

## Project Final Report

# Hotel Review Sentiment Analysis

Team Mr. Goose

Runyu Cao ([runyuc2@illinois.edu](mailto:runyuc2@illinois.edu)) (Captain)

Weijie Wang ([weijiew2@illinois.edu](mailto:weijiew2@illinois.edu))

Qijing Zhu ([qijingz2@illinois.edu](mailto:qijingz2@illinois.edu))

<b>1. Overview</b>	<b>3</b>
1.1 Background	3
1.2 The Dataset	3
1.3 Algorithm	3
<b>2. Implementation</b>	<b>4</b>
2.1 Data Loading	4
2.2 Data Cleaning	4-5
2.3 Feature Engineering - Part 1	5
2.4 Model Training - Part 1	6
2.5 Model Evaluation - Part 1	6
2.6 Feature Engineering - Part 2	6
2.7 Model Training - Part 2	7
2.8 Model Evaluation - Part 2	8
2.9 Evaluate with customized input	9
<b>3. Usage</b>	<b>9-10</b>
<b>4. Recommendation</b>	<b>10</b>
<b>5. Contribution</b>	<b>10</b>

# 1. Overview

## 1.1 Background

Hotel reviews are one of the most important factors influencing a person's booking selection and thus have a high impact on marketing strategies. Our team has chosen to utilize sentiment analysis, a technique of extracting emotions based on the selected hotels' textual reviews and produce feedback and solutions to benefit travelers by comparing different options. A challenge we will be discussing is whether a sentiment analysis model learned on hotel reviews can process different kinds of text across domains.

## 1.2 The Dataset

We use the dataset from [Hotel Reviews Data in Europe](#). This dataset contains 515,000 customer reviews and scoring of 1493 luxury hotels across Europe. Due to the limitation of our local computational resources, we only took the first 100k rows. We separated them to be used as the training, validation, and testing data.

Disclaimer: The data was scraped from Booking.com. All data in the file is publicly available to everyone already. Please be noted that data is originally owned by Booking.com.

## 1.3 Algorithm

Algorithms we used for this project include but are not limited to Stemming, NGram Tokenization, Random Forest Regressor, and XGBoost Regressor.

## 2. Implementation

### 2.1 Data Loading

```
In [50]: %%time
raw_data = pd.read_csv('in_europe\Hotel_Reviews.csv')
raw_data = raw_data.drop_duplicates()
print(len(raw_data))

515212
Wall time: 4.41 s
```

```
In [51]: raw_data.head()
```

```
Out[51]:
```

	Hotel_Address	Additional_Number_of_Scoring	Review_Date	Average_Score	Hotel_Name	Reviewer_Nationality	Negative_Review	Review_1
0	Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	8/3/2017	7.7	Hotel Arena	Russia	I am so angry that i made this post available...	
1	Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	8/3/2017	7.7	Hotel Arena	Ireland	No Negative	
2	Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	7/31/2017	7.7	Hotel Arena	Australia	Rooms are nice but for elderly a bit difficul...	
3	Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	7/31/2017	7.7	Hotel Arena	United Kingdom	My room was dirty and I was afraid to walk ba...	
4	Gravesandestraat 55 Oost 1092 AA Amsterdam ...	194	7/24/2017	7.7	Hotel Arena	New Zealand	You When I booked with your company on line y...	

### 2.2 Data Cleaning

We eliminated irrelevant columns, made everything into lower case, split them into words, removed English stopwords, and stemmed the words. The data was later split into training and testing dataset with 80%-20% rate, respectively..

```
]: individual_review_drop_set = set(['Hotel_Address', 'Additional_Number_of_Scoring', 'Review_Date', 'Average_Score', \
    'Hotel_Name', 'Reviewer_Nationality', 'Review_Total_Negative_Word_Counts', \
    'Total_Number_of_Reviews', 'Review_Total_Positive_Word_Counts', \
    'Total_Number_of_Reviews_Reviewer_Has_Given', 'days_since_review', 'lat', 'lng'])
individual_review_raw_data = raw_data.drop(list(individual_review_drop_set), 1)

# Remove stopwords
individual_review_raw_data['Positive_Review'] = individual_review_raw_data['Positive_Review'].apply(
    lambda x: ' '.join([w for w in x.strip().lower().split() if w not in en_stop]))
individual_review_raw_data['Negative_Review'] = individual_review_raw_data['Negative_Review'].apply(
    lambda x: ' '.join([w for w in x.strip().lower().split() if w not in en_stop]))

# Stemming
ps = PorterStemmer()
individual_review_raw_data['Positive_Review'] = individual_review_raw_data['Positive_Review'].apply(
    lambda x: ' '.join([ps.stem(y) for y in x.split()]))
individual_review_raw_data['Negative_Review'] = individual_review_raw_data['Negative_Review'].apply(
    lambda x: ' '.join([ps.stem(y) for y in x.split()]))

X = individual_review_raw_data.drop(["Reviewer_Score"], 1)
y = individual_review_raw_data["Reviewer_Score"]
raw_X_train, raw_X_test, y_train, y_test = train_test_split(X,y,test_size=0.2, random_state=310)
raw_X_train.head()

D:\programs\Anaconda3\envs\cs410\lib\site-packages\ipykernel_launcher.py:5: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
"""
D:\programs\Anaconda3\envs\cs410\lib\site-packages\ipykernel_launcher.py:20: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
```

```
]:
```

	Negative_Review	Positive_Review	Tags
208	neg nois build work go time awar book stay hug...	staff extrem polit help honestli fault room cl...	['Leisure trip', 'Couple', 'Duplex Double...]
1682	noth	good locat help staff room spaciou well worth ...	['Leisure trip', 'Group', 'Superior King ...]
204	neg	good locat next tram go major revamp nois prob...	['Leisure trip', 'Couple', 'Duplex Double...]
150	sofa need bar better breakfast option water pr...	locat great us close sister alway someone desk ...	['Leisure trip', 'Group', 'Duplex Double ...]
6764	room took warm out air con high temp heat	staff friendli help	['Leisure trip', 'Couple', 'Superior Double ...]

## 2.3 Feature Engineering - Part 1

We then performed 1-3 gram(s) on the positive and negative reviews and picked 1000 of the most common grams as sentimental analysis features.

```
]: %%time
# NGram
ngram_cnt = collections.Counter(
    [a for a in ngrams(" ".join(raw_X_train['Positive_Review']).split(), 1)] + \
    [a for a in ngrams(" ".join(raw_X_train['Positive_Review']).split(), 2)] + \
    [a for a in ngrams(" ".join(raw_X_train['Positive_Review']).split(), 3)] + \
    [a for a in ngrams(" ".join(raw_X_train['Negative_Review']).split(), 1)] + \
    [a for a in ngrams(" ".join(raw_X_train['Negative_Review']).split(), 2)] + \
    [a for a in ngrams(" ".join(raw_X_train['Negative_Review']).split(), 3)])
print([(k, c) for k, c in ngram_cnt.most_common(10) if '|' not in k])
print(len(ngram_cnt))
features = [" ".join(k) for k, _ in ngram_cnt.most_common(1000) if '|' not in k]

[ (('room',), 7747), (('staff',), 3858), (('hotel',), 3413), (('locat',), 3349), (('breakfast',), 1779), (('bed',), 1768)]
236631
Wall time: 251 ms
```

## 2.4 Model Training - Part 1

We used the random forest regressor model as our initial model for its robustness and ability to play a role in further feature selection.

```
rf_model = RandomForestRegressor(n_estimators=500, max_depth=25, n_jobs=-1, min_samples_split=10, min_samples_leaf=2)
rf_model.fit(X_train, y_train)
```

## 2.5 Model Evaluation - Part 1

We compared the model msr with baseline, and based on the data our model shows it is learning (indicating signs of learning). We also tested a selection of our own inputs with positive results.

```
] import random
r = []
for i in range(len(y_test)):
    n = random.randint(1,10)
    r.append(n)
print("baseline msr:", mean_squared_error(r, y_test.astype('float')))
print("model msr:", mean_squared_error(rf_model.predict(X_test), y_test.astype('float')))
```

baseline msr: 18.599230000000002  
model msr: 1.6041460983597615

## 2.6 Feature Engineering - Part 2

The scikit-learn RandomForestRegressor provides a very convenient way to extract feature importance by the role they play in the forest.

```
] fi = rf_model.feature_importances_
imp_fea = []
for i in range(len(fi)):
    imp_fea.append((X_train.columns[i], fi[i]))
imp_fea = sorted(imp_fea, key=lambda t: t[1], reverse=True)
print(len(imp_fea))
imp_fea
```

3316

```
] [ ('feature_n_room', 0.20181664150728615),
  ('feature_p_e', 0.11350931812595522),
  ('feature_n_c', 0.04374767376990346),
  ('feature_n_st', 0.02590620887322868),
  ('feature_p_staff', 0.01938880315020767),
  ('feature_n_staff', 0.018517794517554757),
  ('feature_p_v', 0.013803749904757445),
  ('feature_n_v', 0.013282975017032595),
  ('feature_p_noth', 0.012155106219173872),
  ('feature_n_hotel', 0.010235203493887195),
  ('feature_n_b', 0.009572667786459448),
  ('feature_n_u', 0.008780508703744864),
  ('feature_p_u', 0.008252742716570899),
  ('feature_p_room', 0.007129985895633),
  ('feature_n_clean', 0.006003562940194162),
  ('feature_p_com', 0.005957907587482291),
  ('feature_p_st', 0.00548804291493209),
```

We sort the features by their importance and pick the top 1000 features for later use.

```
] adjusted_X_train = X_train.drop([f[0] for f in imp_fea[1000:]], 1)
adjusted_X_test = X_test.drop([f[0] for f in imp_fea[1000:]], 1)
```

## 2.7 Model Training - Part 2

We then fit a GradientBoostRegressor on the training data. As you can see from the image below, the last step helped improve our MSR and training time.

```
] %%time
xgb_model = GradientBoostRegressor(n_estimators=2000, learning_rate=0.05, max_depth=3)
xgb_model.fit(X_train, y_train)
print("model msr:", mean_squared_error(xgb_model.predict(X_test), y_test.astype('float')))
```

model msr: 1.472844140007202  
Wall time: 9min 6s

```
] %%time
xgb_model = GradientBoostRegressor(n_estimators=2000, learning_rate=0.05, max_depth=3)
xgb_model.fit(adjusted_X_train, y_train)
print("model msr:", mean_squared_error(xgb_model.predict(adjusted_X_test), y_test.astype('float')))
```

model msr: 1.4517227853842232  
Wall time: 3min

We then fine-tune our xgboost model on tree count, learning rate and max depth of trees using training data only.

```
] adjusted_X_train_2, adjusted_X_validation, y_train_2, y_validation = train_test_split(adjusted_X_train, y_train, test_size=0.2, random_state=42)
for n_estimator in [100, 500, 1000, 2000, 5000]:
    for learning_rate in [0.01, 0.05, 0.1, 0.5, 1]:
        for max_depth in [3, 5, 7, 10, 20]:
            xgb_model = GradientBoostRegressor(n_estimators=n_estimator, learning_rate=learning_rate, max_depth=max_depth)
            xgb_model.fit(adjusted_X_train_2, y_train_2)
            print(n_estimator, learning_rate, max_depth, "model msr:", mean_squared_error(xgb_model.predict(adjusted_X_validation), y_validation))
```

100 0.01 3 model msr: 2.179150058587754  
100 0.01 5 model msr: 2.018214234285366  
100 0.01 7 model msr: 1.9598600540397038  
100 0.01 10 model msr: 1.9151585487430085  
100 0.01 20 model msr: 1.8668700011000000

After all the attempts, we selected 5000 as num of estimators, 0.01 as learning\_rate, and 3 as max\_depth

```

2000 0.05 10 model msr: 1.6112815286291093
2000 0.05 20 model msr: 1.8041659562960928
2000 0.1 3 model msr: 1.50618360557577
2000 0.1 5 model msr: 1.6141944812398623
2000 0.1 7 model msr: 1.6799910418813684
2000 0.1 10 model msr: 1.6511865327797741
2000 0.1 20 model msr: 1.808313142362541
2000 0.5 3 model msr: 2.04467289812313
2000 0.5 5 model msr: 2.0566856884114197
2000 0.5 7 model msr: 2.054178643711046
2000 0.5 10 model msr: 2.0674279506904503
2000 0.5 20 model msr: 2.0249982569196154
2000 1 3 model msr: 3.0455458316250743
2000 1 5 model msr: 3.095174905907742
2000 1 7 model msr: 3.166702061555934
2000 1 10 model msr: 3.177734214299881
2000 1 20 model msr: 3.0707992145725864
5000 0.01 3 model msr: 1.4318329778583951
5000 0.01 5 model msr: 1.4448606173294445
5000 0.01 7 model msr: 1.4845252443082817
5000 0.01 10 model msr: 1.5392938406239816
5000 0.01 20 model msr: 1.7999706551877093
5000 0.05 3 model msr: 1.5244819037730422
5000 0.05 5 model msr: 1.6318398062717756
5000 0.05 7 model msr: 1.6761332826167472
5000 0.05 10 model msr: 1.6225995125284467
5000 0.05 20 model msr: 1.7939042894041846
5000 0.1 3 model msr: 1.6407051996237085
5000 0.1 5 model msr: 1.7697292776075315
5000 0.1 7 model msr: 1.7167975378641642
5000 0.1 10 model msr: 1.6875264306497082
5000 0.1 20 model msr: 1.7966498369433024
5000 0.5 3 model msr: 2.2026569409198733
5000 0.5 5 model msr: 2.0539157868155797
5000 0.5 7 model msr: 2.115257335936386
5000 0.5 10 model msr: 1.9735052946908118
5000 0.5 20 model msr: 2.032470937885982
5000 1 3 model msr: 3.401001573151475

```

## 2.8 Model Evaluation - Part 2

We then used our final model to get the MSR for testing data. Our final MSR on the test set is 1.4321.

```

|: %%time
xgb_model = GradientBoostingRegressor(n_estimators=5000, learning_rate=0.01, max_depth=3)
xgb_model.fit(adjusted_X_train, y_train)
print(n_estimator, learning_rate, max_depth, "model msr:", mean_squared_error(xgb_model.predict(
5000 1 20 model msr: 1.4321332220719383
Wall time: 6min 29s

```



## 2.9 Evaluate with customized input

We also developed this simple tool to enable users to try out our model by typing in their own positive reviews and negative reviews. Here are some examples:

```
Positive Review: This hotel is awesome I took it sincerely because a bit cheaper but the  
h close to one awesome park Arrive in the city are like 10 minutes by tram and is super e  
and really cool and the room is incredible nice with two floor and up one super big comf  
sure there The staff very gentle one Spanish man really really good  
Negative Review: No Negative  
Our best estimate of your rating is: 7.257117349604184
```

```
Positive Review: No positive  
Negative Review: I am so angry that i made this post available via all possible sites i  
one will make the mistake of booking this place I made my booking via booking com We stay  
from 11 to 17 July Upon arrival we were placed in a small room on the 2nd floor of the hot  
not the room we booked I had specially reserved the 2 level duplex room so that we would  
ilings The room itself was ok if you don t mind the broken window that can not be closed t  
at contained some sort of a bio weapon at least i guessed so by the smell of it I intimate  
d after explaining 2 times that i booked a duplex btw it costs the same as a simple double  
o the high ceiling was offered a room but only the next day SO i had to check out the next  
r to get the room i waned to Not the best way to begin your holiday So we had to wait till  
new room what a wonderful waist of my time The room 023 i got was just as i wanted to pe  
window We were tired from waiting the room so we placed our belongings and rushed to the  
out that there was a constant noise in the room i guess it was made by vibrating vent tube  
and annoying as hell AND it did not stop even at 2 am making it hard to fall asleep for r  
recording that i can not attach here but if you want i can send it via e mail The next day  
ot able to determine the cause of the disturbing sound so i was offered to change the room  
y booked and they had only 1 room left the one that was smaller but seems newer  
Our best estimate of your rating is: 5.466495537047969
```

One issue we noticed is that since most of the input score ranges from 7.0-9.0, our estimated score will lie primarily in this range, though with additional data augmentation it would be possible to train our model to give out scores in a wider range.

## 3. Usage

To be able to run our Jupyter notebook and evaluation tool, you will need to prepare a Python 3.7 environment with preferably anaconda. Here is a part of our local anaconda dependency .yaml file that should be able to generate an env supports our project:

```
name: cs410  
channels:  
- defaults  
dependencies:  
- ipykernel=6.2.0=py37haa95532_1  
- ipython=7.26.0=py37hd4e2768_0  
- jupyter=1.0.0=py37_7  
- nltk=3.6.5=pyhd3eb1bo_0  
- notebook=6.4.3=py37haa95532_0  
- numpy=1.20.3=py37ha4e8547_0  
- numpy-base=1.20.3=py37hc2deb75_0
```

```
- pandas=1.3.4=py37h6214cd6_o
- python=3.7.11=h6244533_o
- regex=2021.8.3=py37h2bbff1b_o
- scikit-learn=1.0.1=py37hf11a4ad_o
- scipy=1.7.1=py37hbe87c03_2
```

You can use our evaluation tool by simply putting it along with our model info file `cs410\_final\_mmodel\_info.joblib` in the same folder and then run it with Python.

## 4. Recommendation

Due to time limitations, we were unable to explore further, but some highlighted future steps including the following:

1. Normalize score
2. Using BM25 score as feature value
3. Adopting aspect inferring
4. Trying out entity extraction
5. Trying out some popular recurrent neural networks such as LSTM
6. Model fusion

## 5. Contribution

Name	UIUC ID	Contribution
Runyu Cao	runyuc2	Dataset selection, Data cleaning, Model evaluation, Tutorial presentation, Report
Weijie Wang	weijiew2	Data cleaning, Feature engineering, Model training, Model parameter tuning, Model evaluation, Report
Qijing Zhu	qijingz2	Feature engineering, Model evaluation, Tutorial presentation, Report