Recommendation System Using Sentiment Analysis

Garrett Shankel[1], Duy Nguyen[2], David George[3]

Southern Methodist University, Dallas, Texas, USA

[1]gshankel@mail.smu.edu

[2]duynq@mail.smu.edu

[3]davidg@mail.smu.edu

***Abstract* - There is data to be mined within user reviews on travel websites. Currently travel websites offer little to no filtering when it comes to personal factors. Personal factors being things such as cleanliness, safety, friendliness, etc.**

By applying NLP to existing reviews, it is possible to generate a new rating system for each hotel based on these factors. Thus, providing a new way to rank hotels that further personalizes them on a user per user basis.

Users are not encouraged to create profiles on travel websites making it difficult to build a data description that can be personalized. Big data can be leveraged to improve the unidentified users experience by classifying the sentiment of the users that have left previous reviews. In this way with a simple understanding of the unknown user we can create a powerful data profile without requiring them to sign up for an account.

By storing the reviews and their NLP analysis in a database these features can be scaled up to meet any business needs. Scraping and processing take a long time and it would be impractical to expect a machine to process reviews each time a user viewed a set of websites. A well-kept database allows a travel website to store the valuable information alongside its reviews. Not only can the database be queried quickly to solve the existing problem, but further improvements can also be added. More data can be added, and more complex analysis can be done to generate extra value for businesses.

***Keywords*— recommender system; automated suggestion; sentiment analysis; web scraper; lexicon; Natural Language Processing; dictionary**

I. Introduction

Recommender systems are one of the most profitable data science applications in use today by large companies. Travel websites create a unique opportunity for implementing this system since it can be difficult to classify unique shoppers on their websites. Success seen by large companies like Amazon, Youtube, and Netflix when using recommender systems. Netflix is a classic example of this. In 2006 they ran a competition to see who could reduce the RMSE of their recommendation system by 10% and tagged on a gigantic prize of 1 million dollars for the winner. Currently sentiment analysis saves Netflix roughly 1 billion dollars a year and accounts for 3/4ths of their recommendations to users [*Reference 6*]. The value of big data is virtually limitless and companies early on saw the immense power it could bring.

With the method laid out in this paper it will be possible to leverage some of that value in the favor of travel websites. Processing user reviews and creating sentiment ratings will be the first necessary step. The main goal of this method is to use NLP to classify individual reviews for hotels. The classification categories will be based on features that are important to hotel guests; cleanliness, safety, noise, comfort, friendliness, etc.… A score will be given in each of these categories for each review. Then all the reviews will be tallied for a single hotel to give an overall score. This will be the basis of value. From here each hotel gains a new ranking system along with their original overall review score. Now allowing users to compare 2 hotels with the same overall review rating on a new scale.

Next a vehicle for filtering the results will need to be provided. One possible filtering tool would be a survey that an unknown user would be asked to fill out. The survey would simply ask them to rank how important key features are on a scale. From the survey the hotels could be displayed based on their overall rating and their key feature rating. A simpler implementation would be to have checkboxes in the available filter metrics that would allow users to select which features are most important.

II. Scraping

In an effort to simplify the presentation of the solution, TripAdvisor was chosen as one of the well-known websites and was the focus of this study. This website was simple to navigate, simple to scrape, and offered a large body of reviews to work with. Scraping was done with the python library “BeautifulSoup” on the first 10 pages of hotels, at the moment of scraping, that contained 30 hotels each, making the total number of hotels to be 300. From the 300 hotels in Paris, we only managed to capture ratings of 3.5 and above, and not lower. This can be a con because without a more powerful scraping technology, we cannot capture the full spectrum of sentiments that are based on the voluntary ratings metric alone. The username, the number of their contributions, the review’s voluntary rating, the title and body of the review, the date of stay, and the trip type were extracted from the html pages and recorded into a Mongo database. The sample review below shows the highlighted parts that were extracted.

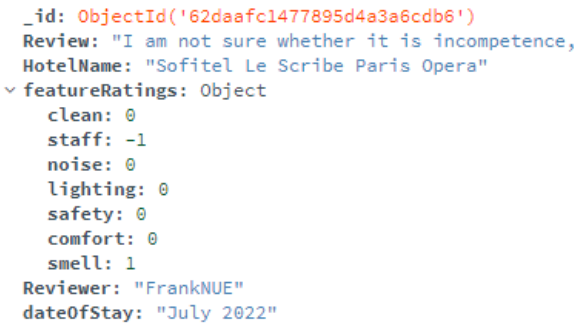
The method used to extract these dataset features was to use the libraries “nturl2path” and “requests” to simulate the actions of entering the desired url of Tripadvisor’s hotels in Paris, through a “for” loop that counts within the URL the strings “”, “oa30-”, “oa60-”, “oa90-” and so on, to scan the entire first page of 30 hotels and the following 9 pages after that, hence the total of 300 hotels. Within this “for” loop, a nested “for” loop was created to count within the URL the strings “”, “10”, “20”, “30”, and so on, to scan the entire first page of each hotel and the following 4 pages after that, which comes out to a total of roughly 14,000 reviews.

Using the processing power of computers at hand, 14,000 reviews were scraped in about 35 minutes, meaning 6.67 reviews per second. The first 10 pages of reviews, which can be set as the current scraping capacity, only captures reviews from the last 3 month, because TripAdvisor sorts their reviews by new and there are always 10 per page. So they can be processed and converted into emotions of negatives and positives using Natural Language Processing.

III. Populating The Database

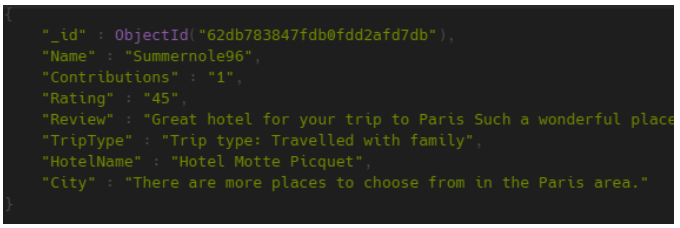
The program will take roughly 150 seconds to process 1000 reviews. 1000 reviews is an extremely small sample of the amount of reviews available online. This program would not be scalable without the addition of a database. Not only to store the reviews, which grow to an incredible size, but also to store the results of the NLP processing so they do not need to be processed again.

For the first version of the program, the mongoDB database is populated using the “pymongo” library from Python. A connection is established to the database. If the database does not exist then pymongo will create one. Once the connection is made, the collection in the database is populated with a dictionary in the following form:



The field “_id” is created automatically by MongoDB. Field “Review” is the review text body, which was extracted by searching for the “quote” HTML tag. Field “HotelName” was extracted by searching for the “heading” class HTML tag. Field “featureRatings” is created using the Python library “BeautifulSoup”, where multiple keywords can be used and stored one after the other in this dictionary format. This field will be further explained later. The rest of the fields should be explanatory and were extracted by specific class tags that were randomly generated.

For the second replication of the program, which was created solely for the purpose of data exploration, some of the methods mentioned previously are used again but excluding the “featureRatings” field. Its database form can be seen below.



To go further into the scraping tactic, the title was joined with the review text body to become the field “Review” because emotion can also be conveyed from review titles, for purposes of complete analysis. Fields “_id” and “Name” were extracted the same way from the first version of the program. Field “Contribution” was extracted from a randomly generated HTML tag to reserve as another purpose in a future project. Fields “Rating”, “TripType”, “HotelName”, and “City” were also extracted the same way and will be further discussed in the next section.

IV. Data Preparation

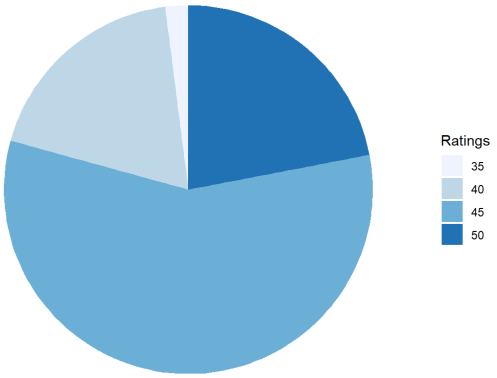
In the first version of the program, the scraped reviews are processed using Python to improve the accuracy of the NLP model. Processing involved turning capital letters to lowercase, removing punctuation, removing stop words, tokenization, and finally lemmatization. The goal of all this processing is to reduce the reviews into their base components and standardize the words in each review. This allows the algorithm to easily view patterns that occur in the data. The accuracy increases when the algorithm no longer must account for the difference between “Yes” and “yes” or “apples” and “apple”. A review that reads “Thanks to everyone for looking after us so well” after processing will read as “thank everyone look we well”.

In the second replication of the program, data preparation is done with the R language. After calling the Mongo database within RStudio with the corresponding fields (“Name”, “Contribution”, “Rating”, “Review”, “TripType”, “HotelName”, “City”), the number of reviews were confirmed again. With a total of 14,000 observations and 7 variables, our first instinct is to omit the “_id” column created by MongoDB simply because it is always redundant in analyses. Next we converted the variables that seemed to naturally be factor variables (“Rating”, “TripType”, “HotelName”, “City”).

V. Exploratory Data Analysis

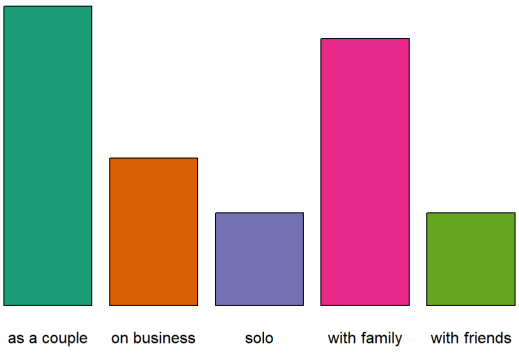
The "Contribution" column is reserved to benefit another rendition of this project in the futon. Information about this column can tell us about the veterancy of the reviewer, alongside with their username from the field "Name", and the individual weight of their review, compared to the many others. With this information in hand, we can group similar types of users and possibly get a grasp of the frequency of their hotel stays, their hot topics when reviewing hotels and likelihood of them usually reviewing either only negative reviews or positive reviews. Knowledge of these unique and obscure facts can be extremely powerful as a hotel owner or corporate, as it can be leveraged to track those certain types of users and target their preferences accordingly and increase profits.

The "Rating" column only has 4 levels ("35", "40", "45", "50"). Half of the reviews rated a 4.5, a quarter of them rated 5.0 and 90% of the other quarter rated 4.0, leaving the rest to be 3.5. No extracted reviews are rated 3.0 and lower. A better visual explanation can be seen below.



For those reviews that are rated 3.5, we can say that there is still a good amount of people that aren't satisfied with the hotels in Paris and those people should be targeted with special offers or the like. However, from the voluntary ratings alone, we can be sure about the high quality of hotels in Paris.

The "TripType" column needed to be trimmed down in order to extract what we really need, the 5 types of travellers inputted voluntarily by Tripadvisor users. The beginning string "Trip type: Travelled " was removed from all rows which leaves us with the traveller types ("as a couple", "on business", "solo", "with family", "with friends"). We were very glad that we can get a good idea of the multiple varieties of people represented by Tripadvisor reviews from this dataset feature alone..



The amount of couples and families (who gave a review on their stay) dominated the other traveller types as expected. It can be said that Paris is the pinnacle of romantic cities, and lately the city has been trying to cater more to families and call itself more kid-friendly. About half of those two numbers are made for business trips, whereas the number of people who travelled solo or with friends is slightly lower than that but they seem to be equal.

The column "HotelName" was extracted using a randomly generated HTML tag and was reserved to use for a future rendition of this project. The idea was to analyze sentiment from words that belong in a hotel, but we do not have the right tools and computing power to get a big enough dataset at the moment.

The column "City" was trimmed down by the beginning string "There are more places to choose from in the " and the trailing string " area." This was also reserved for a future rendition of this project, for purposes of analyzing trends of sentiment from hotels in a different city, as well as hoping for a new perspective of traveller types in a different city. Then an overall analysis of hotels on a specific continent would be created to inspect its trends based on the living conditions and the like of such a continent. Many possibilities and opportunities to increase the value of this study if only we had more time and resources.

Recalling the analysis of reviews, word clouds were created using a corpus system, similar to a dictionary, where sentiment analysis redundancies were removed like punctuation, uppercase characters converted to lower, numbers, and spaces. Additionally, the following unique words were also removed, simply because they do not add values of emotion: "paris", "hotel", "one", "two", "airport", "terminal", "really", "just", "flight", "very", "quite", "rather", "didnt", "youre", "westin", "marriot", "stay", "stayed", "will", "also", "day", "time", "rooms". The last thing to remove is English stopwords, which include pronouns, demonstrative adjectives "this" "that" "these" "those", and etc. A simple Google search can give you a demonstration of these words [*Reference 7*]. An overall word cloud created to demonstrate the frequency of words can be seen below.

However with much more reviews in our database, we will see a more personalized spectrum of words that appear the most for each of the 5 traveller types mentioned earlier. In the meantime, a compilation of word clouds created for each dedicated travel type can be seen below.

As observed above, “room”, “staff”, and “location” are the most mentioned words from the 14,000 reviews. And likewise, those 3 words appeared the most amongst the 5 types of travellers: couples, business trips, solos, families, and friends. However, once a more powerful scraping technology is used, we will hopefully start to see more specific words like “clean”, “safe”, “comfort”, “smell”, “lighting”, etc… In general however, something like the dashboard above can be very profitable for corporate as information about one or some types of travellers who most frequent your hotel and topics that they care most about is very valuable for new hotels as well as long-standing ones.

VI. Natural Language Processing

The current method for NLP is to use the library nltk.sentiment.vader. The processing used is a lexical approach meaning that a value is assigned to each word. This will range from -4 to +4. The total of all of the words in a phrase is taken and averaged to give the resulting sentiment value.

In order to get sentiment values for each individual word the program iterates over the review looking for the appearance of a keyword. When that keyword is identified the program then runs sentiment analysis on the sentence. The prevailing sentiment (negative, positive, or neutral) is taken and a counter is increased for the keyword.

A. VADER

nltk.sentiment.vader was the first toolkit used to perform sentiment analysis. VADER stands for Valence Aware Dictionary for Sentiment Reasoning. Vader was originally chosen for its ease of use and availability.

Vader can classify text on its positive or negative connotation and on the intensity of that emotion. It does this by maintaining a dictionary of common words that have all been pre-classified. Words are given a score from -4 to +4. Lower values being strongly negative and higher values being strongly positive. On top of the predetermined ranking VADER will also recognize words that increase intensity within a sentence. So words such as “very” or “greatly” will have a multiplicative effect on words or phrases. VADER also contains a dictionary of simple phrases. So a phrase like “the bee's knees” can also be classified as positive.

Input	neg	neu	pos	compound
"This computer is a good deal."	0	0.58	0.42	0.44
"This computer is a very good deal."	0	0.61	0.39	0.49
"This computer is a very good deal!!"	0	0.57	0.43	0.58
This computer is a very good deal!! :-)"	0	0.44	0.56	0.74
This computer is a VERY good deal!! :-)"	0	0.393	0.61	0.82

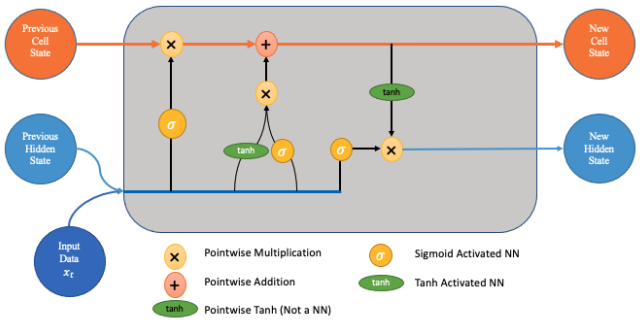
VADER served as an excellent starting point in understanding the data. As will be mentioned later in this section, VADER was even used to automate the task of classifying reviews for another algorithm. However, some issues arose with the use of VADER. VADER is simple to implement and simple to understand making it incredibly useful when beginning NLP, but this simplicity created a hurdle. VADER had trouble classifying reviews that were poorly written, confusing, or that contained passive aggressive statements. For example, the phrase “the hotel was anything but clean” could be misclassified by VADER as a positive sentence in favor of cleanliness simply because VADER could not glean context. This motivated the move to an algorithm that could better understand context, LSTM.

B. LSTM

LSTM stands for Long Short-Term Memory and is a Recurrent Neural Network that specializes in classifying sentences in context. Due to its ability to remember past data and apply it to new context, LSTM served as a reasonable candidate to solve the issues with sentiment that VADER had. Recurrent Neural Networks can remember previous information, like how humans learn. Applying past experiences to new problems. This makes them excellent classifiers for context as they can remember previous phrases or quirks and apply them to new phrases.

Classic RNN’s do have a memory limit, they will begin dropping things they learned if it has not been deemed useful. LSTM was specifically designed to retain information for long +periods of time to create a long term memory for further understanding of data.

LSTM's work in three parts. The first part decides whether data will be relevant, the second part will add new information or update existing information based on input and the last part will pass along the information learned to the next hop.



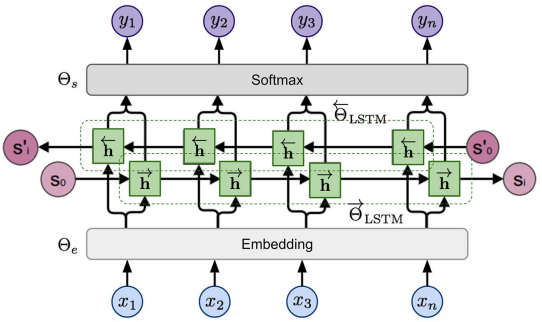
LSTM appeared to be the best fit for this problem, but LSTM requires a large training data set. As with any RNN, a very large data set is required to train the model accurately. The problem though is that the approach employed in this paper is novel. Meaning there are no large pre-classified data sets readily available on the internet. It is not possible to go out and grab a data set that classifies hotel reviews on their friendliness or safety. This meant that a whole new data set needed to be classified to train this model. A **minimum** of 3000 reviews were necessary to gain a decent level of accuracy.

To automate this process VADER was used to make the first pass at classifying reviews. If the review was positive and mentioned "clean" or a synonym of clean it was tagged by VADER as clean. If a word was negative and mentioned "clean" or an antonym of clean it was tagged as dirty. From there manual classification was done to classify 500 reviews. It quickly became apparent that classifying the minimum number of reviews would take an excessive amount of time and classifying enough reviews to get a strong accuracy would not be worth the effort. So again a new method needed to be employed in order to classify the reviews in context.

C. *ELMo*

ELMo or Embeddings from Language Models is an NLP framework that was built and trained on large textual datasets so that the training could be transferred to other problems with smaller datasets.

ELMo is an improvement on the LSTM algorithm and an example of transfer learning. ELMo employs a bidirectional learning model to LSTM meaning it will feed sentences forwards and backwards through its recurrent neural networks. ELMo was designed and trained specifically for text processing and classifying context. It has been shown to outperform previous transfer learning algorithms as well. Thanks to ELMo the 500 classified reviews were able to be used as the training and test sets for the new model and the rest of the reviews could be classified. The original LSTM models would perform at an accuracy of 26%, but after applying ELMo to the same datasets it was possible to increase that accuracy to 72%.



VII. Generating Output Based on User Selection

Once the sentiment count for keywords has been completed the database can be referenced by keyword. If a customer is looking for a hotel that rates highly for cleanliness then the hotels with the highest scores for the "clean" keyword can be pulled. The hotels can then be displayed by overall rating and cleanliness rating.

The simplest implementation would be to provide the user with check boxes or a rating system in line with the filtering options at the top of the page. This gives the user even more control over what they would like to see, and they can change their mind at any time. Along with this, if the user already has a profile created and their preferences are known ahead of time the list of hotels can be pre filtered without the user having to do any work on their part. Creating a more seamless user experience.

VIII. Conclusion

By employing NLP combined with a mongoDB database we have created a new way for websites to filter searches based on criteria that are not traditionally covered. With the ability to narrow down hotels based on user preference, travel websites can offer insight that goes above and beyond a simple average of the review scores. Finding a hotel that scores 4+ stars on a website in a user's price range is not difficult. However, finding a hotel that matches that user's specific preferences offers a whole new area to apply data processing.

To view the code used for this project please go to: https://github.com/shankel3443/DS-7330-Term-Project

References

[1] https://towardsdatascience.com/7-nlp-techniques-you-can-easily-implement-with-python-dc0ade1a53c2

[2] https://www.guru99.com/wordnet-nltk.html
[3] https://stackoverflow.com/questions/714063/importing-modules-from-parent-folder
[4] https://www.tripadvisor.com
[5] https://www.worthwebscraping.com/scrape-tripadvisor-reviews-using-python/
[6] https://www.plytix.com/blog/revenue-optimization
[7] https://gist.github.com/sebleier/554280