REVIEW RATING ADJUSTMENT TO INCORPORATE USER PREFERENCES

Jonah Rubin[1], Honglei Liu[2], Xifeng Yan[2]
[1]*Ridley College*
St. Catharines, ON, Canada L2R 7C3
email address: jonah@innolution.com
[2]*Department of Computer Science, UC Santa Barbara*
CA 93106, USA

Online reviews are an excellent resource for modern consumers. But for all their benefits, the sheer number of available reviews limits the ability of a single person to utilize this resource effectively. In this paper, we propose a fast and effective method for adjusting star ratings to better reflect a consumer's priorities. Using a combinatorial algorithm, we create a system that takes in a set of user "aspect weightings" detailing what the user cares about, and creates a weighted star rating by giving higher influence to reviews that display similar priorities to the user. We test the algorithm on a dataset of hotel reviews crawled from TripAdvisor. Empirical results from this comparison reveal that our solution is significantly faster than a state-of-the-art algorithm, with comparable accuracy.

Keywords: Online Reviews, Review Mining, Combinatorial Algorithms, Aspect Analysis

I. INTRODUCTION

As the web grows increasingly interactive, so too have online reviews grown more prevalent. While they can be an excellent resource, the sheer number of online reviews available makes sifting through all of them a difficult if not impossible task. An average star rating system that summarizes a set of reviews with an overall average, as employed by services such as TripAdvisor, reduces this burden somewhat. However, these more simplistic approaches fail to take into account a user's priorities. For example, a traveller looking for a business hotel would care far more about a hotel's business service and value than the quality of the grounds. Thus, a hotel with a low star rating, but whose reviewers stated it had good value and business service would be more attractive to this particular user than a higher-rated hotel that was rated that way because of its grounds.

To solve this particular problem, one solution would be to give the user information from reviews that have the same priorities. In the aforementioned example, the user more interested in value and business service would be better served by being shown reviews that focus primarily on the value and business service. To effectively perform this task, however, we must be able to determine from the text of reviews the priorities of their respective authors. A significant amount of time and effort has already been dedicated to text analysis and aspect modeling. However, a vast majority of these approaches do much more than simple aspect analysis, and are thus burdened by their scope, and execute too slowly to be useful for the average user.

In this paper, we propose a fast, and effective method of identifying reviewer priorities. First, we construct a network of interconnected words, linking them by the number of times they co-occur in any sentence across the entire corpus. Then, using a number of seedwords that are already assigned to certain predefined aspects, we allow the influence of these aspects to propagate though the network, effectively assigning each word a distribution of how it relates to each aspect (e.g. 50% to value, 10% to room, etc.). By averaging these distributions for each word in each sentence, and each sentence in each review, we are able to generate the weighting distributions required by the problem. We then embed this algorithm in a system that we call Review Weighting Analysis Comparator (RWAC), which adjusts the overall star ratings for hotels using a weighted average derived by giving greater influence to reviews whose reviewers have similar priorities to the user. The key assumption is that the degree to which each aspect influenced a particular reviewer's overall rating is proportional to the amount of text the reviewer allocated to discussing that aspect in his review. Thus, if a reviewer spent 70% of his review discussing the room, and 30% discussing the service, we assume that his overall rating for the hotel is 70% based on his opinion of the room, and 30% based on his opinion of the service. To incorporate this assumption, we create a multiplier for each review based on how closely its aspect weightings match the user's aspect weightings, and use them as the weightings for a weighted average of all of a hotel's star ratings. This approach ensures that the reviewers who have similar priorities to the user will have a larger say on the final result.

II. TERMINOLOGY

This section defines the important terms used throughout this paper.

Aspect: predefined concepts upon which we assume reviewers base their opinions. Each aspect is associated with an identifying word. For hotel reviews, we define them as "Value," "Room," "Location," "Cleanliness," and "Service."

Seedword: a predefined, one-to-one relationship between a word and an aspect. For example, the seedword "dirty" corresponds to the aspect "Cleanliness."

Priorities: the personal preferences that reviewers and users have that determine how much they care about each aspect. For example, a reviewer may place higher priority on room quality than on hotel location.

Weightings: the explicit percentages of text in a review or sentence that discuses certain aspects, represented as a percentage distribution. We assume these weightings reflect the reviewer's priorities.

III. RELATED WORK

A significant amount of effort has already been dedicated to the problem of aspect analysis. Early attempts focused on identifying particular noun phrases **[1]**. This approach works well when aspects tend to be characterized by a particular noun, but does poorly when confronted by aspects that tend to be expressed through the usage of many nouns. For example, while the aspect "Room" could be detected by looking for certain nouns (bed, bathroom, etc.), an aspect such as "Location", which may be indirectly referred to via a variety of nearby attractions and travel methods, would be much harder to spot. Later endeavors used more robust and general topic modeling systems, such as Latent Dirichlet Allocation (LDA) **[2]**. However, such systems require statistical inference to function, and are thus very slow.

Our system is distinguished from many modern solutions **[3]** by the fact that it does not use sentiment analysis, which we argue is unnecessary for this problem, as our system deals exclusively in weightings. Even though it uses a sentiment-based system, the best baseline for comparison is the seedword-based Latent Aspect Regression Analysis (LARA) system, as proposed in **[4]**. As a state-of-the-art existing algorithm, we use LARA's results as a baseline for comparing the results of our aspect weighting system. Our tests were based on LARA's included dataset of hotel reviews crawled from TripAdvisor.

IV. METHOD

Our algorithm is divided into three primary components: the preprocessor, the aspect weighting analyzer and the star rating adjuster. The preprocessor prepares the corpus for analysis. The weighting analyzer is responsible for discovering the weightings of each review and returning a percentage distribution of how that particular review relates to each aspect. The star rating adjuster takes these distributions and calculates a new average star rating by comparing each review's weightings to the user's weightings

1. Preprocessor

Before either of the primary components can be run, the data is preprocessed. First, the Punkt Sentence and Word tokenizers **[5]** are used to divide each review into its component sentences. Next the stopwords as defined in **[6]** are removed from the text. These stopwords are mainly words that cannot relate to any aspect, such as pronouns, article adjectives, and conjunctions. Finally, the data is run through the Porter Stemmer **[7]** to ensure that various forms and tenses of words are roughly comparable. The returned corpus consists of a list of reviews, each containing a list of tokenized sentences, containing lists of tokenized words, including only those words that are useful for computation.

2. Aspect Weighting Analyzer (AWA)

This component of the algorithm discovers the inherent aspect weightings of each word, sentence, and review in the corpus. The system starts by creating a network of interconnected words, where every pair of words has one link for every time they co-occur. An example is shown in **Figure 1**. AWA then determines the probability that a given word "belongs" to each aspect. The result is represented as a percentage distribution across the aspects. The sum of any distribution is always 1. Before the algorithm is run, a set of seedwords is predefined to correspond to one and only one aspect. They receive a weighting of 1 for the aspect they are assigned to, and 0 for all others.

The system then initiates what we call Multiple Anchor Association Cascade (MAAC) throughout the network. All seedwords are considered "already processed" (AP) at the start. During each MAAC iteration, any word that is not AP and has at least five links to words that are AP is processed. The weighting ww_x for any word w, for any aspect x, is given by

$$ww_x = \frac{\sum_{l \in L} l_x}{\sqrt[3]{N_x} * \sum_{a \in A} \sum_{l \in L} l_a} \quad (1)$$

where l is a link between word w and another word contained within list L of all words that are both linked to w

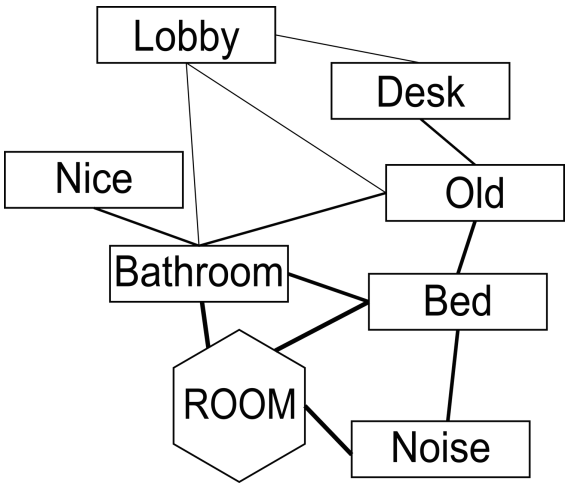

Figure 1. An example of a set of words, and an aspect in an association network. The thicknesses of the links show the number of times the two linked words co-occur. The words "bathroom," "bed," and "noise" are seedwords, and thus directly connected to the aspect, "Room."

and AP, a is an aspect in the list of aspects A, and N_a is the number of seedwords associated with aspect a. This processes iterates until all words are AP, or all remaining non-AP words have less than 10 AP links.

For each sentence, the weighting ws_x for any sentence S, and for any aspect x, is given by

$$ws_x = \frac{\sum_{w \in S} ww_x}{\sum_{x \in A} \sum_{w \in S} ww_x} \quad (2)$$

These weightings are then revised. The highest value is automatically included in the revised average. The second highest percentage is included if it is within 10% of the highest, the third if it is within 10% of the second, and so on. The weighting percentages are then recalculated by averaging only the included values. Each review-aspect distribution is calculated by averaging the percentage weightings from each sentence, in the same manner as the first step of the sentence-distribution discoverer. AWA returns a list of reviews and their corresponding percentage distributions.

3. Star Rating Adjuster (SRA)

The purpose of the SRA is to take in a user-supplied distribution of priorities, and to adjust the overall hotel star rating based on these priorities. Prior to program execution, the user provides a distribution of aspect weightings similar to the ones produced by the AWA system. This distribution reflects how much they value the various aspects. The SRA component calculates each weighting by calculating the Kullback-Leibler Divergence from the user-supplied preference distribution to the distribution of each review, and using that as the weighting for the weighted average. That is to say, the adjusted star rating St_R for hotel R is calculated in terms of the weighting for each review, W_r, such that

$$St_R = \frac{\sum_{r \in R} (W_r * St_r)}{\sum_{r \in R} W_r}$$
$$W_r = \frac{1}{\sum_{x \in A} \ln(\frac{U_x}{r_x}) r_x} \quad (3)$$

where review r is a review for hotel R, St_r is the star rating supplied by the reviewer for review r, U is the distribution of user priorities, and x is an aspect within the list of aspects A. This updated star rating can then be ranked against similarly calculated ratings to provide the response to the user.

V. EXPERIMENTATION

In this section, we elaborate on the baseline algorithm we use for comparison, and discuss the results. To the best of our knowledge, no other project has attempted a similar star-rating-adjustment system to SRA. However, the problem of identifying review aspect weightings, as AWA does, is by no means new. Although many algorithms perform similar tasks, we decided to use the aspect-tagging component of the LARA algorithm as described in **[4]**. Both AWA and LARA were tested on the same dataset consisting of roughly 5400 reviews for nine hotels crawled from TripAdvisor over a one-month period. This test set represents approximately half of the dataset provided with the LARA source code. Both systems were also given the same set of aspect seedwords, as detailed in **Table 1.** We tested, the relative speed of RWAC against that of LARA, and we compared how closely RWAC results compared with LARA results.

Table 1. Test set seedwords for each aspect as used in both AWA and LARA

Value	Room	Location	Cleanliness	Service
value	room	location	clean	service
price	space	locate	dirty	manager
money	bed	near		internet
worth	bathroom	traffic		breakfast
cheap	bedroom	walk		desk
pay	suite	car		maid
dollar	comfortable	pool		check
charge	couch	lobby		concierge
expensive	noise			staff
budget	view			
	bed			
	shower			

Table 2. Execution times for AWA and LARA for trials 1, 2, and 3

	T_1 (s)	T_2 (s)	T_3 (s)	Average
AWA	45	43	48	45
LARA	157	158	169	161

1. COMPUTATION SPEED

For this portion of the testing, both systems were run on the same computer and with the same dataset so as to test their relative computational speeds. The test device was a Mid-2012 Retina MacBook Pro with a 2.6 GHz Intel Core i7 and 8GB of memory. Due to the IDEs in which both AWA and LARA were running (Pharm and Netbeans, respectively), both systems were limited to using one processor core, and no other applications were open at the time. The results are shown in **Table 2**.

It is important to note that AWA was run in Python, while LARA was run in Java, which is generally accepted to be as much as 25x faster than Python **[9]**. Thus, even while using a language generally accepted to be slower than LARA's, our aspect-analysis system was over 3.5 times faster than LARA's.

2. ACCURACY / SIMILARITY

For this portion of the testing, the resulting aspect distributions from AWA were compared to those from LARA's bootstrapping system. Using the same dataset as before, the distributions for each review were calculated using a Kullback-Leibler Divergence Cutoff. The cutoff chosen was at 0.5, as the distributions for reviews under 0.5 were similar enough for almost any usage. The similarity D is defined as

$$D = \frac{\left|\{L_r \mid D_{KL}\left(L_r \parallel W_r\right) < 0.5\}\right|}{\left|L\right|}$$

$$= \frac{\left|\left\{L_r \mid \left(\sum_{x \in A} \ln\left(\frac{L_{rx}}{W_{rx}}\right) L_{rx}\right) < 0.5\right\}\right|}{\left|L\right|} \quad (4)$$

where r is an arbitrary review and aspect x belongs to set of reviews A. The comparator calculated that D=0.7417, meaning that roughly 74% of AWA's distributions were reasonably close to LARA's. It is important to note that LARA itself is not 100% accurate, so the actual accuracy might vary slightly.

VI. CONCLUSION

In this paper, we proposed a fast and effective method called RWAC for discovering aspect weightings and adjusting star ratings to incorporate individual user priorities. Our empirical experimentation revealed that our system is similarly accurate to the baseline, and significantly faster. There certainly exist numerous opportunities for improvement for the algorithm. Most notably, the current incarnation of RWAC cannot handle sentences without major aspect relation. In spite of this, we argue that this algorithm may have potential in many different applications to combat the problems posed by an overabundance of reviews.

VII. ACKNOWLEDGEMENTS

This paper was written as part of the Research Mentorship Program 2014 at UCSB, under the mentorship of Honglei Liu and Dr. Xifeng Yan. Its completion would have been impossible without the assistance of the program staff including director Dr. Lina Kim, and TA Raymond Valdes. We also wish to thank the anonymous reviewers for their invaluable contributions.

VIII. REFERENCES

[1] Hu, Minqing and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD '04: Proc. Of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, New York, NY, USA. Pages 168-177

[2] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. Journal of Machine Learning Research 3:993–1022.

[3] Yohan Jo and Alice H. Oh. 2011. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining* (WSDM '11). ACM, New York, NY, USA, 815-824.

[4] Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (KDD '10). ACM, New York, NY, USA, 783-792.

[5] Tibor Kiss and Jan Strunk. 2006. Unsupervised Multilingual Sentence Boundary Detection. *Comput. Linguist.* 32, 4 (December 2006), 485-525.

[6] Onix text retrieval toolkit stopword list. http://www.lextek.com/manuals/onix/stopwords1.html.

[7] M. Porter. An algorithm for su±x stripping. Program, 14(3):130 { 137, 1980.

[8] Wikipedia contributors. (2014, June 4). Kullback-Leibler Divergence. [Online]. Available: http://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence

[9] Various contributors. (2014, June 4). Computer Language Benckmarks Game. [Online]. Available: http://benchmarksgame.alioth.debian.org/u64q/python