Configuration Manual

MSc Research Project
Data Analytics

Neha Suryawanshi
Student ID: x20169531

School of Computing
National College of Ireland

Supervisor: Majid Latifi

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Neha Suryawanshi		
Student ID:	X20169531		
Programme:	MSc in Data Analytics	**Year:**	2021
Module:	MSc Research Project		
Lecturer:	Majid Latifi		
Submission Due Date:	31/01/2022		
Project Title:	Analysis of Theme Impact in Consumer Reviews using Natural Language Processing Techniques		
Word Count:	1317	**Page Count:**	12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Neha Suryawanshi

Date: 31/01/2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Neha Suryawanshi
Student ID: x20169531

1 Introduction

Various components required for the research have been documented in this configuration manual. The specification for hardware as well as software, environment setup and code artefact snapshots have been documented as follows to give a brief idea about the steps performed to implement the project and achieve the aims of the research. The main tasks performed are:

i. Data selection & collection
ii. Data cleaning and pre-processing
iii. Data Transformation
iv. Data Mining
v. Aspect Extraction
vi. Sentiment Analysis
vii. Final Evaluation

2 System Configuration

2.1 Hardware

The hardware component used for the project implementation are as follows in Table 1.

Table 1- Hardware Components

Machine Model	HP Pavilion
Operating System	Windows
Processor	Intel(R) Core (TM) i7-10510U CPU @ 1.80GHz 2.30 GHz
RAM	8.00 GB
Graphics	None
GPU	None

2.2 Software

Below Table 2 mentions the tools, libraries and programming language used in this research project.

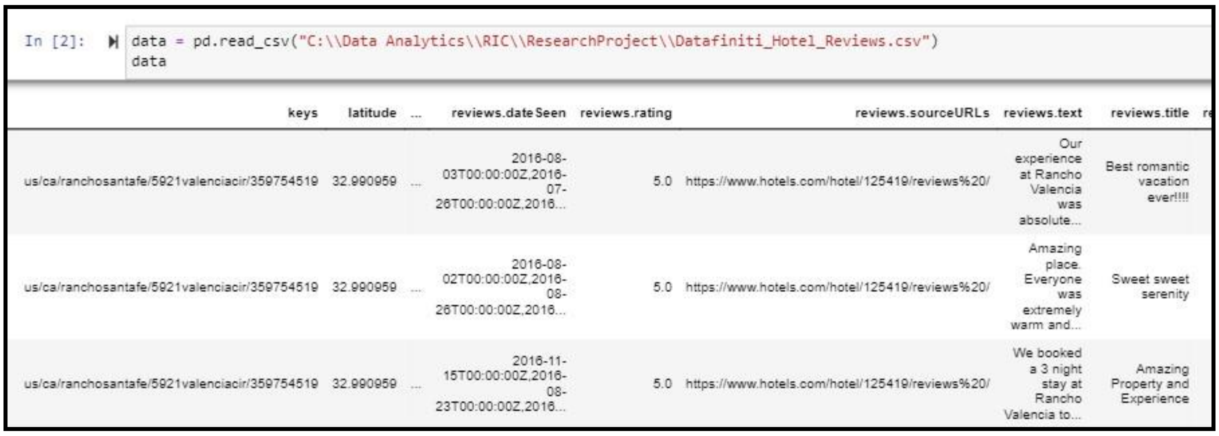
Table 2 - Software Components

Tools	Jupiter Notebook Excel Text Notebook
Libraries	Langdetect Gensim Textblob spaCy Matplotlib

3 Project Development

3.1 Data Selection and Collection

The Datafiniti's Business Database has made a hotel review dataset of 1000 reviews available on Kaggle[1]. This dataset in in csv format and hence following code is used to load the data in the data frame. The code and structure of data frame is shown in Figure 1.



```
In [2]:  data = pd.read_csv("C:\\Data Analytics\\RIC\\ResearchProject\\Datafiniti_Hotel_Reviews.csv")
         data
```

keys	latitude	...	reviews.dateSeen	reviews.rating	reviews.sourceURLs	reviews.text	reviews.title
us/ca/ranchosantafe/5921valenciacir/359754519	32.990959	...	2016-08-03T00:00:00Z,2016-07-26T00:00:00Z,2016...	5.0	https://www.hotels.com/hotel/125419/reviews%20/	Our experience at Rancho Valencia was absolute...	Best romantic vacation ever!!!!
us/ca/ranchosantafe/5921valenciacir/359754519	32.990959	...	2016-08-02T00:00:00Z,2016-08-26T00:00:00Z,2016...	5.0	https://www.hotels.com/hotel/125419/reviews%20/	Amazing place. Everyone was extremely warm and...	Sweet sweet serenity
us/ca/ranchosantafe/5921valenciacir/359754519	32.990959	...	2016-11-15T00:00:00Z,2016-08-23T00:00:00Z,2016...	5.0	https://www.hotels.com/hotel/125419/reviews%20/	We booked a 3 night stay at Rancho Valencia to...	Amazing Property and Experience

Figure 1- Data loading

3.2 Data Cleaning and Pre-processing

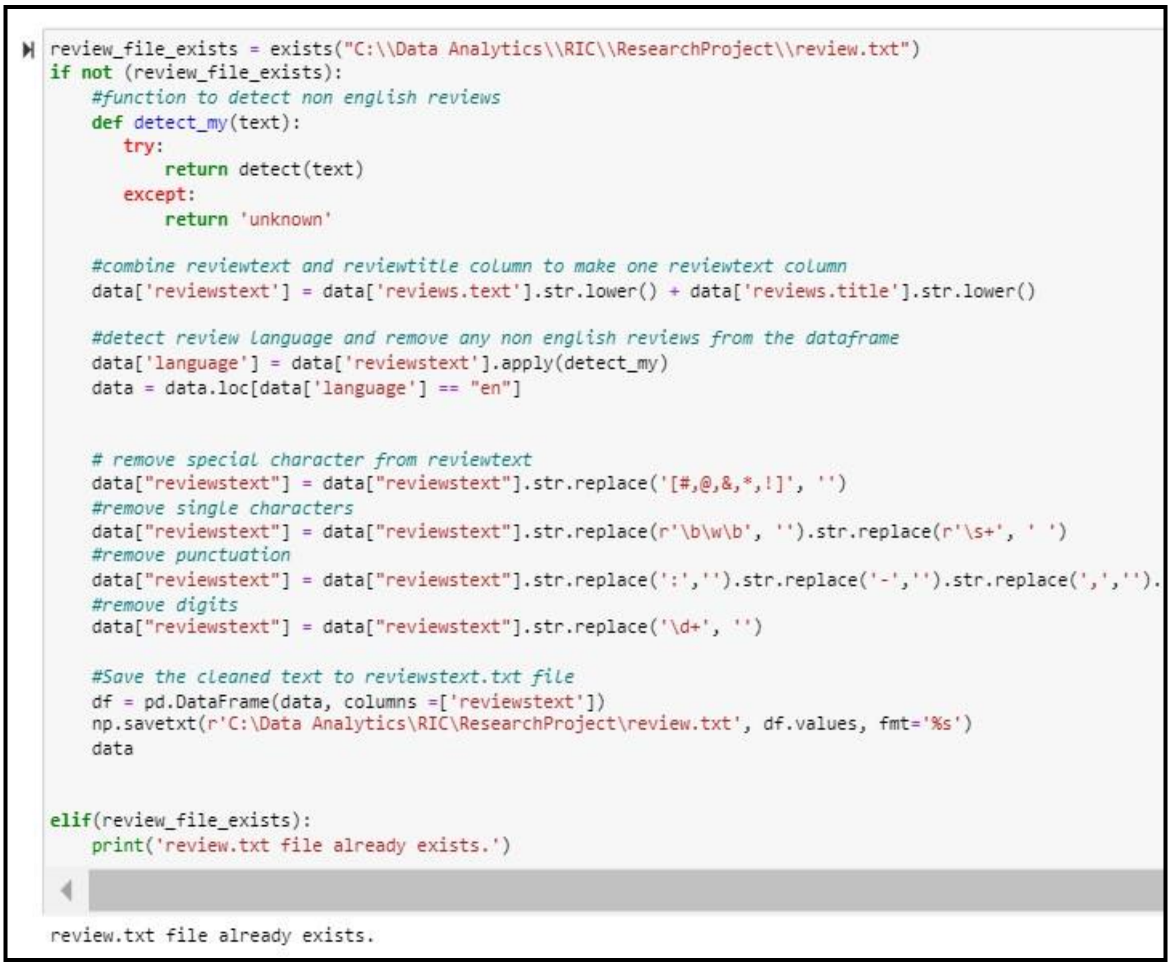
The two columns, reviews.text and reviews.title in the dataset are observed to the hotel guest reviews. Since these two columns are useful for the research they are combined into one column to make the data cleaning and pre-processing easier. Using the langdetect library the non-English reviews are filtered out from the data frame. Further the data is cleaned as per below:

- Special characters such as '@', '#', '!', etc., removal
- Single characters removal

[1] https://www.kaggle.com/datafiniti/hotel-reviews

- Punctuation removal (except full stop)
- Digit removal

All these reviews are stored in a text file called review.txt. The objective to create a new file is to make the code reproducible and avoid any unnecessary code run if the clean review file already exists. This saves time and resources. The snapshot of the code used to achieve this task is below in Figure 2.

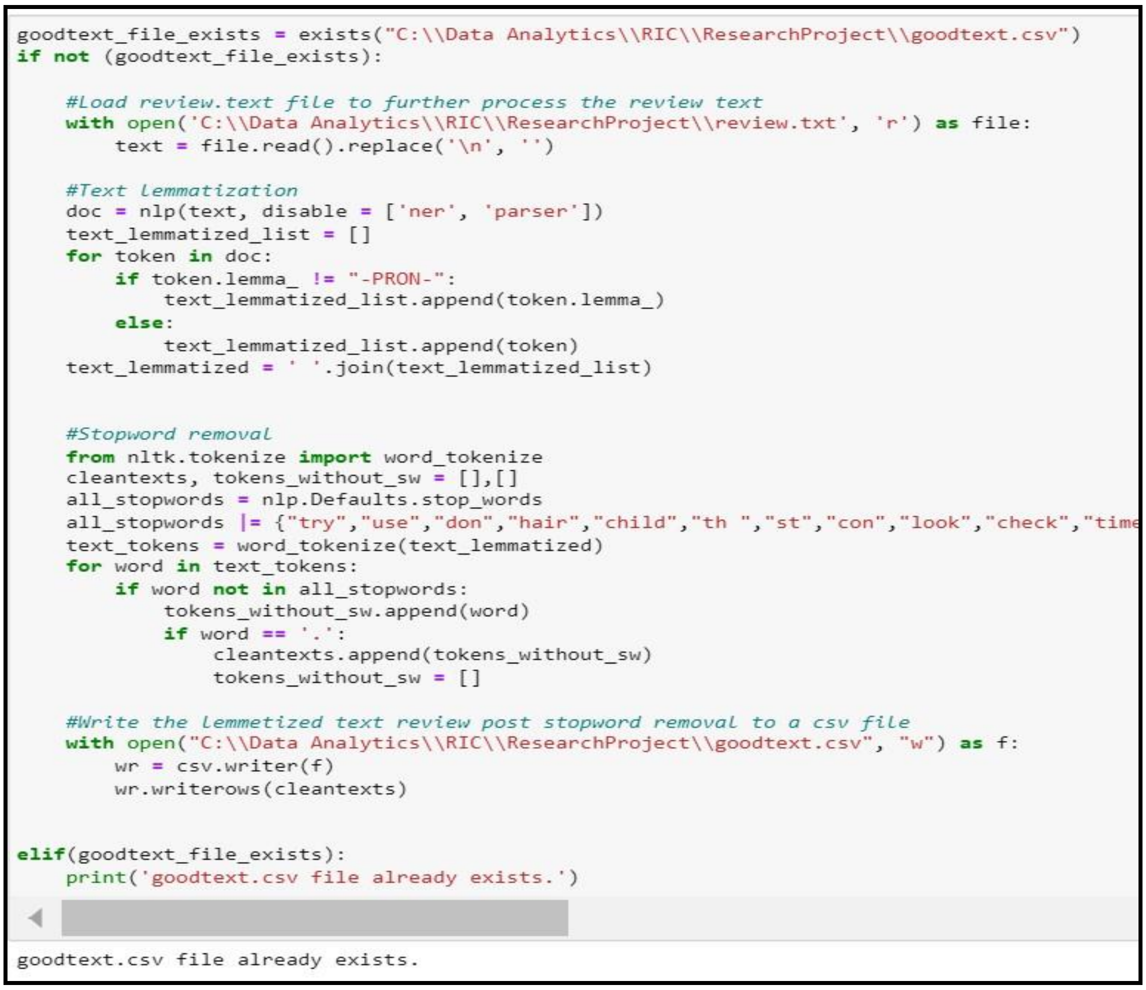


```
review_file_exists = exists("C:\\Data Analytics\\RIC\\ResearchProject\\review.txt")
if not (review_file_exists):
    #function to detect non english reviews
    def detect_my(text):
        try:
            return detect(text)
        except:
            return 'unknown'

    #combine reviewtext and reviewtitle column to make one reviewtext column
    data['reviewstext'] = data['reviews.text'].str.lower() + data['reviews.title'].str.lower()

    #detect review language and remove any non english reviews from the dataframe
    data['language'] = data['reviewstext'].apply(detect_my)
    data = data.loc[data['language'] == "en"]


    # remove special character from reviewtext
    data["reviewstext"] = data["reviewstext"].str.replace('[#,@,&,*,!]', '')
    #remove single characters
    data["reviewstext"] = data["reviewstext"].str.replace(r'\b\w\b', '').str.replace(r'\s+', ' ')
    #remove punctuation
    data["reviewstext"] = data["reviewstext"].str.replace(':','').str.replace('-','').str.replace(',','').
    #remove digits
    data["reviewstext"] = data["reviewstext"].str.replace('\d+', '')

    #Save the cleaned text to reviewstext.txt file
    df = pd.DataFrame(data, columns =['reviewstext'])
    np.savetxt(r'C:\Data Analytics\RIC\ResearchProject\review.txt', df.values, fmt='%s')
    data


elif(review_file_exists):
    print('review.txt file already exists.')
```

```
review.txt file already exists.
```

Figure 2- Data Cleaning(1)

The cleaning task continues with lemmatization and tokenization of text and also the removal of stop words. For this task the spaCy library is used. Few of the custom stop words are also added to the list of stop words available with spaCy library. The revies are then appended to a list of list where review is seperated on the basis of full stop. Here each sentence is the considered as one review which helps do the analysis on a granular level. These words are discovered during the initail phase of LDA topic modelling. These words have been added after analysing the frequent words in the review set which do not add any value in the textual analysis. Post these steps the tokenised set of reviews are saved in a goodtext.csv file for the same reason stated above. Code snap shot for the same is below in Figure 3.

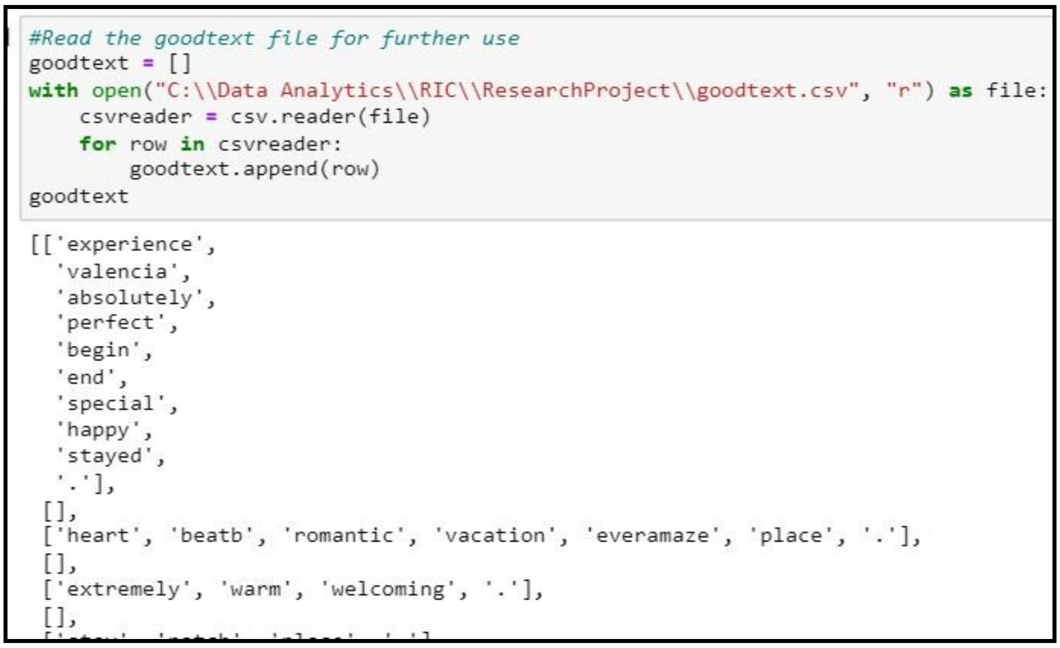


```
goodtext_file_exists = exists("C:\\Data Analytics\\RIC\\ResearchProject\\goodtext.csv")
if not (goodtext_file_exists):

    #Load review.text file to further process the review text
    with open('C:\\Data Analytics\\RIC\\ResearchProject\\review.txt', 'r') as file:
        text = file.read().replace('\n', '')

    #Text Lemmatization
    doc = nlp(text, disable = ['ner', 'parser'])
    text_lemmatized_list = []
    for token in doc:
        if token.lemma_ != "-PRON-":
            text_lemmatized_list.append(token.lemma_)
        else:
            text_lemmatized_list.append(token)
    text_lemmatized = ' '.join(text_lemmatized_list)


    #Stopword removal
    from nltk.tokenize import word_tokenize
    cleantexts, tokens_without_sw = [],[]
    all_stopwords = nlp.Defaults.stop_words
    all_stopwords |= {"try","use","don","hair","child","th ","st","con","look","check","time
    text_tokens = word_tokenize(text_lemmatized)
    for word in text_tokens:
        if word not in all_stopwords:
            tokens_without_sw.append(word)
            if word == '.':
                cleantexts.append(tokens_without_sw)
                tokens_without_sw = []

    #Write the Lemmetized text review post stopword removal to a csv file
    with open("C:\\Data Analytics\\RIC\\ResearchProject\\goodtext.csv", "w") as f:
        wr = csv.writer(f)
        wr.writerows(cleantexts)


elif(goodtext_file_exists):
    print('goodtext.csv file already exists.')
```

```
goodtext.csv file already exists.
```

Figure 3- Data Cleaning (2)

Figure 4 below shows how the textual data looks like after its imported from the goodtext.csv file into a list.



```
#Read the goodtext file for further use
goodtext = []
with open("C:\\Data Analytics\\RIC\\ResearchProject\\goodtext.csv", "r") as file:
    csvreader = csv.reader(file)
    for row in csvreader:
        goodtext.append(row)
goodtext
```

```
[['experience',
  'valencia',
  'absolutely',
  'perfect',
  'begin',
  'end',
  'special',
  'happy',
  'stayed',
  '.'],
 [],
 ['heart', 'beatb', 'romantic', 'vacation', 'everamaze', 'place', '.'],
 [],
 ['extremely', 'warm', 'welcoming', '.'],
 [],
```

Figure 4 – Clean text

3.3 Data Transformation

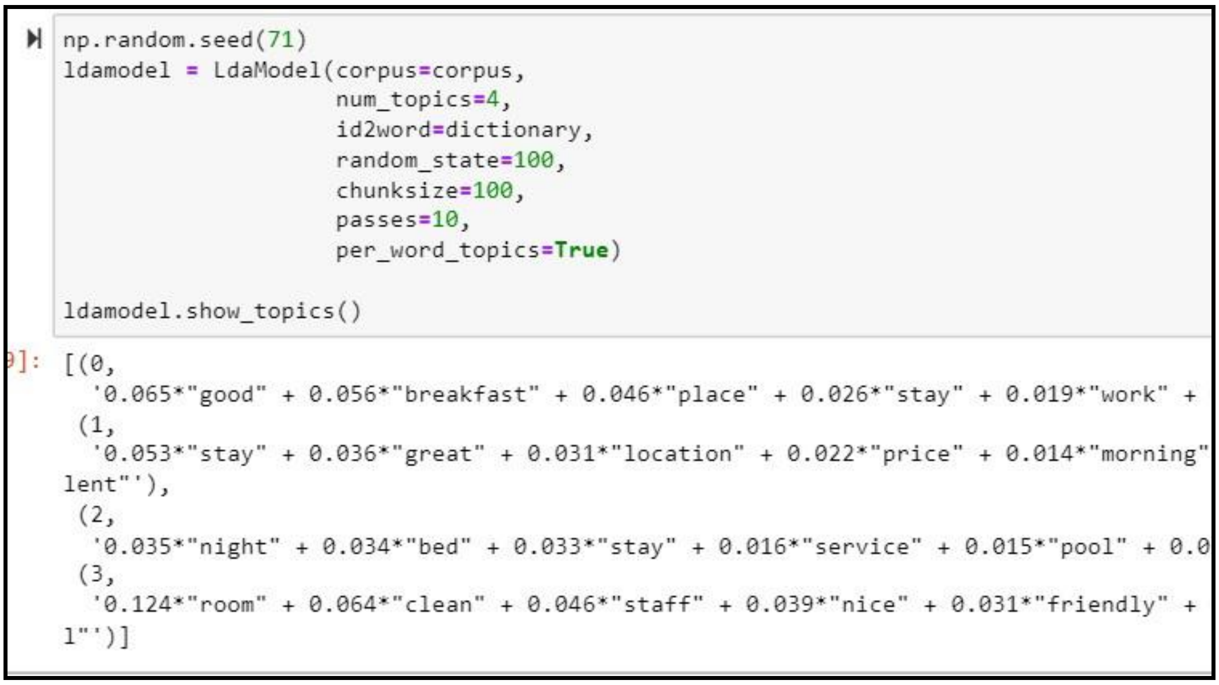
The goodtext list as shown above, is used to prepare a dictionary and a corpus which will be used by the LDA model to topic modelling. The unwanted empty lists and full stops seen in Figure 4 are removed from the dictionary to have more accuracy in the task of topic modelling.

3.4 Data Mining

Data mining is performed on the textual data using the LDA model of topic modelling. The topic modelling facilitated the extraction of aspect from the review text. For each aspect a text file is created and reviews matching any aspect is placed in its respective text file. This text categorization task helped division of aspect wise reviews in an easier manner.

3.4.1. Topic Modelling – Baseline LDA model

A baseline LDA model with number of topics 4 is run at first. This model helped discover four aspects – room, food, location, staff. The topics are still unclear and not easy to identify. The calculated coherence score is very low – 0.2983. Following is the code for first LDA model in Figure 5, along with the model visualization in Figure 6.



```
np.random.seed(71)
ldamodel = LdaModel(corpus=corpus,
                    num_topics=4,
                    id2word=dictionary,
                    random_state=100,
                    chunksize=100,
                    passes=10,
                    per_word_topics=True)

ldamodel.show_topics()
```

```
]: [(0,
    '0.065*"good" + 0.056*"breakfast" + 0.046*"place" + 0.026*"stay" + 0.019*"work" +
   (1,
    '0.053*"stay" + 0.036*"great" + 0.031*"location" + 0.022*"price" + 0.014*"morning"
   lent"'),
   (2,
    '0.035*"night" + 0.034*"bed" + 0.033*"stay" + 0.016*"service" + 0.015*"pool" + 0.0
   (3,
    '0.124*"room" + 0.064*"clean" + 0.046*"staff" + 0.039*"nice" + 0.031*"friendly" +
   l"')]
```

Figure 5 _ Baseline LDA model

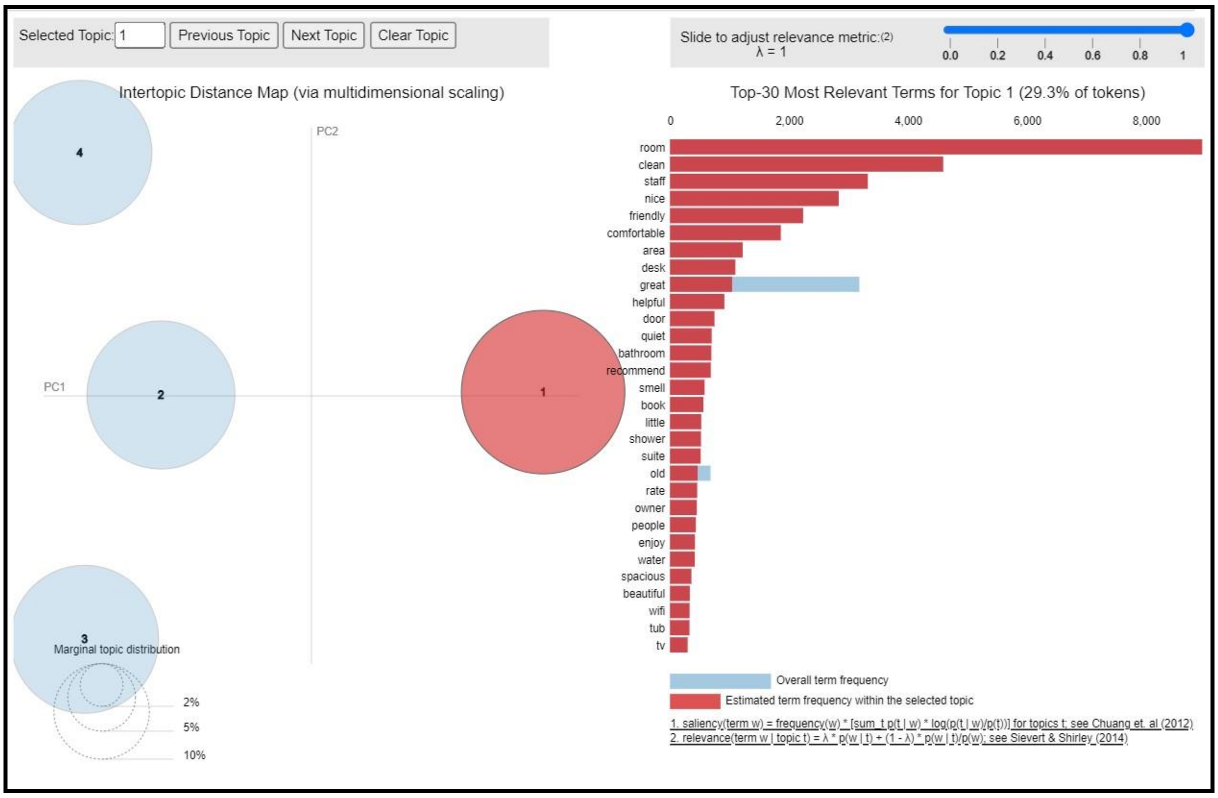
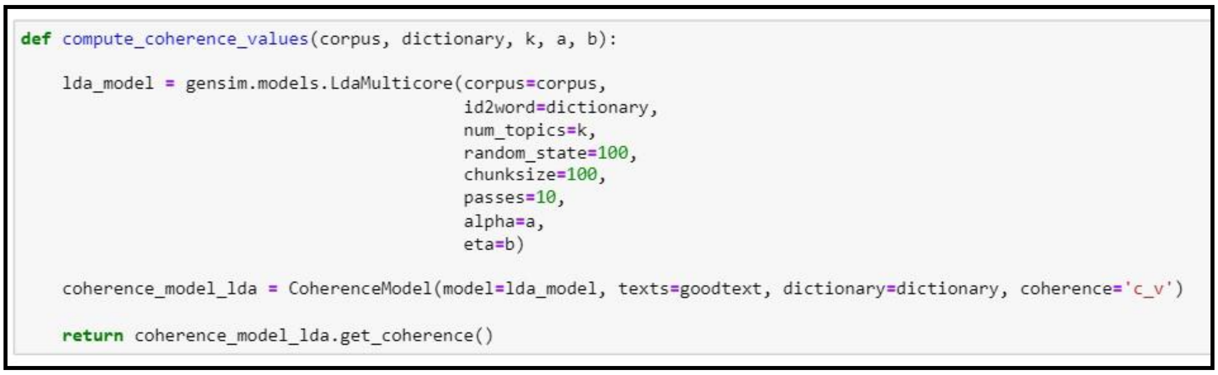


Figure 6 _ Baseline LDA model visualozation

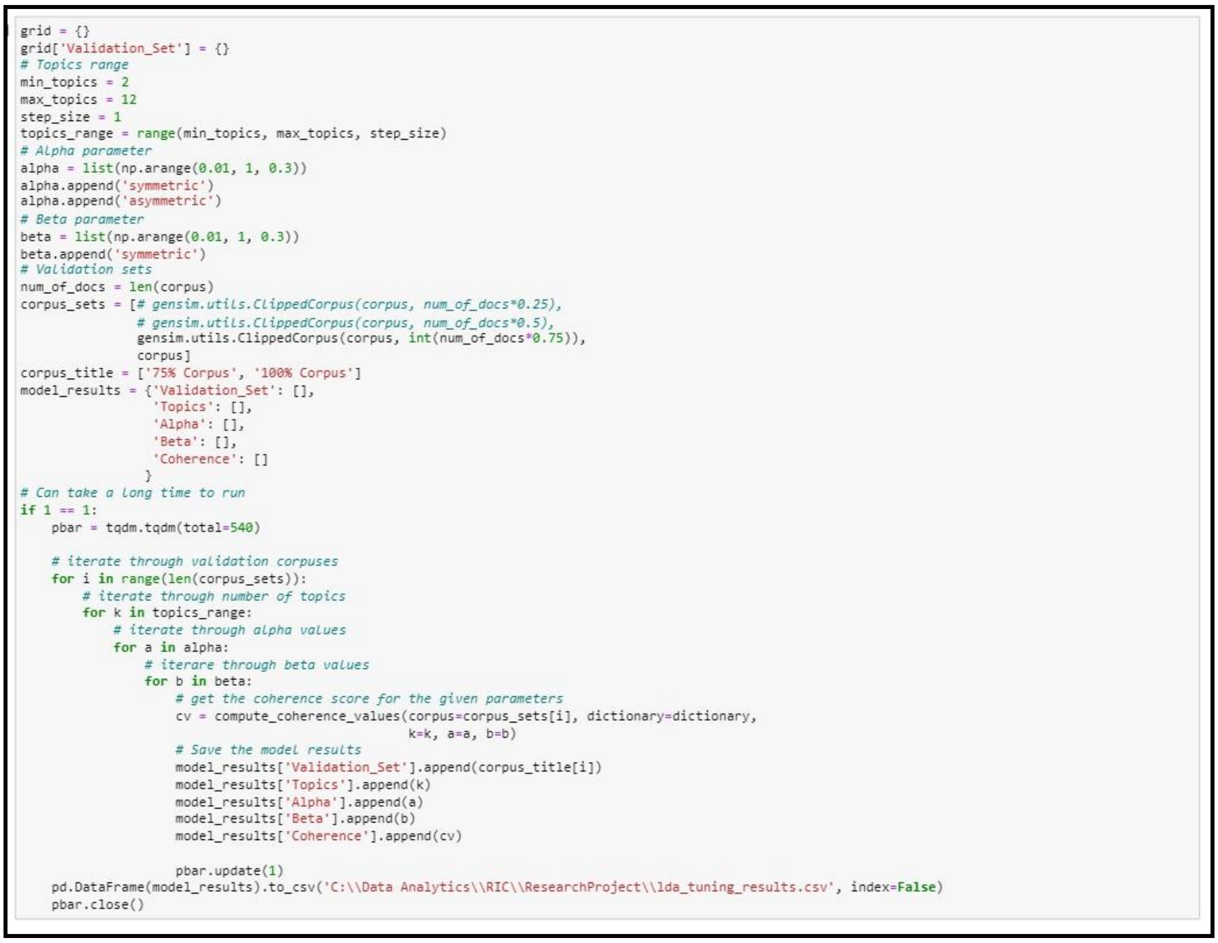
3.4.2. Hyperparameter Tuning

The hyperparameters of LDA are num_topics, alpha and eta. To find the optimal value for these hyperparameters the model is run on a with a range of values for these parameters and coherence score is calculated. The model is run changing the value of one parameter at a time and keeping the rest of the parameters constant. Following Figure 7 and Figure 8 show the code used for same.



```
def compute_coherence_values(corpus, dictionary, k, a, b):

    lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                           id2word=dictionary,
                                           num_topics=k,
                                           random_state=100,
                                           chunksize=100,
                                           passes=10,
                                           alpha=a,
                                           eta=b)

    coherence_model_lda = CoherenceModel(model=lda_model, texts=goodtext, dictionary=dictionary, coherence='c_v')

    return coherence_model_lda.get_coherence()
```

Figure 7 _ Defination to compute coherence



```
grid = {}
grid['Validation_Set'] = {}
# Topics range
min_topics = 2
max_topics = 12
step_size = 1
topics_range = range(min_topics, max_topics, step_size)
# Alpha parameter
alpha = list(np.arange(0.01, 1, 0.3))
alpha.append('symmetric')
alpha.append('asymmetric')
# Beta parameter
beta = list(np.arange(0.01, 1, 0.3))
beta.append('symmetric')
# Validation sets
num_of_docs = len(corpus)
corpus_sets = [# gensim.utils.ClippedCorpus(corpus, num_of_docs*0.25),
               # gensim.utils.ClippedCorpus(corpus, num_of_docs*0.5),
               gensim.utils.ClippedCorpus(corpus, int(num_of_docs*0.75)),
               corpus]
corpus_title = ['75% Corpus', '100% Corpus']
model_results = {'Validation_Set': [],
                 'Topics': [],
                 'Alpha': [],
                 'Beta': [],
                 'Coherence': []
                }
# Can take a long time to run
if 1 == 1:
    pbar = tqdm.tqdm(total=540)

    # iterate through validation corpuses
    for i in range(len(corpus_sets)):
        # iterate through number of topics
        for k in topics_range:
            # iterate through alpha values
            for a in alpha:
                # iterare through beta values
                for b in beta:
                    # get the coherence score for the given parameters
                    cv = compute_coherence_values(corpus=corpus_sets[i], dictionary=dictionary,
                                                  k=k, a=a, b=b)
                    # Save the model results
                    model_results['Validation_Set'].append(corpus_title[i])
                    model_results['Topics'].append(k)
                    model_results['Alpha'].append(a)
                    model_results['Beta'].append(b)
                    model_results['Coherence'].append(cv)

                    pbar.update(1)
    pd.DataFrame(model_results).to_csv('C:\\Data Analytics\\RIC\\ResearchProject\\lda_tuning_results.csv', index=False)
    pbar.close()
```

Figure 8 – Code to set range of values to hyperparameters

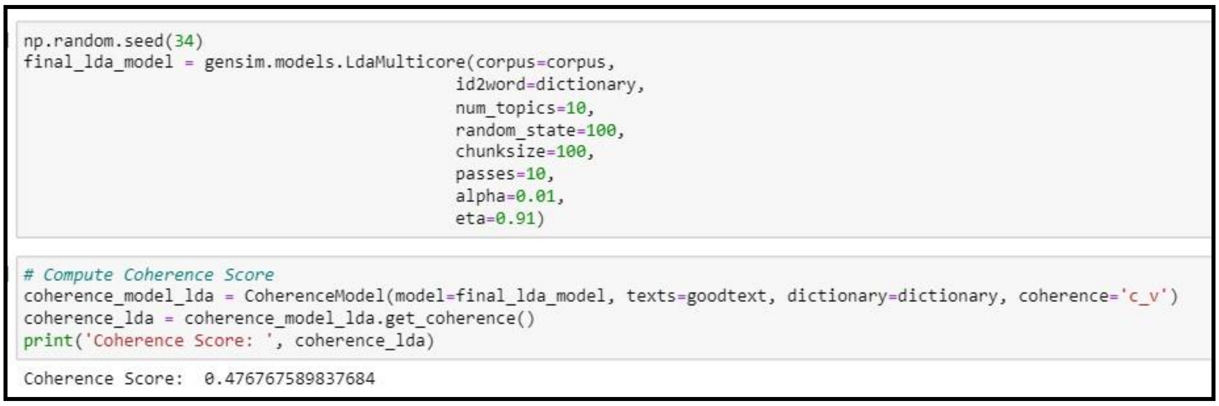
The result of the hyper tuning is saved in a csv file named lda_tuning_results.csv. This result is analysed using filters in Excel. Following is the analysed result in Table 3. The highlighted row displays the most optimal values for the hyperparameters.

Table 3 – Hyperparameter Tuning Results

Validation_ et	Topics	Alpha	Beta	Coherence
100% Corpus	8	asymmetric	0.91	0.503231258
100% Corpus	9	asymmetric	0.91	0.500581261
100% Corpus	9	asymmetric	0.61	0.494660459
100% Corpus	7	asymmetric	0.61	0.491913908
100% Corpus	10	0.01	0.91	0.472133629
100% Corpus	7	0.01	0.91	0.471055795
100% Corpus	8	symmetric	0.91	0.462420889
100% Corpus	5	0.01	0.91	0.460108454
100% Corpus	8	0.01	0.91	0.458805676

3.4.3. LDA model with tuned hyperparameters

The LDA model which was run with tuned hyperparameters yields a much better result. It also helps learn two more hidden aspects in the dataset – Amenities and Value for Money.
The coherence score of this model is 0.4767 which shows 62% increase in the performance. The code for this model and coherence calculation is as follows in Figure 9.



```
np.random.seed(34)
final_lda_model = gensim.models.LdaMulticore(corpus=corpus,
                                   id2word=dictionary,
                                   num_topics=10,
                                   random_state=100,
                                   chunksize=100,
                                   passes=10,
                                   alpha=0.01,
                                   eta=0.91)
```

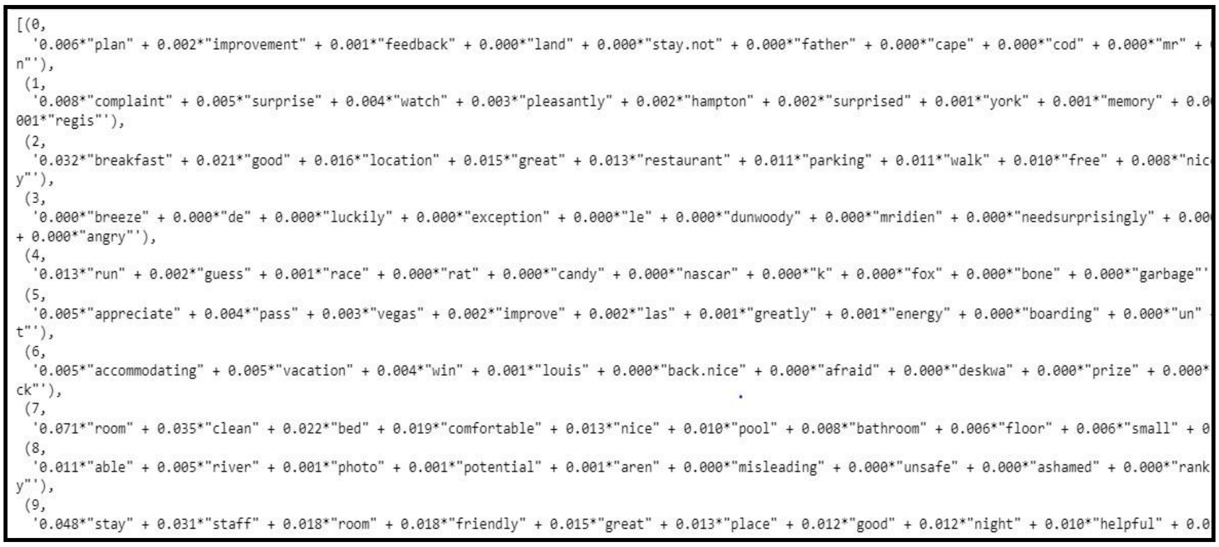
```
# Compute Coherence Score
coherence_model_lda = CoherenceModel(model=final_lda_model, texts=goodtext, dictionary=dictionary, coherence='c_v')
coherence_lda = coherence_model_lda.get_coherence()
print('Coherence Score: ', coherence_lda)
```

```
Coherence Score:  0.476767589837684
```

Figure 9 _ Hypertuned LDA model

3.4.4. Aspect Extraction and Text Categorization

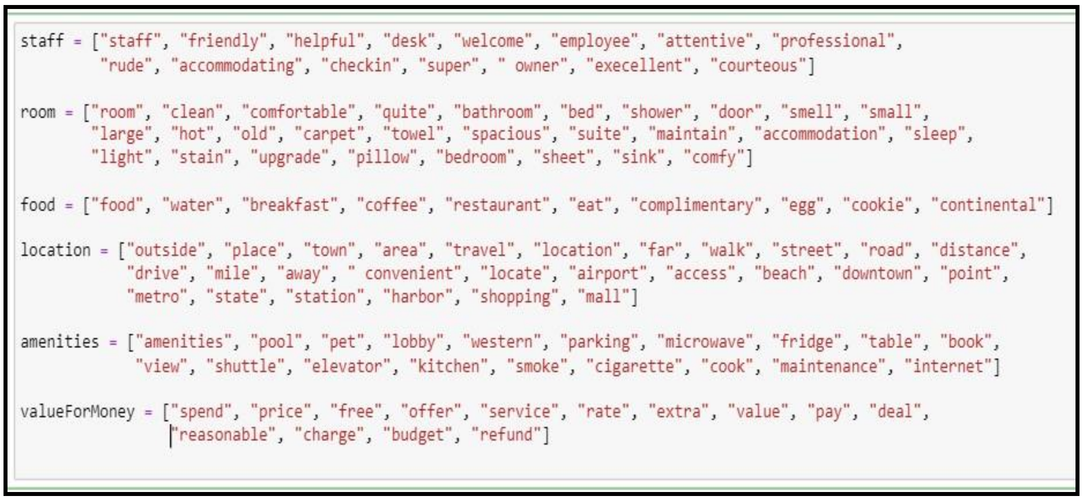
Depending on the topics by the hyper tuned LDA model six aspects are extracted. The aspects are extracted by analysing the topics displayed below in Figure 10.



```
[(0,
  '0.006*"plan" + 0.002*"improvement" + 0.001*"feedback" + 0.000*"land" + 0.000*"stay.not" + 0.000*"father" + 0.000*"cape" + 0.000*"cod" + 0.000*"mr" +
n"'),
 (1,
  '0.008*"complaint" + 0.005*"surprise" + 0.004*"watch" + 0.003*"pleasantly" + 0.002*"hampton" + 0.002*"surprised" + 0.001*"york" + 0.001*"memory" + 0.0
001*"regis"'),
 (2,
  '0.032*"breakfast" + 0.021*"good" + 0.016*"location" + 0.015*"great" + 0.013*"restaurant" + 0.011*"parking" + 0.011*"walk" + 0.010*"free" + 0.008*"nic
y"'),
 (3,
  '0.000*"breeze" + 0.000*"de" + 0.000*"luckily" + 0.000*"exception" + 0.000*"le" + 0.000*"dunwoody" + 0.000*"mridien" + 0.000*"needsurprisingly" + 0.00
+ 0.000*"angry"'),
 (4,
  '0.013*"run" + 0.002*"guess" + 0.001*"race" + 0.000*"rat" + 0.000*"candy" + 0.000*"nascar" + 0.000*"k" + 0.000*"fox" + 0.000*"bone" + 0.000*"garbage"'
 (5,
  '0.005*"appreciate" + 0.004*"pass" + 0.003*"vegas" + 0.002*"improve" + 0.002*"las" + 0.001*"greatly" + 0.001*"energy" + 0.000*"boarding" + 0.000*"un"
t"'),
 (6,
  '0.005*"accommodating" + 0.005*"vacation" + 0.004*"win" + 0.001*"louis" + 0.000*"back.nice" + 0.000*"afraid" + 0.000*"deskwa" + 0.000*"prize" + 0.000*
ck"'),
 (7,
  '0.071*"room" + 0.035*"clean" + 0.022*"bed" + 0.019*"comfortable" + 0.013*"nice" + 0.010*"pool" + 0.008*"bathroom" + 0.006*"floor" + 0.006*"small" + 0
 (8,
  '0.011*"able" + 0.005*"river" + 0.001*"photo" + 0.001*"potential" + 0.001*"aren" + 0.000*"misleading" + 0.000*"unsafe" + 0.000*"ashamed" + 0.000*"rank
y"'),
 (9,
  '0.048*"stay" + 0.031*"staff" + 0.018*"room" + 0.018*"friendly" + 0.015*"great" + 0.013*"place" + 0.012*"good" + 0.012*"night" + 0.010*"helpful" + 0.0
```

Figure 10 _ LDA Topics

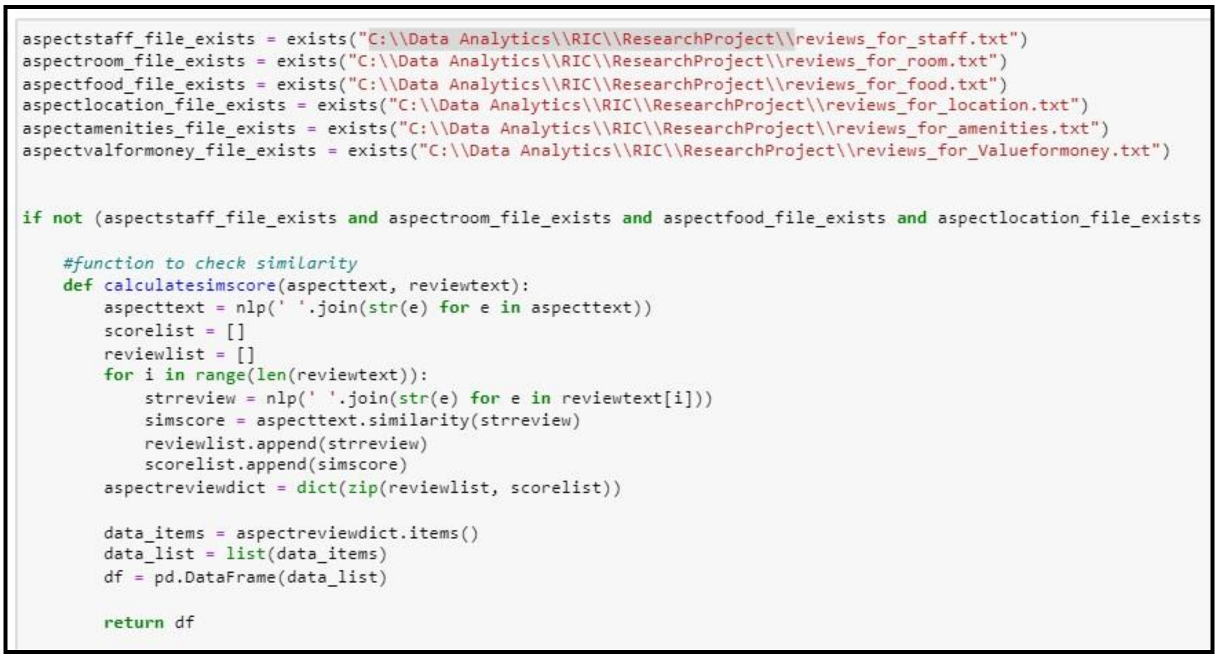
Aspects staff, room, location, food, amenities and value for money are identified along with the related frequently occurring words using the LDA topic modelling. These words are then appended in a list for each aspect as shown below in Figure 11.



```python
staff = ["staff", "friendly", "helpful", "desk", "welcome", "employee", "attentive", "professional",
         "rude", "accommodating", "checkin", "super", " owner", "execellent", "courteous"]

room = ["room", "clean", "comfortable", "quite", "bathroom", "bed", "shower", "door", "smell", "small",
        "large", "hot", "old", "carpet", "towel", "spacious", "suite", "maintain", "accommodation", "sleep",
        "light", "stain", "upgrade", "pillow", "bedroom", "sheet", "sink", "comfy"]

food = ["food", "water", "breakfast", "coffee", "restaurant", "eat", "complimentary", "egg", "cookie", "continental"]

location = ["outside", "place", "town", "area", "travel", "location", "far", "walk", "street", "road", "distance",
            "drive", "mile", "away", " convenient", "locate", "airport", "access", "beach", "downtown", "point",
            "metro", "state", "station", "harbor", "shopping", "mall"]

amenities = ["amenities", "pool", "pet", "lobby", "western", "parking", "microwave", "fridge", "table", "book",
             "view", "shuttle", "elevator", "kitchen", "smoke", "cigarette", "cook", "maintenance", "internet"]

valueForMoney = ["spend", "price", "free", "offer", "service", "rate", "extra", "value", "pay", "deal",
                 "reasonable", "charge", "budget", "refund"]
```

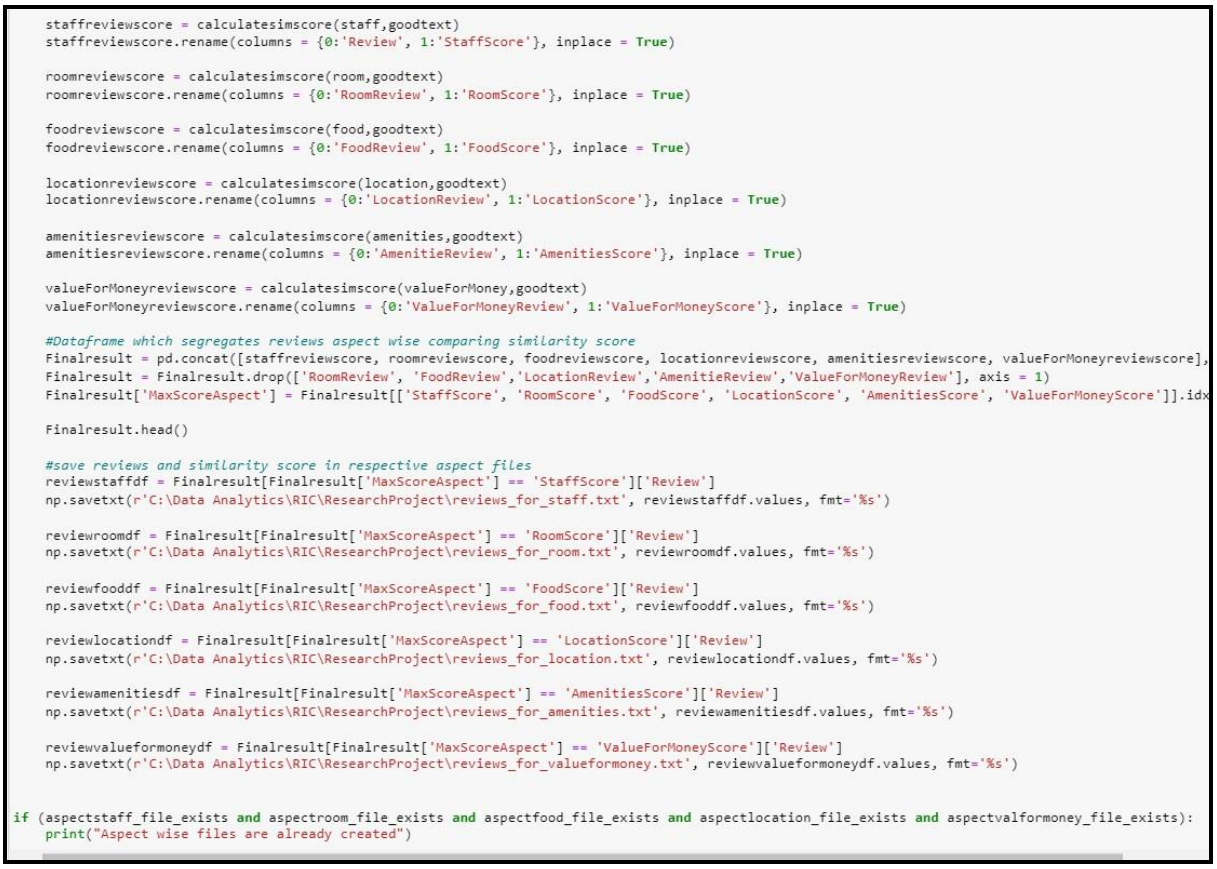
Figure 11 – Aspects and related words in a list

Once the aspect list is ready it is matched one by one with each review item in the goodtext list and a similarity score is calculated. To calculate this similarity score the similarity method from the spaCy library is used. The review which has the highest similarity score with the a particular aspect is stored in its respective aspect txt file. The code for same in below in Figure 12 and Figure 13.



```python
aspectstaff_file_exists = exists("C:\\Data Analytics\\RIC\\ResearchProject\\reviews_for_staff.txt")
aspectroom_file_exists = exists("C:\\Data Analytics\\RIC\\ResearchProject\\reviews_for_room.txt")
aspectfood_file_exists = exists("C:\\Data Analytics\\RIC\\ResearchProject\\reviews_for_food.txt")
aspectlocation_file_exists = exists("C:\\Data Analytics\\RIC\\ResearchProject\\reviews_for_location.txt")
aspectamenities_file_exists = exists("C:\\Data Analytics\\RIC\\ResearchProject\\reviews_for_amenities.txt")
aspectvalformoney_file_exists = exists("C:\\Data Analytics\\RIC\\ResearchProject\\reviews_for_Valueformoney.txt")


if not (aspectstaff_file_exists and aspectroom_file_exists and aspectfood_file_exists and aspectlocation_file_exists

    #function to check similarity
    def calculatesimscore(aspecttext, reviewtext):
        aspecttext = nlp(' '.join(str(e) for e in aspecttext))
        scorelist = []
        reviewlist = []
        for i in range(len(reviewtext)):
            strreview = nlp(' '.join(str(e) for e in reviewtext[i]))
            simscore = aspecttext.similarity(strreview)
            reviewlist.append(strreview)
            scorelist.append(simscore)
        aspectreviewdict = dict(zip(reviewlist, scorelist))

        data_items = aspectreviewdict.items()
        data_list = list(data_items)
        df = pd.DataFrame(data_list)

        return df
```

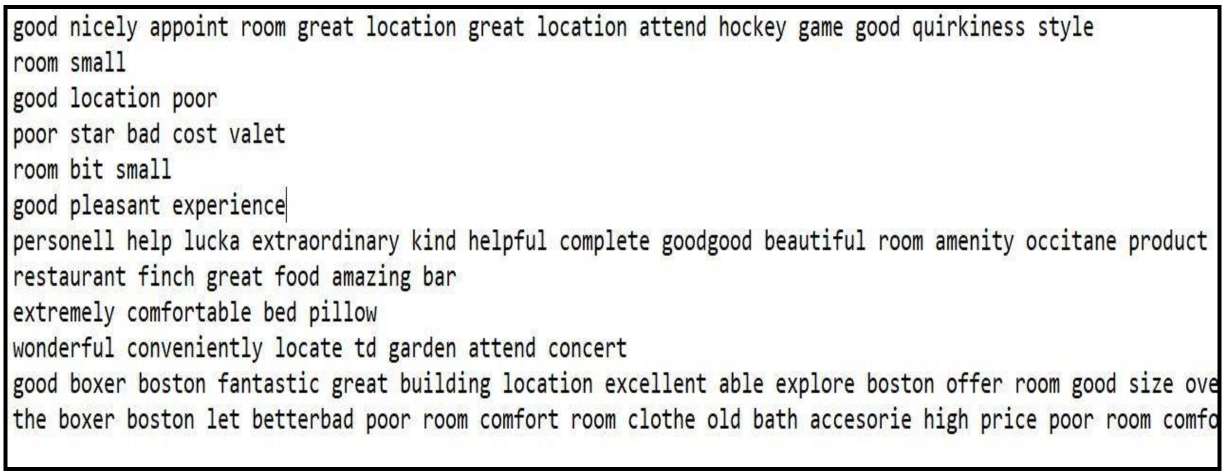
Figure 12 _ Function to calculate similarity



```python
    staffreviewscore = calculatesimscore(staff,goodtext)
    staffreviewscore.rename(columns = {0:'Review', 1:'StaffScore'}, inplace = True)

    roomreviewscore = calculatesimscore(room,goodtext)
    roomreviewscore.rename(columns = {0:'RoomReview', 1:'RoomScore'}, inplace = True)

    foodreviewscore = calculatesimscore(food,goodtext)
    foodreviewscore.rename(columns = {0:'FoodReview', 1:'FoodScore'}, inplace = True)

    locationreviewscore = calculatesimscore(location,goodtext)
    locationreviewscore.rename(columns = {0:'LocationReview', 1:'LocationScore'}, inplace = True)

    amenitiesreviewscore = calculatesimscore(amenities,goodtext)
    amenitiesreviewscore.rename(columns = {0:'AmenitieReview', 1:'AmenitiesScore'}, inplace = True)

    valueForMoneyreviewscore = calculatesimscore(valueForMoney,goodtext)
    valueForMoneyreviewscore.rename(columns = {0:'ValueForMoneyReview', 1:'ValueForMoneyScore'}, inplace = True)

    #Dataframe which segregates reviews aspect wise comparing similarity score
    Finalresult = pd.concat([staffreviewscore, roomreviewscore, foodreviewscore, locationreviewscore, amenitiesreviewscore, valueForMoneyreviewscore],
    Finalresult = Finalresult.drop(['RoomReview', 'FoodReview','LocationReview','AmenitieReview','ValueForMoneyReview'], axis = 1)
    Finalresult['MaxScoreAspect'] = Finalresult[['StaffScore', 'RoomScore', 'FoodScore', 'LocationScore', 'AmenitiesScore', 'ValueForMoneyScore']].idx

    Finalresult.head()

    #save reviews and similarity score in respective aspect files
    reviewstaffdf = Finalresult[Finalresult['MaxScoreAspect'] == 'StaffScore']['Review']
    np.savetxt(r'C:\Data Analytics\RIC\ResearchProject\reviews_for_staff.txt', reviewstaffdf.values, fmt='%s')

    reviewroomdf = Finalresult[Finalresult['MaxScoreAspect'] == 'RoomScore']['Review']
    np.savetxt(r'C:\Data Analytics\RIC\ResearchProject\reviews_for_room.txt', reviewroomdf.values, fmt='%s')

    reviewfooddf = Finalresult[Finalresult['MaxScoreAspect'] == 'FoodScore']['Review']
    np.savetxt(r'C:\Data Analytics\RIC\ResearchProject\reviews_for_food.txt', reviewfooddf.values, fmt='%s')

    reviewlocationdf = Finalresult[Finalresult['MaxScoreAspect'] == 'LocationScore']['Review']
    np.savetxt(r'C:\Data Analytics\RIC\ResearchProject\reviews_for_location.txt', reviewlocationdf.values, fmt='%s')

    reviewamenitiesdf = Finalresult[Finalresult['MaxScoreAspect'] == 'AmenitiesScore']['Review']
    np.savetxt(r'C:\Data Analytics\RIC\ResearchProject\reviews_for_amenities.txt', reviewamenitiesdf.values, fmt='%s')

    reviewvalueformoneydf = Finalresult[Finalresult['MaxScoreAspect'] == 'ValueForMoneyScore']['Review']
    np.savetxt(r'C:\Data Analytics\RIC\ResearchProject\reviews_for_valueformoney.txt', reviewvalueformoneydf.values, fmt='%s')


if (aspectstaff_file_exists and aspectroom_file_exists and aspectfood_file_exists and aspectlocation_file_exists and aspectvalformoney_file_exists):
    print("Aspect wise files are already created")
```

Figure 13 – File creation for each aspect and its respective reviews based on similarity score

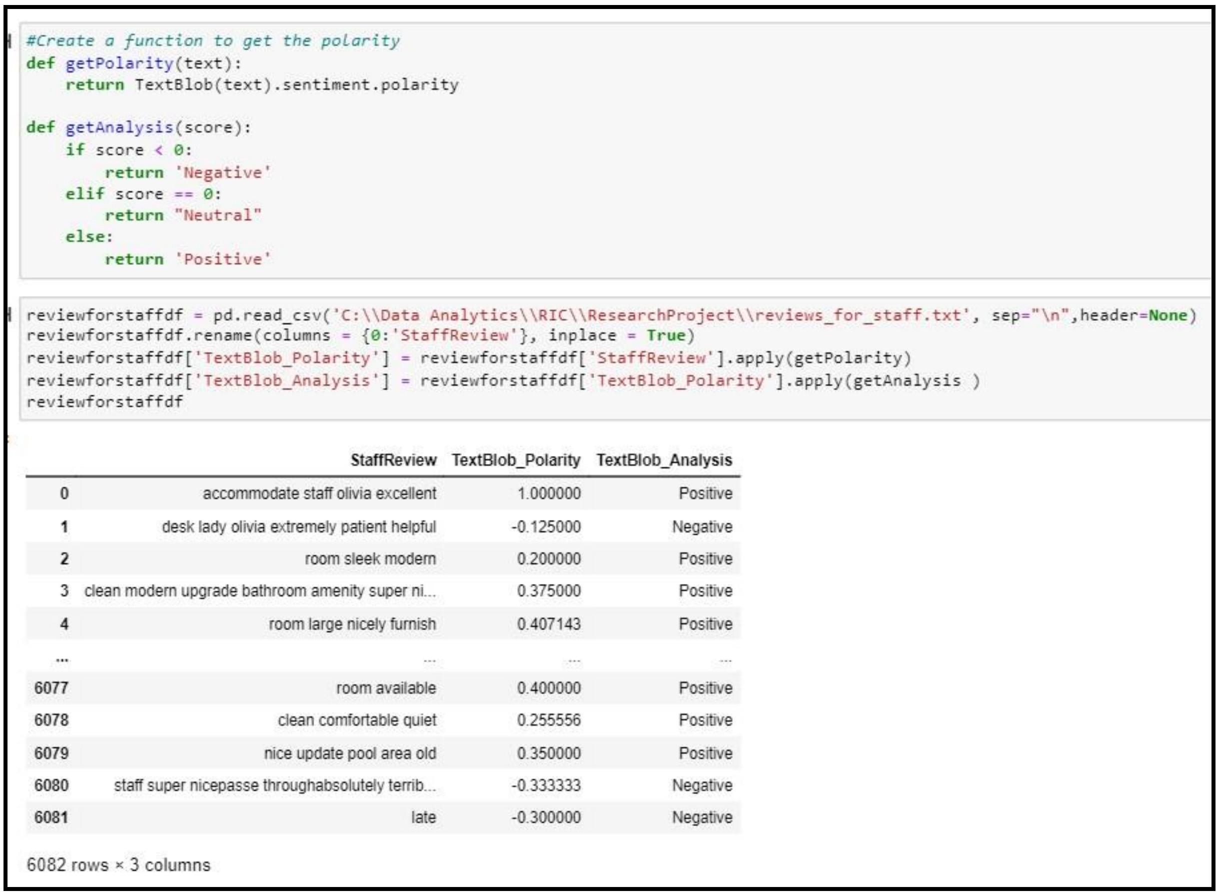
The snapshot of the room.txt file created is below in Figure



```
good nicely appoint room great location great location attend hockey game good quirkiness style
room small
good location poor
poor star bad cost valet
room bit small
good pleasant experience
personell help lucka extraordinary kind helpful complete goodgood beautiful room amenity occitane product
restaurant finch great food amazing bar
extremely comfortable bed pillow
wonderful conveniently locate td garden attend concert
good boxer boston fantastic great building location excellent able explore boston offer room good size ove
the boxer boston let betterbad poor room comfort room clothe old bath accesorie high price poor room comfo
```

3.5 Sentiment Analysis

The TextBlob library is used to calculate the polarity of each review file wise and following is its code in Figure 14. Each aspect file makes use of the custom function - getPolarity() for sentiment analysis and this is further visualized for each file. For staff as aspect sentiment analysis visualization see Figure 14.



```python
#Create a function to get the polarity
def getPolarity(text):
    return TextBlob(text).sentiment.polarity

def getAnalysis(score):
    if score < 0:
        return 'Negative'
    elif score == 0:
        return "Neutral"
    else:
        return 'Positive'
```

```python
reviewforstaffdf = pd.read_csv('C:\\Data Analytics\\RIC\\ResearchProject\\reviews_for_staff.txt', sep="\n",header=None)
reviewforstaffdf.rename(columns = {0:'StaffReview'}, inplace = True)
reviewforstaffdf['TextBlob_Polarity'] = reviewforstaffdf['StaffReview'].apply(getPolarity)
reviewforstaffdf['TextBlob_Analysis'] = reviewforstaffdf['TextBlob_Polarity'].apply(getAnalysis )
reviewforstaffdf
```

	StaffReview	TextBlob_Polarity	TextBlob_Analysis
0	accommodate staff olivia excellent	1.000000	Positive
1	desk lady olivia extremely patient helpful	-0.125000	Negative
2	room sleek modern	0.200000	Positive
3	clean modern upgrade bathroom amenity super ni...	0.375000	Positive
4	room large nicely furnish	0.407143	Positive
...	...	...	...
6077	room available	0.400000	Positive
6078	clean comfortable quiet	0.255556	Positive
6079	nice update pool area old	0.350000	Positive
6080	staff super nicepasse throughabsolutely terrib...	-0.333333	Negative
6081	late	-0.300000	Negative

6082 rows × 3 columns

Figure 14 – Sentiment analysis

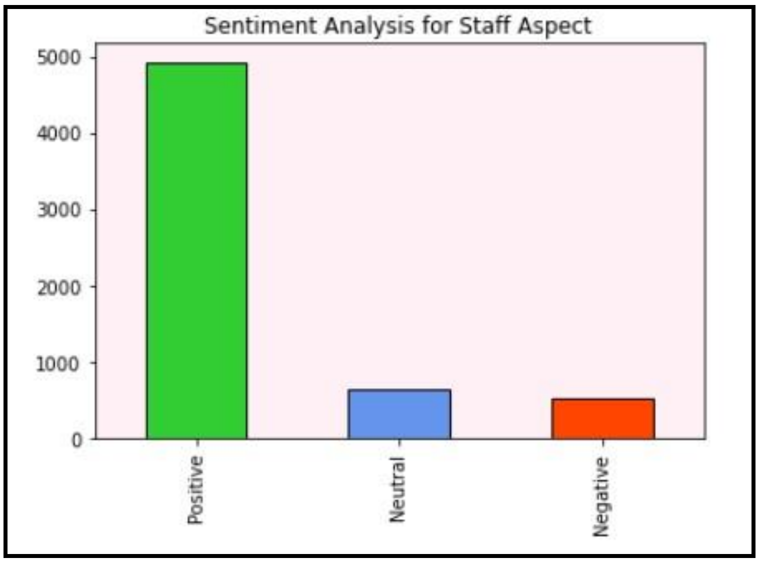
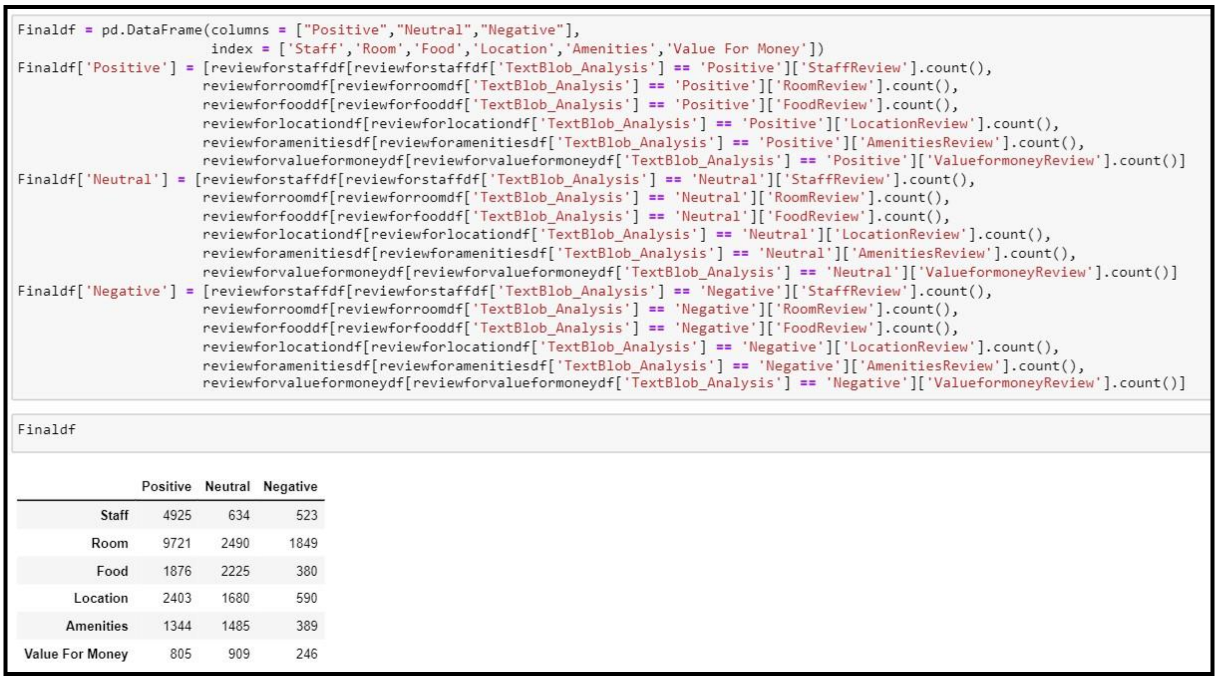


Figure 15 – Staff aspect sentiment analysis

The result is aggregated into one single data frame as shown in Figure 16. Also a visualization for the aggregated results is seen in Figure 17.



```
Finaldf = pd.DataFrame(columns = ["Positive","Neutral","Negative"],
                       index = ['Staff','Room','Food','Location','Amenities','Value For Money'])
Finaldf['Positive'] = [reviewforstaffdf[reviewforstaffdf['TextBlob_Analysis'] == 'Positive']['StaffReview'].count(),
                      reviewforroomdf[reviewforroomdf['TextBlob_Analysis'] == 'Positive']['RoomReview'].count(),
                      reviewforfooddf[reviewforfooddf['TextBlob_Analysis'] == 'Positive']['FoodReview'].count(),
                      reviewforlocationdf[reviewforlocationdf['TextBlob_Analysis'] == 'Positive']['LocationReview'].count(),
                      reviewforamenitiesdf[reviewforamenitiesdf['TextBlob_Analysis'] == 'Positive']['AmenitiesReview'].count(),
                      reviewforvalueformoneydf[reviewforvalueformoneydf['TextBlob_Analysis'] == 'Positive']['ValueformoneyReview'].count()]
Finaldf['Neutral'] = [reviewforstaffdf[reviewforstaffdf['TextBlob_Analysis'] == 'Neutral']['StaffReview'].count(),
                      reviewforroomdf[reviewforroomdf['TextBlob_Analysis'] == 'Neutral']['RoomReview'].count(),
                      reviewforfooddf[reviewforfooddf['TextBlob_Analysis'] == 'Neutral']['FoodReview'].count(),
                      reviewforlocationdf[reviewforlocationdf['TextBlob_Analysis'] == 'Neutral']['LocationReview'].count(),
                      reviewforamenitiesdf[reviewforamenitiesdf['TextBlob_Analysis'] == 'Neutral']['AmenitiesReview'].count(),
                      reviewforvalueformoneydf[reviewforvalueformoneydf['TextBlob_Analysis'] == 'Neutral']['ValueformoneyReview'].count()]
Finaldf['Negative'] = [reviewforstaffdf[reviewforstaffdf['TextBlob_Analysis'] == 'Negative']['StaffReview'].count(),
                      reviewforroomdf[reviewforroomdf['TextBlob_Analysis'] == 'Negative']['RoomReview'].count(),
                      reviewforfooddf[reviewforfooddf['TextBlob_Analysis'] == 'Negative']['FoodReview'].count(),
                      reviewforlocationdf[reviewforlocationdf['TextBlob_Analysis'] == 'Negative']['LocationReview'].count(),
                      reviewforamenitiesdf[reviewforamenitiesdf['TextBlob_Analysis'] == 'Negative']['AmenitiesReview'].count(),
                      reviewforvalueformoneydf[reviewforvalueformoneydf['TextBlob_Analysis'] == 'Negative']['ValueformoneyReview'].count()]
```

```
Finaldf
```

	Positive	Neutral	Negative
Staff	4925	634	523
Room	9721	2490	1849
Food	1876	2225	380
Location	2403	1680	590
Amenities	1344	1485	389
Value For Money	805	909	246

Figure 16 _ Result aggregation of sentiment analysis of all aspects

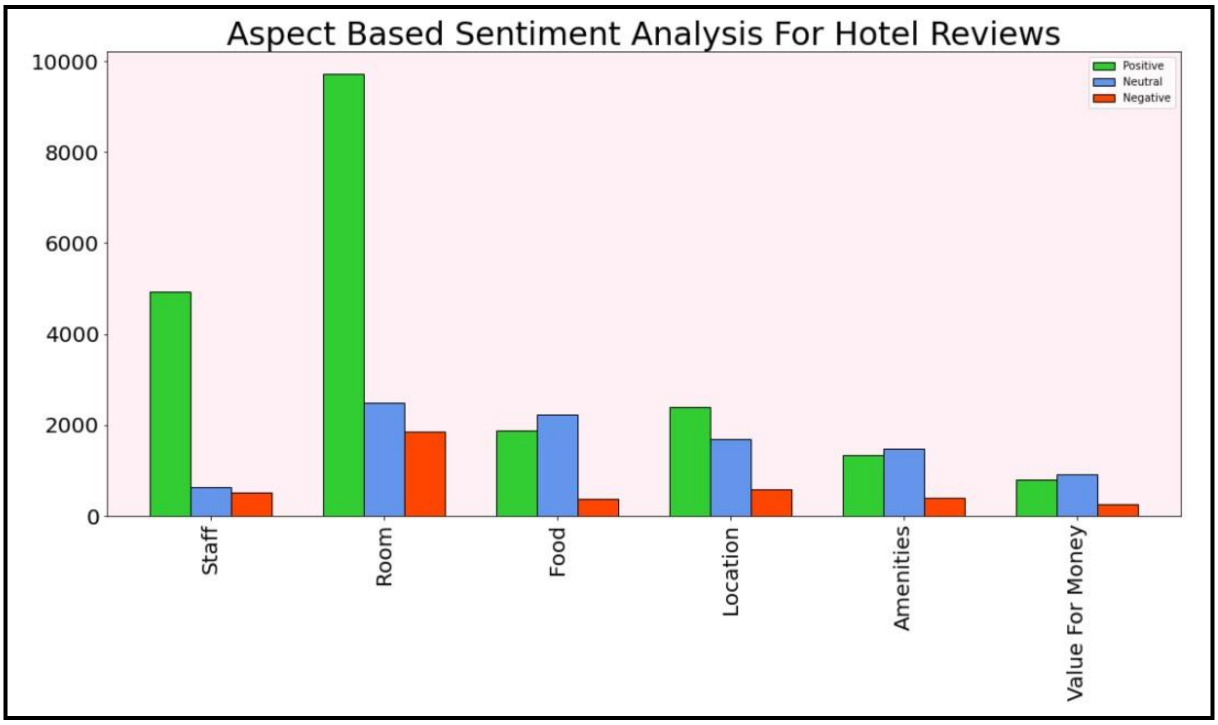


Figure 17 _ Aspect wise Sentiment Analysis