Alison Kahn
Gina McFarland
Paul O'Leary
COMP 4447

Can NLP on Airbnb Reviews Predict Ratings?

https://github.com/Poleary6/TOOL1_FINAL_PROJECT

Image Credit: https://airbnb.com

Introduction

Airbnb has grown from a mere 6 million users in 2012 to over 150 million users as of 2018 (Curry). In that time, the number of listings available on the site has grown from 300,000 to 7 million. As with any rental accommodation, the quality of the listings can range from awful to exemplary. As a company, Airbnb has worked to weed out the worst of the listings, but customers often have to rely on reviews to ensure that they rent accommodations that meet their needs. To facilitate this, Airbnb not only allows for open text reviews of a listing once the customer has completed their stay, but they also provide aggregate rankings of the listing on a 1-10 scale based on six different categories: Accuracy, Location, Communication, Check In, Cleanliness, and Value.

Motivation

With the explosion of Airbnb usage, there are always extreme experiences at either end of the positive and negative spectrum. But as high categorical rankings flourish and "Airbnb host" becomes a profession, Airbnb users have started wondering if the host-customer interaction isn't leading to artificially inflated categorical numbers. In the article "Why Airbnb Reviews Don't Tell the Whole Story" (Mann), author Sonya Mann posits that hosts can impact how their

customers rank them, and this can lead to artificially high ratings. On the flipside, Airbnb text reviews are an unfiltered stream of thought from the reviewer to the world. They have slang, grammatical errors, misspellings (sometimes intentional), and multiple languages. The customer is able to identify negative experiences through a myriad of word choices. The human reviewer easily breaks this group of thoughts down. However, reflecting this information and its sentiment is a tremendous challenge for a computer. Natural Language Processing (NLP) is a tool that has been developed for such applications and is our tool for exploring the relationship between the text reviews of an Airbnb listing and its aggregate categorical ratings.

Research Question

Given both open text reviews and ordinal numeric ratings on an Airbnb listing, can we use NLP to predict ratings in each of the six categories (Accuracy, Location, Communication, Check In, Cleanliness, and Value), or do the results indicate a bias towards a higher or lower categorical rating? Of the three approaches outlined within this project, which yields the most accurate results and why?

Photo credit: http://previewchicago.com/2018/10/03/5-best-skyline-views-in-chicago/

Dataset

The data is an extensive collection of Airbnb reviews from stays in Chicago collected from data.world. (https://data.world/bjblock/airbnb-chicago). This dataset was selected partly because the sheer volume of data would give us a wide field to work with, and also due to the fact that the original state of the text reviews would necessitate a good deal of cleaning and engineering to provide useful analysis. Additionally, because all of the data was from the Chicago area,

there would be a lot of common features within the text data that could be used to identify common topics of interest. The original data from data.world contained four .csv files:

- **hosts.csv**: a set of Airbnb hosts in the Chicago area, with identifying data
- **listings.csv**: Airbnb listings owned in the Chicago area with location information, descriptive attributes of the accommodation, and average review information
- **availabilities.csv**: listing IDs of properties with dates, availabilities, price and minimum and maximum number of nights to stay
- **reviews.csv**: reviews for each property, identified by ID, along with reviewer ID, reviewer name, and long form text review

Because this project aimed to show how the long form text reviews could be analyzed to identify the sentiment of the reviewer and ultimately predict the rating of the listing, we focused on two files: **listings.csv** and **reviews.csv**.

From listings.csv, we utilized the following fields:

- *id*: The Airbnb identification number of the listing
- *review_scores_accuracy*: The average score of the accuracy of the listing
- *review_scores_cleanliness*: The average score of the cleanliness of the listing
- *review_scores_checkin*: The average score of the check in process
- *review_scores_communication*: The average score of the communication between the host and renter
- *review_scores_location*: The average score of the location of the listing
- *review_scores_value*: The average score of the value of the listing

All review scores were given on a scale from 1 to 10 with 1 being the lowest score, and 10 being the highest.

From reviews.csv, we used the fields:

- *listing_id*: The Airbnb identification number of the listing
- *comments*: Plain text comments on the listing based on the user's experience

The reviews.csv file contained 350,673 individual reviews which formed the basis for our project.

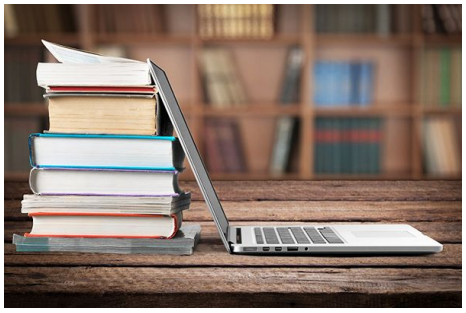


Photo credit: https://www.library.unsw.edu.au/study/information-resources/literature-reviews

Literature Review

Because this topic dealt with the relatively popular subject areas of NLP and Sentiment Analysis, there was much research available on the current states of the topics, and future work that could help inform our project. Below are a few reports that helped inform our understanding of the existing topic areas, introduce us to advanced topics being performed in the field of review ratings predictions, and determine how our project would help answer existing questions when it came to the topic of Airbnb data.

Natural Language Processing

"Natural language processing: an introduction" (Nadkarni et al.)

This article discusses the birth of natural language processing (NLP) and its evolution to its state at the time of publication, 2011. NLP began in the 1950s as a way to combine the fields of artificial intelligence and linguistics. The field evolved from word-to-word translation to grammar analysis in the late 1950s. From this grew methods of validating programming code syntax and the grep regular expression utility on UNIX. In the 1970s, lexical analyzer programs transformed text into tokens based on these grammar rules. Another advancement in the 1970s was the creation of the Prolog language, which is more suitable for NLP and differs from other programming languages in the manner in which it is parsed. Due to the ambiguous nature of natural language, these problems sets became unmanageably large. This led to process changes in the 1980s in which NLP analysis became more formulaic and scientific. Simple approximations replaced the deeper analyses of the past, evaluations became more mathematically rigorous, machine learning using probabilities became more prominent, and corpora were developed to train text. The article discusses the low and higher level problems addressed by NLP as well as the issues that make this type of processing particularly

challenging. Statistical and machine learning approaches were introduced including Support Vector Machines (SVMs), Hidden Markov models (HMMs), Conditional Random Fields (CRFs), and N-grams. The article ends with stating the NLP is still geared towards the advanced programmer, but is overdue for commoditization. This article was published in 2011, and in the last 10 years, that has been answered with multiple libraries, including Vader and TextBlob.

Advanced Review Ratings Prediction

"Combining Review Text Content and Reviewer-Item Rating Matrix to Predict Review Rating" ***(Wang et al.)***

The Airbnb data demonstrates a difference among sentiment analysis in reviews that will become more prevalent as NLP is used to extract meaningful content from consumer reviews. Much of the published NLP exercises in user review sentiment analysis focuses on determining whether the review in and of itself is positive or negative. Additionally, techniques such as Latent Dirichlet Allocation (LDA) are often used to identify prevalent topic themes based on the content of the reviews themselves. Our project does the reverse of this by attempting to identify and then quantify the sentiment based on known categories of importance.

This journal article proposes combining textual content and collaborative filtering to better predict review ratings. There are tremendous financial incentives for businesses to ensure they are getting good ratings. As the article points out, consumers are willing to pay 20 - 99% more for products with better reviews. Currently review rating prediction (RRP) has two methods: 1) NLP of the review test and 2) collaborative filtering. Collaborative filtering focuses on the matrices based on previously reviewed items of recommenders and not on the review text. This article suggested combining these two information sets. In order to do so, the authors proposed three new methods: RRP by combining Linear Regression and k-Nearest Neighbor, RRP by Combining Linear Regression and Matrix Factorization, and RRP by Combining Linear Regression Model, k-Nearest Neighbor, and Matrix Factorization. These methods significantly enhanced the review prediction performance. While these methods went further than we did in our study, it shows that the combining of external information using model projection with existing NLP techniques can enhance the accuracy of a model, and demonstrated a trajectory to take with our research.

Extending Project Beyond Current Research

“After a Disappointing Airbnb Stay, I Realized There's a Major Flaw in the Review System” ***(Mulshine)***

This article proposes a theory as to why the reviews on Airbnb are consistently effusive in their praise. Since the reviews are not anonymous, this incentivizes the reviewer to be positive so as not to be seen as someone who is negative or makes trouble for future hosts. Our analysis will move beyond this premise to show how closely the numbers and ratings match, with the idea that though they are consistently high, perhaps there is more fundamental truth behind the text than the ratings numbers given.

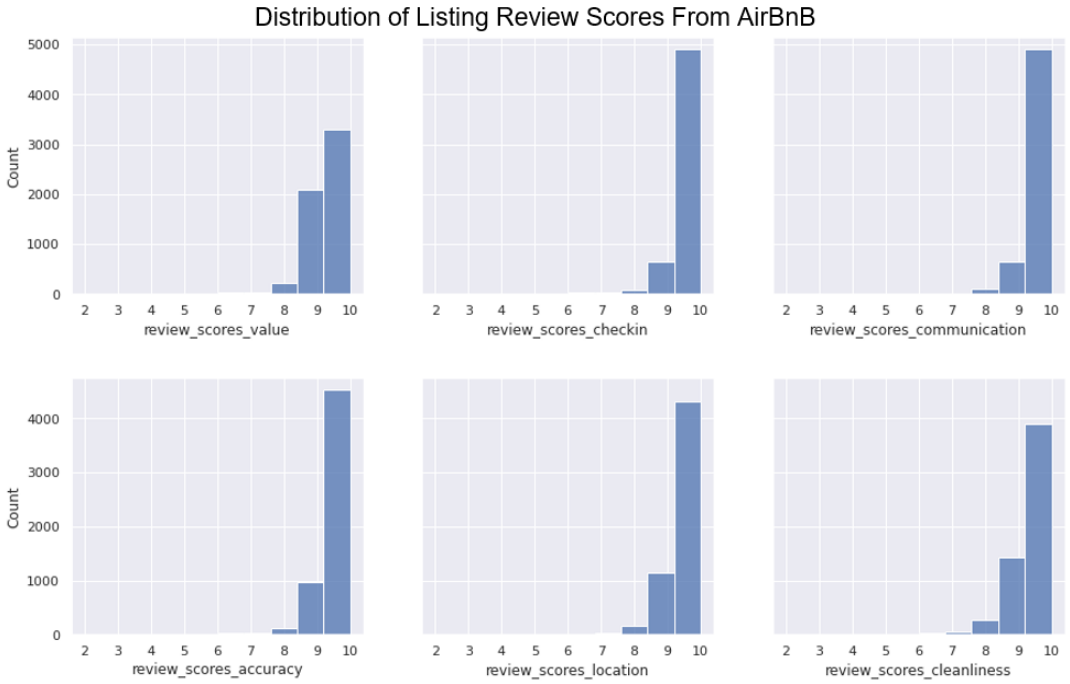
Image credit: https://www.inzata.com/what-is-data-cleaning/

Data Cleaning and Analysis

Before beginning the project itself, the team was forced to evaluate how we wanted to execute the project. Because of the nature of our proposed research, it seemed difficult for each person to work on individual tasks in parallel with the others. Additionally, all members had different ideas relating to how the project could be implemented. Given these two constraints, we decided to each implement our own interpretation of the review rating predictions on a subset of the data. We would then compare our results on the others’ data, and draw conclusions regarding the efficacy of each model. Review data was divided evenly among the three team members. The first sample contained record numbers 1 through 116839. The second sample constrained record numbers 116840 through 233904. The third and final sample contained records 233905 through 350675.

Prior to working on the individual samples of data, the team looked at some overall distributions; namely, did the existing Airbnb users’ concerns that the listings’ review scores were too high seem justified? Judging from the charts below, the answer appears to be *yes*. These histograms

represent the reviews given for the entire data set in all six categories. Data is extremely skewed towards positive ratings in every category.



The next step for each member of the team was to use the text reviews to determine if the ratings were reasonable, or if there seems to be a bias towards rating higher when using a numeric scale versus a text review.

In the sections that follow, each team member will outline the steps that they took to implement their review rating prediction algorithm.

Naive Sentiment Processing - Procedure and Analysis

Paul O'Leary

To begin, basic exploratory data analysis was conducted on the files. This batch of 117,064 records included 64 with NaN in the comments field. Those records were dropped, leaving 116990 records for processing. EDA also revealed 4680 records that were not in English.

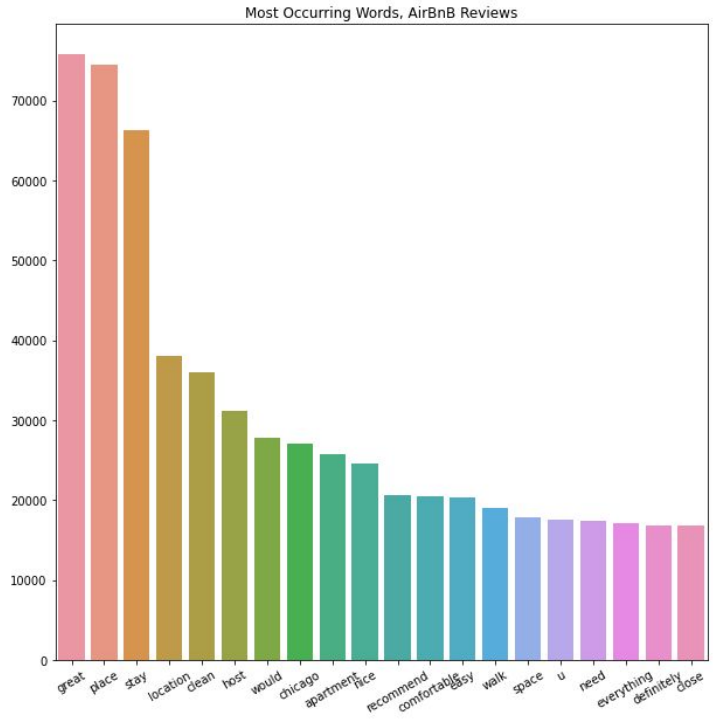
Translating these records became the first major clean up task. Python library 'langdetect' includes a function called 'detect'. Using this, a list of lists was created that included the record number and the language detected. Numerous unsuccessful attempts were made to use

Google Translator and TextBlob translation services. Finally, an implementation of Google Translator available from the Python library **pygoogletranslation** was used successfully. However, this service stopped translating after about 1200 records, and the service was locked for 24 hours. Over four days the 4680 records were successfully translated. The fully English reviews file was saved to disk, and this version was reloaded for further processing.

To explore the reviews, and NLP text processing, the Natural Language Toolkit (NLTK) library was used. For initial processing the data was broken down into individual words: The raw comments were sentence tokenized (nltk.sent_tokenize(cmt)), resulting in a list of lists, with each inner list containing multiple sentences. Sentences will be used later for sentiment analysis. For this initial exploration, the tokenized sentences were further tokenized into words (nltk.word_tokenize(snt)) and those words were made lowercase. At this point, the list contained 7,034,069 tokenized words! This list contained punctuation marks and stop words. Regex filtering and **nltk.corpus.stopwords('english')** were used to remove these, dropping the count to 3,314,046 words in the list. Contractions were then corrected. After this processing, the list still contained the "'" (apostrophe) symbol. These were removed, dropping the word count to 3,298,215.

The nltk WordNetLemmatizer was then used. Lemmatizing is the process of reducing words to a base root word. For example, "apartments" would be reduced to "apartment" and "stayed" will be reduced to "stay". This is an important step in NLP, ensuring that various forms of the same root word are considered to have the same meaning. For this initial analysis, POS (part of speech) processing was not used. Lemmatization reduced the count to 2,635,284 words.

Nltk.FreqDist was used to produce a list of the words that appear the most frequently in this reduced list, and that data used in a Seaborn barplot shows the frequency distribution of the top 20 words in the reviews from the dataset:



Wordcloud provides another way to visualize frequency distribution of word data:

In order to attempt to predict numeric ratings for six categories of the listing, individual words are not sufficient. Initially, the words above in the worddcloud were tokenized into individual sentences. The primary goal of this research is to predict the numeric ratings given by renters in six categories - Accuracy of the listing, Cleanliness, the Check In process, Communication with the host, the Location, and the Value of the unit - by analyzing text reviews entered by the renter. A number of Python libraries exist to assist with this form of “sentiment analysis”. This researcher, however, attempted to do it “from scratch”.

The sentiment analysis of the Airbnb reviews was attempted using a simplistic, possibly naive, approach. As described earlier, the reviews were tokenized into sentences. The sentences were treated as a single sentiment. To determine the category that a sentence might be referring to, sets of trigger words were created for each of the six. For example, the following set for cleanliness was created::

```
cleanliness_set = {‘floor', 'toilet', 'shower', 'wall', 'smell', 'bathroom', 'clean',
                   'fresh', 'spotless', 'tidy', 'pure', 'orderly', 'order', 'door', 'window',
                   'stair', 'light', 'odor', 'mess', 'messy'}
```

Similar sets were created for the other five categories, with different trigger words. Additionally, positive and negative sentiment word sets were created. For example:

```
positive_rev_words = {'great', 'excellent', 'good', 'nice', 'pretty', 'clean', 'easy', 'positive', 'cozy',
                      'awesome', 'big', 'love', 'quiet', 'perfect', 'sweet', 'huge', 'vast', 'exceptional',
                      'marvelous', 'satisfactory', 'superb', 'wonderful', 'please', 'pleasing', 'prime',
                      'first-class', 'superior', 'ok', 'beautiful', 'affordable', 'value', 'accurate', 'fair',
                      'true', 'quiet', 'large', 'modern', 'pure', 'definitely'}
```

These sets include words added after multiple iterations of testing, examination of the reviews, and even the use of a thesaurus, to determine possible trigger words for each category and each sentiment.

The first attempt employed a strictly binary approach to the analysis. Each sentence was parsed to find a category trigger word and if found, the sentence was compared to the sentiment sets. The only possible results were 10 for a positive result and 0 for a negative. Counts were tracked and average scores based on this binary approach were computed and compared to the actual reviews received. This initial attempt resulted in mean differences between predicted and actual scores of 5 and higher, very poor results.

To improve the model, a few strategies were used. First, reviews frequently included only a single word, such as “Great!”, or short phrases, such as “It was awesome!” These comments had initially been ignored by the triggers. The code was adjusted to check strictly the positive or negative sentiment for short phrases, and if positive, all categories were updated with a positive score. Additionally, a ‘neutral’ category was added, in the case where a sentence did not include a positive or negative word. Initially, this neutral sentence was rated as a 5.

These changes did improve the mean difference between the predicted and actual values, but not significantly. The skewed nature of renter reviews remained an issue. As shown above, regardless of what is written in a text review, the great majority of ratings for all categories are 9s and 10s. In fact, listings.describe() shows that the actual review ratings are all greater than 9.5! Therefore, the values for a negative or neutral review for a triggered category were raised higher. With these changes, the Total Mean Difference between the predicted and actual results from the three batches of reviews are shown below.

Results: Naive Sentiment Analysis

Sample 2

Category	Mean_Difference
accuracy_diff	3.282758
cleanliness_diff	1.851901
checkin_diff	2.126833
communication_diff	2.375808
location_diff	1.941085
value_diff	2.087862

Sample 1

Category	Mean_Difference
accuracy_diff	3.412333
cleanliness_diff	1.921278
checkin_diff	2.196188
communication_diff	2.456525
location_diff	2.009803
value_diff	2.169984

Sample 3

Category	Mean_Difference
accuracy_diff	3.516571
cleanliness_diff	1.976589
checkin_diff	2.250804
communication_diff	2.520619
location_diff	2.064760
value_diff	2.234720

Clearly, using this simplistic approach, the accuracy score was the most difficult to predict. Sentences addressing accuracy were difficult to catch with a single trigger word, and often did not contain a clear sentiment.

Limitations with this naive approach to determining rating scores from text reviews would prevent the predicted scores in all categories from being consistently close to the actual scores. This approach was improved iteratively, with more trigger and sentiment words being added. Significantly skewing the scores predicted for neutral and negative comments showed some promise on this particular data set, but would likely not work on more normally distributed scores. Next steps to improve this naive approach would be to attempt to analyze phrases, as opposed to simply individual words, and to greatly improve and expand the trigger and sentiment word/phrase sets.

TextBlob Text Processing - Procedure and Analysis

Gina McFarland

I chose to use the Python text processing library TextBlob for this analysis. I initially analyzed the first portion of the file, 116,772 rows, and later tested my method on the other two file sections. I read in the data from the reviews.csv file into a Jupyter Notebook. I found there were 60 values with nulls in the comments. Alternative methods of dealing with null values, such as filling with data from other rows, did not make sense in this context. The individual data from each reviewer was valuable and any sort of substitution method would alter the results. Therefore, these 60 rows were dropped. 116,655 rows of reviews remained in the data set.

The Airbnb reviews had six different categories: Accuracy, Check In, Communication, Location, Value, and Cleanliness. I created six different word lists for each of those categories. I initially put in words I thought would pick up the appropriate review lines. I refined each of the categories by trial and error. At this point, the only preprocessing performed on the data was to convert the text to lowercase. In this sorting system, the idea was that the review contained one thought per sentence and that thought would be captured in its appropriate category. Naturally, that was not often true. Once a key word for a category was found, the entire sentence was placed in the dictionary. To mitigate the issue of multiple ideas in one sentence, I ranked my search by topics that appeared less often (check in and accuracy) to those that appeared the most (location). This way I was able to get the more rare check in data while not losing much precision if that sentence was combined with yet another location review, for example. The sentences that did not have predefined category trigger words went into the Unspecified dictionary. My original intention was to use the Unspecified data in final processing to raise or lower the score of each category since those sentences tended to give an overall feel for the visit. For example, “Everything was great!” is an example of a sentence that would fall in this category. However, the resulting calculated scores for each category tended to reflect the given score fairly accurately in most cases. There was a category that was fairly far off, Accuracy, but I intended to apply this correction from the Unspecified group in a uniform manner and that would have skewed results for other categories. All foreign language text was added to the Unspecified category to be translated and sorted later because this turned out to be a very time consuming and difficult process.

Following the sorting process, the intensive cleaning and text processing began for each group. First, the Unspecified group was analyzed to translate the foreign language text and re-sort those sentences to their proper groups. This turned out to be a tremendously time consuming task! After trying Google Translate but having a myriad of problems, I settled on using TextBlob. While their translation service worked well, I learned the hard way that they require a 5 second rate limit between calls. After being blocked for 24 hours, I tried again. However, the service would still often disconnect and force the process to begin again. It took multiple days to complete the translations.

After all sentences were in their proper categories post translation, I performed further text cleaning and processing. TextBlob's spelling correction method revised spelling errors, but it was quite time consuming. It added up to 8 hours to runs, so I often commented it out without much degradation in accuracy. In the Mean Absolute Difference scores comparing my calculated scores with the given review scores, the numbers with and without spelling correction differed only in the thousandths digit. After spelling correction, punctuation was removed, and the words were tokenized and lemmatized. After lemmatization, stop words were removed. Finally, all of the normalized words for each listing by category were combined into a single list and TextBlob's sentiment utility was called to analyze the text.

TextBlob's sentiment score has a polarity and subjectivity component. First, I shifted the scale of polarity from -1 to 1 to a 1 to 10 scale. Next, if the subjectivity score was greater than or equal to 0, I increased the score by 0.5. If the score was less than 0, I decreased the score by 0.5. The rationale was that a stronger subjectivity rating amplified the score, either in a positive or negative way. I initially increased scores by 1.5 based on sentiment and only decreased scores by only 0.5 in an effort to more closely match the given scores in the listings file. However, I decided that was artificially skewing the results from the sentiment analysis. Based on our literature reviews, it may be argued the lower calculated numbers are more accurate and that the listing reviews numbers are artificially inflated. In the end, I kept the consistent 0.5 increase or 0.5 decrease based on the results of the subjectivity score.

Feature engineering techniques included missing data removal, and data normalization by way of translation, spelling correction, and lemmatization, and stop word removal. Feature extraction from the data was the calculated scores for each of the six categories.

TextBlob is a useful library and gave results consistent with the other NLP library, Vader. Its utilities for language translation and spelling correction, while useful, are quite slow which makes them cumbersome to use. The sentiment score seemed fairly accurate. Further investigation of natural language processing theory may be helpful in discerning more a precise way of combining the polarity and subjectivity scores into a meaningful overall score.

Results: TextBlob Analysis

Sample 1

	Mean Abs Diff
accuracy_diff	3.042062
clean_diff	1.909921
checkin_diff	2.894851
comm_diff	2.221797
location_diff	1.742865
value_diff	2.417353

Sample 2

	Mean Abs Diff
accuracy_diff	1.585943
clean_diff	0.839224
checkin_diff	1.642709
comm_diff	0.951660
location_diff	0.792408
value_diff	1.261304

Sample 3

	Mean Abs Diff
accuracy_diff	3.042062
clean_diff	1.909921
checkin_diff	2.894851
comm_diff	2.221797
location_diff	1.742865
value_diff	2.417353

VADER Sentiment Analysis- Procedure and Analysis
Alison Kahn

With the Airbnb review data, we were dealing with relatively short snippets of text. The project aimed to evaluate sentiment on six predefined topics: Accuracy, Location, Communication, Check In, Cleanliness, and Value. The following were some of the issues that I noted with the text itself.

1. The text was often short and captured thoughts on different rating areas within one review
2. Sentiment could range between positive and negative within the span of one review
3. Emphasis was often added via punctuation or capitalization within the review
4. Some reviews were not written in English

As this was a text based project, there was quite a bit of data cleaning and feature engineering that had to be performed before the data could be analyzed.

Before any text modification was performed, it made sense to ensure that the dataset had a consistent language baseline. To do that, translation had to be performed. Although there are many Python language translation modules available, almost all of them use the Google Cloud API. I ended up using **googletrans** as it seemed less prone to errors in execution. To initially evaluate the interpreted language, I used the **langdetect** library, as it did not throw any HTTP exceptions. Using langdetect, I was able to iterate through all rows of the dataframe, and make a list of the columns that contained a non-English language. The next step was to iterate through those indices and make the translations using the googletrans library. Because of the API limits ("Quotas and Limits | Cloud Translation"), the translation would often get cutoff for 24 hours before processing could complete. I was able to successfully translate subsets of text that did not hit the Google Cloud limits. Ultimately, because of the size of the initial dataset, I felt that it was acceptable to remove the untranslated non-English text and proceed with the remaining data.

The first two issues listed earlier regarding the amount of content within a fairly brief review meant that even though the text provided for each review was short, I could not treat it as a single entity. Therefore, I first modified each comment field within my data frame to a list of individual sentences. Each sentence would be analyzed individually to determine the main subject matter and sentiment.

The other issue that I hoped to capture was the range of positive and negative sentiment for each. I chose the **VADER (Valence Aware Dictionary and sEntiment Reasoner)** library from **nltk.sentiment** specifically because it is tuned towards social media, and how people type casually on the internet. Not only does it have a broad lexicon of sentiment words, but VADER also takes into account punctuation, such as exclamation points, and capitalization of text to capture emphasis within its scoring mechanism.

Because of this, I ran the sentiment analysis on the comment sentences *without* modifying the text. Usually text preprocessing for NLP indicates that you would standardize all characters to lowercase, and remove all punctuation to eliminate unnecessary characters, but I wanted to underscore how VADER's scoring would function on unaltered text. VADER's polarity score ranges from -1 to 1, with -1 being a completely negative review and 1 being a completely positive review. The **SentimentIntensityAnalyzer.polarity_scores** function outputs a 4 component dictionary, giving the negative, neutral, and positive numerical component scores for

a string, followed by a compound overall score for the string. Because I had broken down the text into individual sentences, only the compound score was used for this project, with one score for each sentence within a review.

Upon breaking out the sentences and computing the polarity scores, I was left with a list of numbers indicating the sentiment polarity of each sentence in the review. The length of the string was equal to the number of sentences in the review. All numbers ranged from -1 to 1, but this did not correlate to the 10 point scale given to the Airbnb categories. From there I engineered the feature by creating a function to transform the values into corresponding integer values. Once again, one integer was present for each sentence, based on the polarity score.

It was at this point, after VADER had fully interpreted the sentiment, that the comment text was tokenized, stop words were removed and the remaining text was lemmatized. By performing these tasks, only relevant text would be evaluated for topic related content. Similar to my teammates' process above, I used a word map with a list of trigger words for each of the relevant categories. To create the word map, I began by generating a wordcloud similar to the image that Paul displayed above, and I ensured that any subjective keywords were put in the correct category. I also scanned through the entirety of my assigned reviews to determine if there were additional words to capture. Once the word map was completed, all lemmatized text was evaluated, and when a trigger word was found, that subject was captured for that particular sentence. Although it is possible for a sentence to trigger multiple topics, I only captured the first topic.

Finally, for each review (or each dataframe row), I created a dictionary that captured an average of the binned polarity scores for each subject. For example, if a review had two sentences covering cleanliness, the 'cleanliness' key would contain an average of the two related polarity scores. After the dictionaries were created, each key was expanded out into its own dataframe column, and the score for each listing was present. Where a review did not mention a particular category, I kept the "NaN" value so that that particular row would not be calculated in the corresponding category mean. The final results are shown below.

Results: VADER Analysis

Sample 3

	Mean Abs Diff
accuracy_diff	2.748379
cleanliness_diff	1.601139
checkin_diff	2.516216
communication_diff	2.491152
location_diff	2.404127
value_diff	1.966738

Sample 1

	Mean Abs Diff
accuracy_diff	2.643473
cleanliness_diff	1.585339
checkin_diff	2.572225
communication_diff	2.383438
location_diff	2.466437
value_diff	1.823578

Sample 2

	Mean Abs Diff
accuracy_diff	2.699403
cleanliness_diff	1.614411
checkin_diff	2.506694
communication_diff	2.448135
location_diff	2.436137
value_diff	1.930338

Conclusions

Considering the complexity of NLP and Sentiment Analysis, the attempt to determine ratings for six distinct categories, and the skewed ratings numbers in the data set, our results are encouraging.

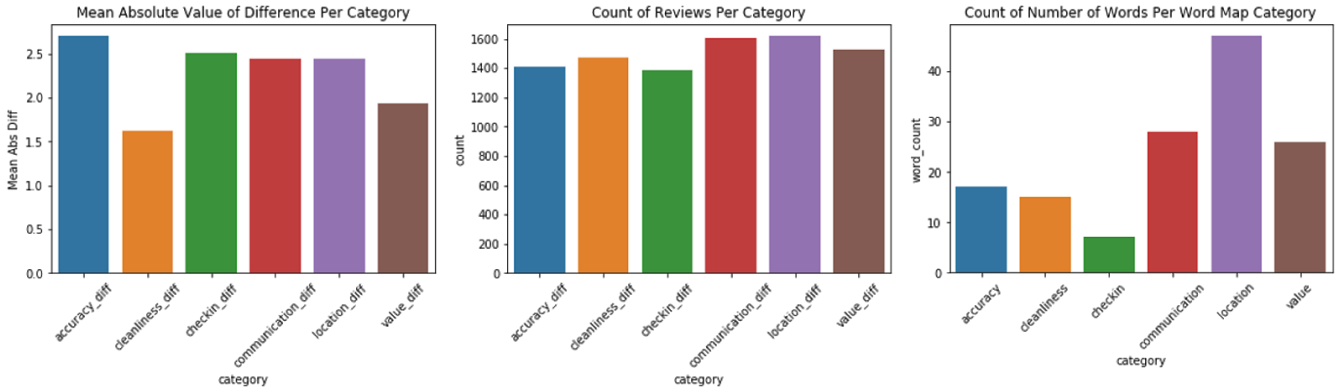
Paul took a naive approach to predicting the ratings, using single word analysis from individual sentences and sets of trigger words to determine category and sentiment. Iterative improvements to the trigger and sentiment word sets resulted in improved results for the binning of the sentences into the appropriate category, but the weights of the sentiment results had to be absurdly adjusted to match the skewed rating data to achieve mean differences near 2, making this model meaningless for more normally distributed data.

Gina employed TextBlob's sentiment analysis tools. The two sentiment results returned from TextBlob, polarity and subjectivity, were combined using a scoring function. While this scoring

was consistently applied, it was arbitrarily contrived based on trial and error to obtain reasonable results to match the data. The sentiment results from TextBlob seemed fairly accurate and consistent with other methodologies.

Alison used VADER sentiment analysis, hypothesizing that the increases in contextual polarity analysis would lead to more accurately categorized data. Ultimately, the VADER library did provide consistently accurate representation of the sentiment, but the results were not out of the realm of either of the other 2 methodologies.

Ultimately, we did not want to see results where the differences could be explained away by the thoroughness of the word map, or the number of reviews, and looking at the charts below, where we can see the absolute mean difference for each category against both the review counts and the word map counts for the same categories, (from the VADER implementation) it looks as if there is an additional silent factor that determines the true difference between the mean review values for each category.



This means that all of our methodologies have some merit in predicting the review ratings, and that although the ratings shown by Airbnb are high, the reviews do reflect similar content as far as subject matter and corresponding emotion.

The bottom line, improvements are being made all the time. Libraries such as VADER and TextBlob are refining their scoring mechanisms to represent the unique manner of communication today. The utilization of punctuation, capitalization and other techniques such as emoji representation will allow NLP to extract meaning from text in fewer and fewer characters. Additionally, refining techniques such as aspect based sentiment analysis, or the user analysis models mentioned in the literature review will allow condensed text passages to hold a wealth of

information. And what of the Airbnb user claims that hosts can artificially inflate their review scores? At least in the Chicago data we reviewed, there were no widespread indications that a host with a horrible listing was receiving artificially inflated reviews. It may just be that Airbnb has worked to refine their listings and quickly remove any that do not meet their standards.

References

Curry, David. “Airbnb Revenue and Usage Statistics (2021).” *Business of Apps*, 28 Feb. 2021, www.businessofapps.com/data/airbnb-statistics.

Kiatkawsin, Kiattipoom, et al. “A Comparative Automated Text Analysis of Airbnb Reviews in Hong Kong and Singapore Using Latent Dirichlet Allocation.” Tourism Industry Data Analytics Lab (TIDAL), Department of Hospitality and Tourism Management, Sejong University, Seoul 05006, Korea, 20 July 2020, https://www.mdpi.com/2071-1050/12/16/6673

Mann, Sonya. “Why Airbnb Reviews Don’t Tell the Whole Story.” *Inc.Com*, 6 Feb. 2020, www.inc.com/sonya-mann/airbnb-nightmare-experiences.html.

Mulshine, Molly. “After a Disappointing Airbnb Stay, I Realized There’s a Major Flaw in the Review System.” *Business Insider*, 18 June 2015, www.businessinsider.com/why-airbnb-reviews-are-a-problem-for-the-site-2015-6#:~:text=Then%20I%20understood%20one%20reason,future%20hosts%20depended%20on%20it.

Nadkarni, Prakash, et al. “Natural Language Processing: An Introduction.” *US National Library of Medicine National Institutes of Health*, Journal of the American Medical Informatics Association, 2011, www.ncbi.nlm.nih.gov/pmc/articles/PMC3168328.

“Quotas and Limits | Cloud Translation |.” *Google Cloud*, cloud.google.com/translate/quotas. Accessed 14 Mar. 2021.

Wang, Bingkun, et al. “Combining Review Text Content and Reviewer-Item Rating Matrix to Predict Review Rating.” *PubMed Central (PMC)*, Computer Intelligence Neuroscience, 3 Jan. 2016, www.ncbi.nlm.nih.gov/pmc/articles/PMC4735905.