star) ratings; we call these *23-queries* and *45-queries*, respectively. This allows us to compare the ranking based on how well the query user liked the query hotel; we might expect it to be easier to identify h_t from a 45-query than a 23-query. 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.

The results are presented in Figure 2(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. 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. As expected, we can more successfully recommend the target hotel when using a 45-query than a 23-query. As we increase w (adding sentiment) the hit-rate of the query-based algorithm improves for both 45- and 23-query groups, for w up to 0.6-0.8; hit-rate increases 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 $w = 0.8$). 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.

We have been looking for a single hotel but the results ignore the quality of the other recommendations. For example, what is the average rating of other hotels in the top-20 recommended? And how does this change with w? This tells us about the overall quality of the recommendations and is shown in Figure 2(b) as the average TripAdvisor rating for the top-20 recommendations for increasing w. We see there is a benefit to introducing sentiment: as we increase w the av-

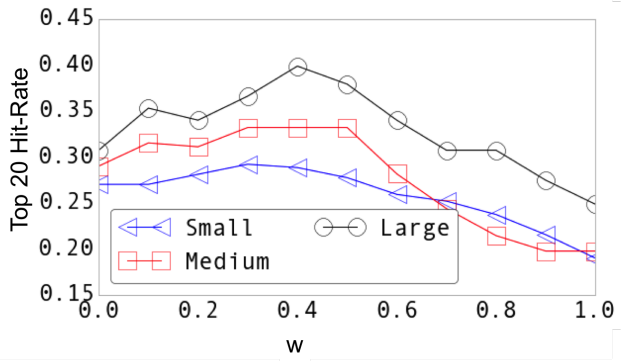


(a) Top 20 Hit-Rate.

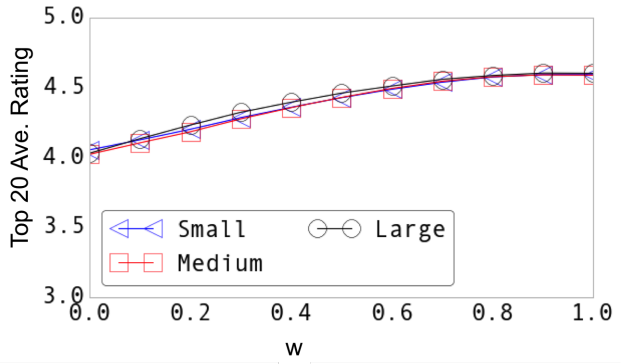


(b) Top 20 Ave. Rating.

Figure 2: Query-based recommendation.



(a) Top 20 Hit-Rate.



(b) Top 20 Ave. Rating.

Figure 3: User-based Recommendations.

erage rating of the recommended hotels increases from about 4-stars to over 4.5-stars. We can see that the average rating for the 'easier' 45-queries is higher than the average rating for the 23-queries but there is no turning point on this average rating graph because we are guiding recommendation towards hotels with more and more positive features and so we can expect their average ratings to increase accordingly.

4.2 Evaluating User-Based Recommendation

We follow a similar approach to evaluate the personalized, user-based 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 a hotel that u_q has previously rated as 5-stars. Our recommendation test will use u_q as a query to recommend h_t. There are 665 of these test pairs. We divide profiles into *small*, *medium*, and *large* based on their number of reviews. *small* profiles have up to 10 reviews and there are 274 pairs from these with 249 unique hotels and 211 unique users; *medium* profiles between 11 and 20 reviews, and there are 238 pairs from these involving 215 hotels and 166 users; *large* profiles have more than 20 reviews and there are 153 pairs with 142 hotels and 84 users. The intuition is that small profiles will represent a tougher user-based recommendation test than medium or large profiles.

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 regenerate 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 3 (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. There is an optimal w in the 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).

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.

5 Conclusions

This work (based on [Dong and Smyth, 2016]) describes and evaluates a recommendation approach to generate recommendations based on preferences mined from product reviews using a combination of shall NLP and statistical data-mining techniques. Using TripAdvisor data we show the benefits of mixing sentiment and similarity during recommendation in both query-based and user-based recommendation scenarios.

References

[Aciar *et al.*, 2007] Silvana Aciar, Debbie Zhang, Simeon Simoff, and John Debenham. Informed recommender: Basing recommendations on consumer product reviews. *Intelligent Systems, IEEE*, 22(3):39–47, 2007.

[Bridge *et al.*, 2005] Derek Bridge, Mehmet H. Göker, Lorraine McGinty, and Barry Smyth. Case-based recommender systems. *Knowl. Eng. Rev.*, 20(3):315–320, September 2005.

[Burke, 2002] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[Chen *et al.*, 2015] Li Chen, Guanliang Chen, and Feng Wang. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction*, 25(2):99–154, 2015.

[Dong and Smyth, 2016] Ruihai Dong and Barry Smyth. Personalized opinion-based recommendation. In *Case-Based Reasoning Research and Development - 24th International Conference, ICCBR 2016, Atlanta, GA, USA, October 31 - November 2, 2016, Proceedings*, pages 93–107, 2016.

[Dong *et al.*, 2013] Ruihai Dong, Markus Schaal, Michael P. O'Mahony, and Barry Smyth. Topic Extraction from Online Reviews for Classification and Recommendation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1310–1316. AAAI Press, 2013.

[Dong *et al.*, 2014] Ruihai Dong, Michael P O'Mahony, and Barry Smyth. Further Experiments in Opinionated Product Recommendation. In *Proceedings of The 22nd International Conference on Case-Based Reasoning*, pages 110–124, Cork, Ireland, 2014.

[Dong *et al.*, 2015] Ruihai Dong, Michael P O'Mahony, Markus Schaal, Kevin McCarthy, and Barry Smyth. Combining similarity and sentiment in opinion mining for product recommendation. *Journal of Intelligent Information Systems*, pages 1–28, 2015.

[Ganu *et al.*, 2009] Gayatree Ganu, Noemie Elhadad, and Amélie Marian. Beyond the stars: Improving rating predictions using review text content. In *WebDB*, volume 9, pages 1–6. Citeseer, 2009.

[Hu and Liu, 2004] Minqing Hu and Bing Liu. Mining opinion features in customer reviews. In *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI '04, pages 755–760. AAAI Press, 2004.

[Justeson and Katz, 1995] John S Justeson and Slava M Katz. Technical terminology: some linguistic properties and an algorithm for identification in text. *Natural language engineering*, 1(01):9–27, 1995.

[Koren *et al.*, 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[Liu *et al.*, 2013] Hongyan Liu, Jun He, Tingting Wang, Wenting Song, and Xiaoyang Du. Combining user preferences and user opinions for accurate recommendation. *Electronic Commerce Research and Applications*, 12(1):14–23, 2013.

[Liu, 2012] Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

[Musat *et al.*, 2013] C-C Musat, Yizhong Liang, and Boi Faltings. Recommendation using textual opinions. In *IJCAI International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 2684–2690. AAAI Press, 2013.

[Pazzani and Billsus, 2007] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[Pero and Horváth, 2013] Štefan Pero and Tomáš Horváth. Opinion-driven matrix factorization for rating prediction. In *User Modeling, Adaptation, and Personalization*, pages 1–13. Springer, 2013.

[Resnick *et al.*, 1994] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[Sarwar *et al.*, 2001] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[Schafer *et al.*, 2007] J.Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, pages 291–324. Springer Berlin Heidelberg, 2007.

[Smyth, 2007] Barry Smyth. Case-based recommendation. In *The Adaptive Web, Methods and Strategies of Web Personalization*, pages 342–376, 2007.

[Wang and Chen, 2015] Feng Wang and Li Chen. Review mining for estimating users ratings and weights for product aspects. 2015.

[Wang and Li, 2015] Hao Wang and Wu-Jun Li. Relational collaborative topic regression for recommender systems. *Knowledge and Data Engineering, IEEE Transactions on*, 27(5):1343–1355, 2015.