Better Hotel Reviews Report

CS410 - Text Information Systems

Team Members:
Jeremiah Rohwedder
Joshua Albers
Sagar Dhawan
Yuchen Qian

Table of Contents:

Abstract

We are exploring the possibility to improve hotel ratings by giving users the ability to remove unwanted reviews from consideration. In our system, reviews removed by the user will not affect the hotel's overall rating. In this way, the scores reflected in a hotel's overall rating will only include scores that user deems pertinent to him or her.

Our idea can be compared to adding "filters" in search. The main motivation being the fact that people tend know what they *don't want* more than what they *do want*. This leaves users with more personalized ratings than the conventional rating systems.

Its not hard to imagine a future where filtering methods like ours could be incorporated into broader user services like those provided by Google, Apple, or Facebook. Instead of explicitly requiring the user to enter filter queries, our filtering method could use the data on user interests and disinterests from these larger services.

Introduction

Anyone who has used online user reviews as a tool to aid in decision making when booking a hotel knows that user review systems are not perfect. While many websites have reviews of hotels, none currently have the ability to flag a review and remove it from the final rating. Our team's idea was to create a function that would not only remove the reviews, but also recalculate the final rating of the hotel without the removed hotels.

This is an advanced filtering method that gives users a revised set of ratings based on their filters. In regular filters, hotels that do not meet a certain criteria would just be dropped from the search results. This is not ideal, especially in the case of a many different reviews. Our goal was to then, demote hotels with negative filters instead of eliminating them completely.

Related Work

There are many websites that currently will allow user reviews of hotels. Examples include hotels.com, travelocity.com, even google.com. However, while these websites allow reviews, including spots for negative and positive comments, none of these websites allow users to flag a review as inappropriate and remove the review from consideration of the overall rating. Google will even allow you to flag a review as inappropriate, but it doesn't have an effect on the rating.

While discussing our project, we considered modifying a couple of different existing review feedback systems. The first system is websites that sell products and allow users to identify reviews that were helpful. The second system we considered is the upvote/downvote system used in many discussion boards. We considered recalculating hotel ratings by giving greater weight to reviews with many upvotes or that were helpful to many people.

In the end, however, we decided removing reviews altogether was more useful in the recalculation of ratings because each user values reviews differently. Also, removing the reviews that match our users' filters allows a fine grained filtering over the entire dataset. This means that the ratings won't necessarily be drastically different, because good hotels still rank high as the sheer number of good reviews would dominate but, it still affects the overall rating enough that the ratings are now "personalized" for the user.

Problem Definition

The problem with getting an average rating for a hotel is that it is based on the reviews that are given to it by the users. While it is good to have many users reviewing a service, it does become challenging to show a user relevant reviews. An example of this would be someone who complains because the hotel was crowded with people for a convention. If the user(s) who commented on convention put fewer stars(or more if they like convention space), a person just spending the night would not find the review helpful or useful. If there was such a way to remove the review not only from being seen, but also from consideration to the final overall review that would be most ideal. Thus, leaving a more personalized rating for the user to see.

Another problem specific to hotels, is their proximity to a user's desired location. By allowing the user to specify location or by obtaining it via a user database, as mentioned above, we can immediately discount certain hotels in our dataset.

Finally, a problem our idea solves automatically is quite simply, that of information overload. With the sheer number of reviews on popular services, most users wouldn't ever go past the top few reviews. By filtering out reviews that are deemed to be irrelevant to the user, we make sure that a user will only see reviews that are relevant.

Methods

The first problem that would be difficult to tackle would be the most obvious, the user input. Ideally, a text box would be created where the user is asked to input a phrase or sets of phrases. After the input(s) are entered any reviews with the phrase or phrases incorporated would be eliminated from consideration into the overall rating of the hotel.

Before we ask for input from the user, we create an index of the dataset we're interested in using Lucene libraries. Since we're only concerned with removing individual reviews, these are the documents we index. However, we also need to keep track of the hotels these reviews are attached to. We create a database of hotels in memory to keep track of the numbers necessary to dynamically calculate its ratings. If our idea is turned into a web application, one challenge would be to only remove reviews for a particular user since others may find those reviews he removed useful. To further expand on that. We could let corporations with user databases adapt our project idea to automatically filter reviews for their users. Such that a user will not be asked for their preferences but will be shown reviews that are pre-filtered based on the user database.

To recalculate the ratings, we first gather all the documents that match the user's query. For each match, we remove the rating and review from the hotel database and also remove the document from the index. Finally, we recalculate the ratings for each hotel affected by this query.

We also support multiple positive and negative filters that the user may input. For each new filter that the user adds, we re-iterate through the dataset with priority to the positive filter terms. For example, if the user enters "+food, +good ambience, -dirty, -expensive" as their filter, we go through the dataset and add weight to the positive terms while completely removing or penalizing reviews that have the negative terms. Ultimately leaving us with a set of overall ratings and reviews that match precisely what the user was looking for.

Finally, We display the top 10 hotels with the new ratings to the user based on their original query.

Evaluation/Results

The results show that our method works. We could clearly see the difference in the results before and after personalizing our dataset. By changing the ratings of the hotels we affect the top 10 hotels that would be returned. Since more relevant hotels and by extension, reviews are now higher up, we make it far more likely for a user to come across a better hotel than before.

We were able to successfully test the positive and negative filters and made sure that they affect the dataset only for one user at a time. This was important because making actual changes to the dataset is not desirable as that will cause us to lose information as more and more queries are given.

For certain location based queries, we saw a drastic change in the top 10 returned results. This again, was expected as proximity to user was one of the more important filtering factors.

One of the problems that we ran into was that we thought we could not get our crawler to work. For reasons described in the next section, we couldn’t gather the data consistently in the format we preferred. We additionally had a problem with our multiple input query. We were actually able to create a multiple query removing algorithm that would sort through a small created list of comments and remove comments that would contain the given phrase or phrases. However, as the way that the lucene libraries were created our working algorithm was not able to be implemented the way that we would like to do.

Conclusions/Future Work

From collecting the dataset to actually attempting to out do popular hotel rating schemes, our final project was a great learning experience. We learned that building a web crawler to gather specialized information such as hotel reviews can be tricky for a few reasons. Different review sites use different scales to rate hotels, users that give reviews may not complete every field for the review, and after a certain point in a list of all reviews, you have to click on a link to expand and read the full review. These complications during our research on web crawlers led us to the decision to use an existing dataset, so that we could focus on the more interesting part of our idea. We also learned a bit more about Lucene and some of it’s classes when implementing our index of hotel reviews.

For the Future, we could add compatibility for “user databases” to automatically offer personalized reviews and ratings to the user. Unfortunately most user databases, like those used by Google and Facebook, are internal to the companies and not publically accessible. Even so, it would be great if we could “predict” what a user likes and dislikes and automatically personalize the ratings to that users profile. This would be done similarly to the way that the Amazon algorithm works.

Another option for the future is something we discussed during our planning stages was to remove reviews based on grammar and spelling. We considered this because a user needs to be able to understand reviews for them to be relevant to him. This idea could be implemented as an option or an extension of what we have done as our filtering function is very modular.

Additionally, this system does not have to isolated to hotel reviews. A possible extension for the future is to make this idea reusable for any review system.

Lastly, we also would have liked to allow a multiple query removal as a user may want to remove multiple phrases from the criteria, but unfortunately with the way we had implemented the index, we weren’t yet able to remove multiple queries.

Individual Contributions

Jeremiah Rohwedder: I used lucene libraries to create an index from the dataset[1] we used in our project. I also handled removing reviews from the index that match the user's query.

Joshua Albers: Created a working algorithm for removing multiple queries, that did not ultimately mesh with the lucene libraries. Also attempted to create a web crawler, finding that we were not allowed to copy the data.

Sagar Dhawan: Worked with Yuchen on collecting the data. Experimented with a web crawler to collect Hotel Reviews. Unfortunately, we couldn't really do that cleanly, for most cases we ran into legal limitations on access to information. We did however find a dataset that matched our needs. Finally I worked on a Python script that would parse the entire dataset and convert it into a Java Database that Jeremiah and Joshua could later use to implement the front end.

Yuchen Qian: Worked with Sagar on collecting the data. Determining which retrieval method should we use and try to use the lucene. Also, we use the dataset and using python script to transform the dataset into our format of using. And the presentation with Sagar.

References

list references of work related to your project or websites with related systems here.

[1] Timan dataset

[2] Apache Lucene libraries. http://lucene.apache.org/core/

[3] Learning to recommend helpful hotel reviews.
http://dl.acm.org/citation.cfm?id=1639774

[4] Thumbs up or Thumbs down? : Semantic orientation applied to unsupervised classification of reviews. http://dl.acm.org/citation.cfm?id=1073153