Course Project
Airbnb Review Analysis
CS410 - Sravan Kumar Gardas

Code file - https://sgardas2.github.io/CS410/AirBnb_Review_Analysis.py

Airbnb Review Analysis

After studying Airbnb Reviews and also after using Airbnb as a guest when I travel. I have understood that reviews are very important for guests to book a place.

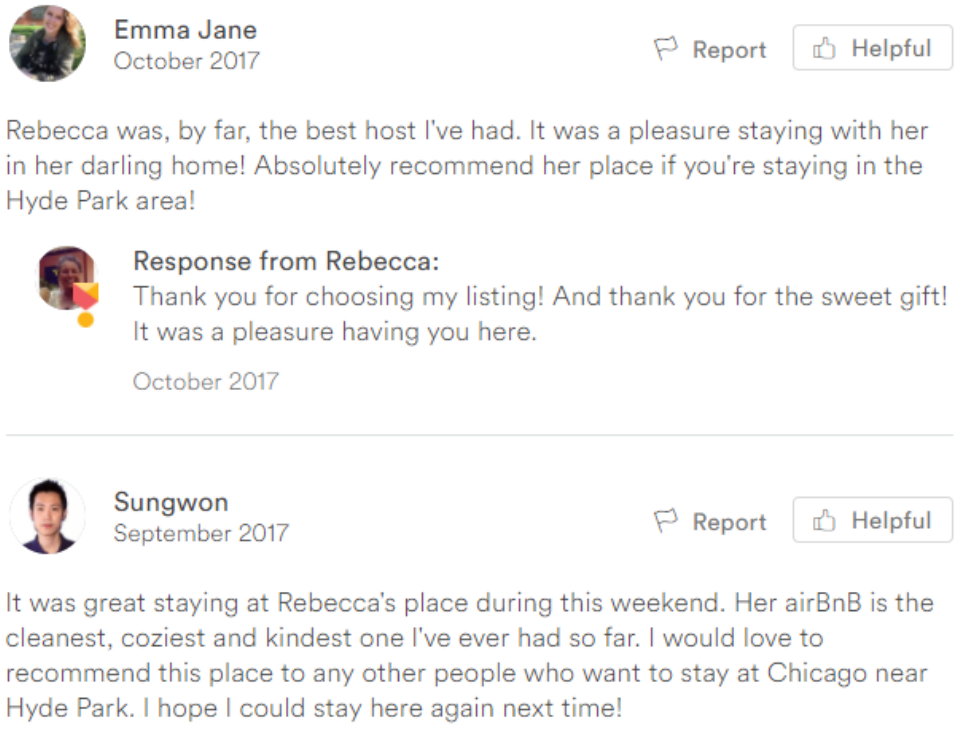
I picked Chicago data from this link

http://insideairbnb.com/get-the-data.html

About the data

1) Listings File- Contains the data about the listings, price, rate, rating based on reviews.
2) Reviews File – Contains the reviews by user per listing with date
3) Calendar file – Contains if the listing is available, its price and date
4) Neighbor hoods- The different neighbor hoods of Chicago

Example of the reviews



Step1 – Reading the data and Data Cleanup

There are two important files

- Listings File has 95 fields about the listings, the important fields considered have the information about the about the listings and the composite ratings overall over 0-100[100 being highest) also Composite ratings by Accuracy, cleanliness, location, check-in and value over 0-10[10 being highest).
- Reviews Files has six fields (id , listing_id, date, reviewer_id, r eviewer_name, comments)

reviews=pd.read_csv("C:/reviews.csv") (Here I am reading the huge Reviews file for Chicago which has 140k reviews for all the listing)

listing=pd.read_csv("c:/listings.csv") (Here I am reading the listing file which has all the listings in Chicago land – around 6000)

Step2- Language Identification (output 1)

- Using the NLTK stopwords for identifying the language .
- Assessing the language for every comment and then sorting the languages in reverse order picking the first assigned language.
- This is giving 98.5% accuracy, small phrases which don't have stop words are not being judged correctly.
- Created a dataframe with all the language identified for all the reviews and the output of this dataframe is uploaded here in CSV zipped file OutPut-Language_Classfied_Reviews.zip

Step3 – Filtered on English language

- Filtered all the reviews on English language and then applied the SentimentIntensityAnalyzer of NLTK Vader package for each comment.
- s_filter=s[(s.commentexist==True)&(s.language=='english')]

Step4 – Normalized the positive sentiment scores for every rating

- Got the sum of positive sentiment scores for the listing by summing the positive sentiment and then normalizing this by the number of reviews per listing. Used the NLTK Vader Sentiment package for sentiment classification.
- rev_group=scored_reviews.groupby('listing').apply(lambda df: (sum([positivity for positivity in df['positivity'].values]))/len([positivity for positivity in df['positivity'].values]))

Step5- Merging these sentiment scores with the listing (output 2)

- The scores of the Sentiment are then merged with the Listing ratings and the output is then saved in a CSV file here https://sgardas2.github.io/CS410/Output-NormalizedSentiment-Listing.csv
- **Below is snapshot of the output sentiment file**

id	sentiment	numbe	review	review	review	review	review	review	review	price	neighb
37738	0.306346939	196	100	10	10	10	10	10	10	$74.00	Uptown
39742	0.313229462	353	93	9	9	10	10	10	10	$75.00	Near North
80640	0.305560976	41	97	10	10	10	10	10	10	$155.00	Logan Squa
95544	0.314914894	47	97	10	9	10	10	10	9	$150.00	Logan Squa
110705	0.310470588	102	97	10	10	10	10	9	10	$99.00	Uptown
189821	0.312958904	292	99	10	10	10	10	10	10	$150.00	Logan Squa
192652	0.33753719	242	97	10	10	10	10	10	9	$65.00	West Town
199681	0.301989247	93	92	9	8	10	10	9	9	$50.00	West Ridge
207116	0.315013605	147	94	9	9	10	10	9	10	$45.00	West Tow
220660	0.30475	148	95	10	9	10	10	9	9	$45.00	West Town
233933	0.308566372	113	95	10	10	10	10	9	9	$55.00	Irving Park
241514	0.31653125	96	98	10	10	10	10	10	10	$55.00	Edgewater
257601	0.322960854	281	95	9	9	9	10	10	9	$33.00	Loop

Step6- Top3 Bigrams collocation for each listing based on the reviews (output 3)

- Reviews of listings have high frequent words such as “it could be closer to a Blue-line train or major attraction”. This is an interesting feature. The outputs of the bi-gram collocation are uploaded here. https://sgardas2.github.io/CS410/Output-Bigrams-Listing.csv

id	bigrams
2384	[('public', 'transport'), ('block', 'away'), ('went', 'above')]
4505	[("stone's", 'throw'), ('Little', 'Village'), ('record', 'player')]
6715	[('URL', 'HIDDEN'), ('twin', 'beds'), ('year', 'old')]
7126	[('frying', 'pans'), ('cell', 'phone'), ('Blue', 'Line')]
9811	[('Lincoln', 'Park'), ('walking', 'distance'), ('Great', 'location')]
10610	[('I', 'did'), ('we', 'were'), ('close', 'to')]
12068	[('Gold', 'Coast'), ('Old', 'Town'), ('red', 'line')]
12140	[('Lincoln', 'Park')]
24833	[('DePaul', 'University'), ('long', 'term'), ('Lincoln', 'Park')]
25879	[('timely', 'manner'), ('last', 'minute'), ('queen', 'bed')]
28749	[("don't", 'mind'), ('Wicker', 'Park'), ('blue', 'line')]
32346	[]
37738	[('THANK', 'YOU'), ('Hop', 'Leaf'), ('five', 'star')]
39742	[('automated', 'posting'), ('History', 'Museum'), ('Navy', 'Pier')]
41422	[('William', 'Laurence'), ('highly', 'recommend'), ('mate', 'William')]
41425	[('The', 'location'), ('make', 'our'), ('location', 'is')]
44020	[('automated', 'posting'), ('water', 'pressure'), ('exactly', 'what')]
71930	[('Wicker', 'Park'), ('laid', 'back'), ('right', 'across')]
79101	[('Highly', 'recommended'), ('Green', 'Mill'), ('My', 'husband')]
80640	[('Highly', 'recommended'), ('Blue', 'Line'), ('Longman', 'Eagle')]
84042	[('highly', 'recommend'), ('recommend', 'staying'), ("Michael's", 'place')]

The Code File

- Here is the code file in Python using Pandas and NTLK package.
- If we run the code with the entire reviews set it might take 30-40mins. We could see the sample output by limiting the reviews to first 2500 records
- reviews=pd.read_csv("C:/reviews.csv")[0:2500]
- https://sgardas2.github.io/CS410/AirBnb_Review_Analysis.py