Capstone project report on

Recommendation System and topic modelling of hotel reviews

Submitted to

Praxis Business School, Kolkata

Post Graduate Program

in

Data Science

By

Hitesh Tiwari (C22009)
Saswata Pradhan (C22020)
Daksh Bhandari (C22007)
Sumit Arora (C22026)
Madhurima Pande (C22015)
Satadru Mukherjee (C22021)

Under guidance of

Prof. Dr. Subhasis Dasgupta

Department of Data Science

Year 2022-2022

Acknowledgement

We are highly grateful to Prof. Dr. Subhasis Dasgupta, Head of Machine Learning & Analytics, for his expert guidance and continuous encouragement throughout the whole project to reach at a significant standpoint. We would like to convey our thanks to Prof. Charanpreet Singh, Prof. Dr. Prithwis Mukherjee, Prof. Dr. Sourav Saha, Founder, Director and Academic Dean of this institute and all other professors of Data Science Department of this institute who taught us each and every course of this program.

Content

Abstract

For any hotel organization, review analysis and recommendation of the right hotels is one of the important parts to understand the hotel's performance and to recommend the user same type of hotel as per their requirements. The reviewer may write review about quality of food, rooms and other features. They can give ratings for each of the attributes available. The ratings can also be written in a positive, negative or neutral way. Understanding the topics of the review and the rating of the overall and attributes helps to identify the hotels conditions. If it's possible to understand, then the hotel organization can focus on specific features of their hotels which are not being liked by the customers. So, in this project we have tried to achieve to build the recommendation engine and so that it can help any user to find proper hotel.

Introduction:

Until recently, the conventional wisdom was that while AI was better than humans at data-driven decision-making tasks, it was still inferior to humans for cognitive and creative ones. But in the past two years language-based AI has advanced by leaps and bounds, changing common notions of what this technology can do. Natural Language Processing, or NLP for short, is broadly defined as the automatic manipulation of natural language, like speech and text, by software. It combines computational linguistics- rule based modelling of human language- statistical, machine learning and deep learning models. This project Is also an application of NLP.

The purpose of this project is to understand the topics of hotel reviews, ratings of hotels and recommend the hotels to the users based on the ratings also recommend the hotels based on the similar type of reviews. From the extracted topics we can build a special type of recommendation engine which will suggest the hotel based on the topics you search, obviously this will be extended part of the project.

Citation:

Fadhel Aljunid, Mohammed & D H, Manjaiah. (2018). ***An Improved ALS Recommendation Model Based on Apache Spark***: Second International Conference, ICSCS 2018, Kollam, India, April 19–20, 2018, Revised Selected Papers. 10.1007/978-981-13-1936-5_33.

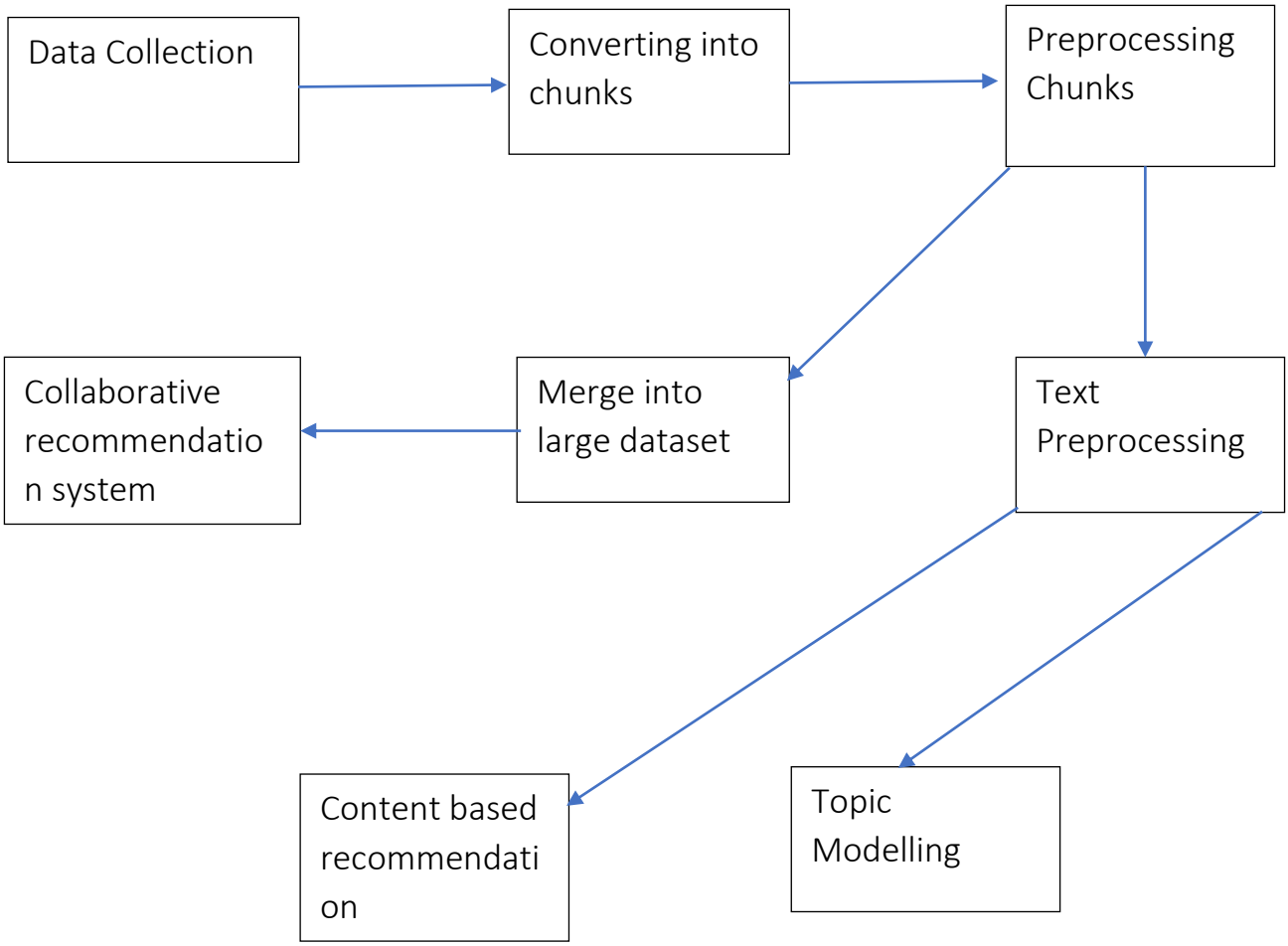
Subhashini Gupta , Grishma Sharma, 2021, ***Topic Modeling in Natural Language Processing***, INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 10, Issue 06 (June 2021),

Lemdani, Roza & Polaillon, Géraldine & Bennacer Seghouani, Nacéra & Bourda, Yolaine. (2011). ***A Semantic Similarity Measure for Recommender Systems***. 183-186. 10.1145/2063518.2063545.

Objective:

The objective of this project is to analyse a large dataset and to extract some key topics from the reviews given by the users and to build recommendation engine to recommend the hotels based on ratings and the reviews.

Workflow:



Explanations of steps:

1. Data Collection:

The data is about hotel reviews that is given by different users. There are almost five crores' records. The data was initially saved in .txt format. The compressed data is thirteen GB and the full size is around fifty GB.

```
import os
size_bytes = os.stat('/content/HotelRec.txt').st_size
size_bytes
```

```
50409998857
```

HotelRec, a very large-scale hotel recommendation dataset, based on TripAdvisor, containing 50 million records. For each review, we collected: the URL of the user's profile and hotel, the date, the overall rating, the summary (i.e., the title of the review), the written text, and the multiple sub-ratings when provided.

The data was initially in zip format. We have extracted the zip file. The data is consisted of seven columns. The column name is hotel_url, author, date, rating, title, text, property_dict.

The property_dict has different sub ratings. The sub ratings are cleanliness rooms, service, value, sleep quality, location, check in / front desk, business service (e.g., internet access).

The rating has been given in between zero to five.

Converting into chunks:

We converted the large file into fifty-one chunks and each chunk consisted of 1000000 records.

The chunks are converted into pickle file and from each pickle file we extracted the columns and formed a CSV file that is readable in pandas.

In Python, one can use the pickle module to serialize objects and keep them to a document. you may then deserialize the serialized document to load them returned when needed

```python
import pandas as pd
import pickle

in_path = "/content/HotelRec.txt" #Path where the large file is
out_path = "/content/drive/MyDrive" #Path to save the pickle files to
chunk_size = 1000000 #size of chunks relies on your available memory
separator = "\t"

reader = pd.read_csv(in_path,sep=separator,chunksize=chunk_size,
                     low_memory=False)


for i, chunk in enumerate(reader):
    out_file = out_path + "/data_{}.pkl".format(i+1)
    with open(out_file, "wb") as f:
        pickle.dump(chunk,f,pickle.HIGHEST_PROTOCOL)
```

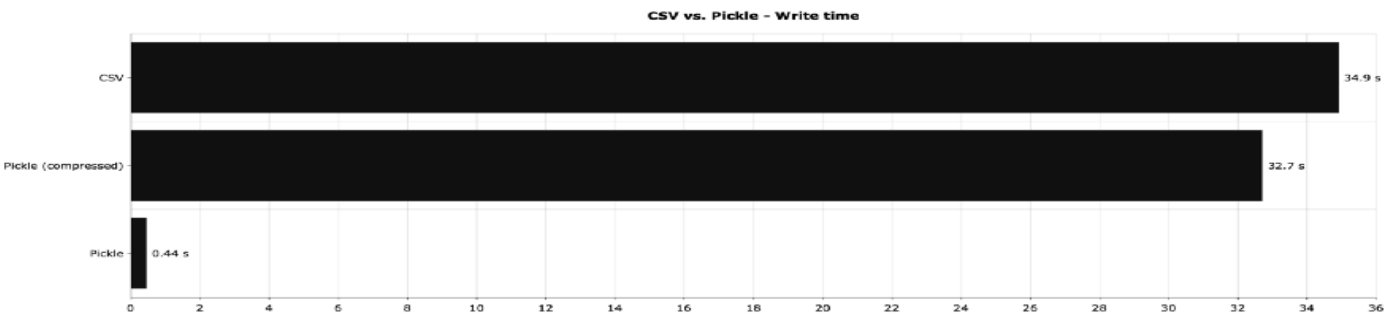
The reason for converting the chunks over CSV:

Pickle has one major advantage over alternative formats — you'll be able to use it to store any Python object. That's correct, you're not restricted to data. one in all the foremost wide used functionalities is saving machine learning models once the coaching is complete. That way, you don't need to retrain the model anytime you run the script.

CSVs give read and edit privileges, as anyone can open them. that might even be thought of as a drawback, for obvious reasons. Moreover, you can't save machine learning models to a CSV file.

The following chart shows the time required to save lots of the Data Frame from the last section locally:

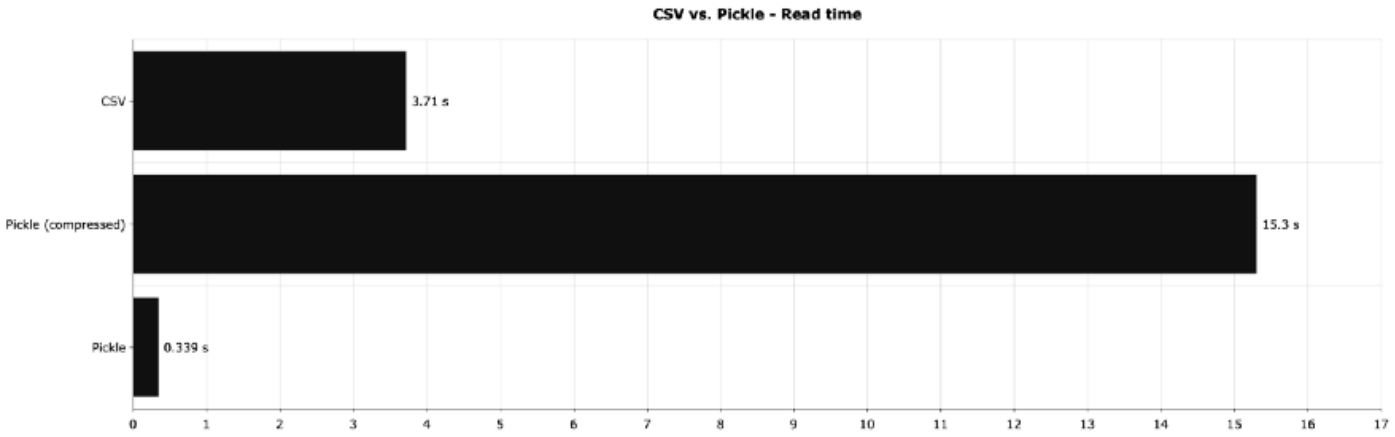
CSV vs. Pickle local save time in seconds (CSV: 34.9; Pickle (compressed): 32.7; Pickle: 0.44).



That's around eighty times speed increase, if you don't care regarding compression.

CSV vs. Pickle scan time in seconds (CSV: 3.71; Pickle (compressed): fifteen.3; Pickle: 0.339)

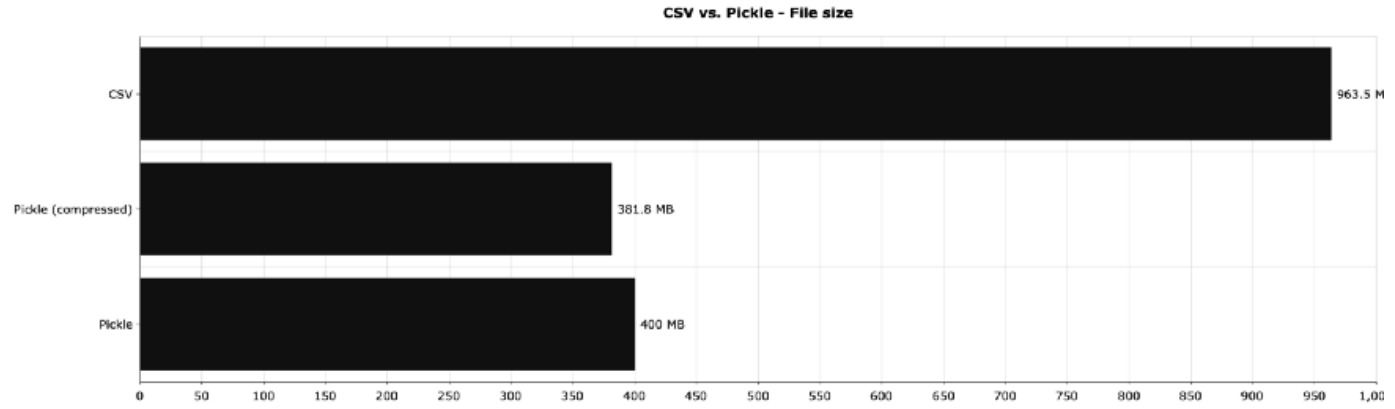
Pickle is around eleven times quicker this point, once not compressed. The compression may be a vast pain purpose once reading and saving files. But, let's see what quantity space will it



save.

CSV vs. Pickle file size in MB (CSV: 963.5; Pickle (compressed): 381.8; Pickle: 400)

The file size decrease in comparison to CSV is significant, however the compression doesn't



save that a lot of space during this case.

Pre-processing each chunk:

Each of the converted pickle file has been pre-processed after converting it into csv file.

For converting pickle in csv file, we have used several libraries.

tqdm is a library in Python which is used for creating Progress Meters or Progress Bars. tqdm got its name from the Arabic name taqaddum which means 'progress'. Implementing tqdm can be done effortlessly in our loops, functions or even Pandas.

Pickle is used for serializing and de-serializing Python object structures, also called marshalling or flattening. Serialization refers to the process of converting an object in memory to a byte stream that can be stored on disk or sent over a network so we are

```
from tqdm import tqdm
import pickle
import pandas as pd
import json
import numpy as np

def __to_dataframe(pkl_file):
    df_dict = {'hotel_url':[],'author':[],'date':[],'rating':[],'title':[],'text':[],'property_dict':[]}
    df = pd.read_pickle(pkl_file)
    for row in tqdm(df.values):
        j = json.loads(row[0])
        df_dict['hotel_url'].append(j['hotel_url'])
        df_dict['author'].append(j['author'])
        df_dict['date'].append(j['date'])
        df_dict['rating'].append(j['rating'])
        df_dict['title'].append(j['title'])
        df_dict['text'].append(j['text'])
        df_dict['property_dict'].append(j['property_dict'])
    return pd.DataFrame(df_dict)
```

```
df_1 = __to_dataframe('/content/drive/MyDrive/data_30.pkl')
```

```
100%|██████████| 1000000/1000000 [00:10<00:00, 98861.84it/s]
```

importing pickle.

Here we have created a function __to_dataframe and we have created a dictionary with the features and each feature consists of empty list. We are loading each row by using json from import json library and appending all the values in the empty features list.

```
df_1.head()
```

	hotel_url	author	date	rating	title	text	property_dict
0	Hotel_Review-g255326-d1774543-Reviews-Hotel_Ca...	bomomulls	2015-10-01T00:00:00	5.0	Food and accommodation	Absolutely amazing food. Massive servings of q...	{}
1	Hotel_Review-g255326-d1774543-Reviews-Hotel_Ca...	maceb23	2015-09-01T00:00:00	4.0	Functional and well priced	Was looking for a decent motel that was well p...	{}
2	Hotel_Review-g255326-d1774543-Reviews-Hotel_Ca...	Thegutho	2015-07-01T00:00:00	3.0	Basic but acceptable	Stayed here two nights, rooms were presented w...	{}
3	Hotel_Review-g255326-d1774543-Reviews-Hotel_Ca...	Anggood4567	2015-06-01T00:00:00	4.0	Occidental Motel - Orange	This motel, even though at the back of a pub, ...	{'cleanliness': 5.0, 'rooms': 4.0, 'service': ...
4	Hotel_Review-g255326-d1774543-Reviews-Hotel_Ca...	JonnyDundee	2015-04-01T00:00:00	3.0	One of the more basic motels	First of all, this is not a bad motel. It's ju...	{'value': 3.0, 'rooms': 3.0, 'service': 4.0}

After converting into csv, we have seen many nan values in the dataframe and property_dict column is consisted of sub ratings.

```
#normalize nested dictionary
df2 = pd.json_normalize(df_property['property_dict'])
```

```
df2.head(3)
```

	cleanliness	rooms	service	value	sleep quality	location	check in / front desk	business service (e.g., internet access)
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

We have normalized the nested dictionary that is property_dict using .json_normalize method. As every user has not rated for all the attributes we can see many nan values in this dataframe.

```
[ ] data2['Number_of_non_nans'] = data2.notna().sum(axis=1)
```

```
[ ] data2
```

	cleanliness	rooms	service	value	sleep quality	location	check in / front desk	business service (e.g., internet access)	Number_of_non_nans
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
3	5.0	4.0	5.0	NaN	NaN	NaN	NaN	NaN	3
4	NaN	3.0	4.0	3.0	NaN	NaN	NaN	NaN	3
...	...	...	...	...	...	...	...	...	...
999995	3.0	3.0	4.0	NaN	NaN	NaN	NaN	NaN	3
999996	NaN	NaN	1.0	1.0	NaN	3.0	NaN	NaN	3
999997	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0
999998	3.0	NaN	5.0	5.0	NaN	NaN	NaN	NaN	3
999999	NaN	3.0	4.0	5.0	NaN	NaN	NaN	NaN	3

We have created another column which shows the number of non nan values in the dataframe as we have limited resources, we cannot able to work with all the data. Some

```
#Selecting non NANs greater than equal to 7
df=data2[data2['Number_of_non_nans'] >= 7]
```

```
df.head()
```

	hotel_url	author	date	rating	title	text	cleanliness	rooms	service	value	sleep quality	location	check in / front desk	business service (e.g., internet access)	Number_of_non_nans
112	Hotel_Review-g551609-d1126210-Reviews-Old_Bank...	HONEST63	2008-08-01T00:00:00	3.0	Half a sausage	I have never been to Jersey and essentially bo...	4.0	3.0	3.0	3.0	NaN	3.0	3.0	3.0	7
406	Hotel_Review-g14126710-d319651-Reviews-Nakamur...	thetlistnet	2008-10-01T00:00:00	2.0	Not the best I have stayed	I stayed at various lodges/guesthouses during ...	2.0	2.0	3.0	2.0	NaN	3.0	4.0	3.0	7
407	Hotel_Review-g14126710-d319651-Reviews-Nakamur...	swam_nw	2008-09-01T00:00:00	2.0	Dirty bathroom	We didn't like the Nakamuraya Ryokan at all. F...	1.0	1.0	4.0	2.0	NaN	3.0	3.0	2.0	7
410	Hotel_Review-g14126710-d319651-Reviews-Nakamur...	Jobaki	2007-09-01T00:00:00	4.0	a real one	It is a very good traditional family owned ryo...	4.0	3.0	3.0	5.0	NaN	4.0	3.0	5.0	7
741	Hotel_Review-g60763-d93581-Reviews-The_Inn_at_...	sugarwater11	2008-05-01T00:00:00	3.0	Not what it used to be & yet even more expensive!	I have been staying at the Inn occasionally ov...	4.0	4.0	3.0	2.0	NaN	5.0	3.0	2.0	7

rows has number of non-nans as zero as they have all nan values.

We have taken those rows which has number of non-nans seven or more than seven. After doing that on an average we have eleven thousand records for each of the chunks.

Hotel name was in form of URL in the dataset. We have removed the URL part and convert it into a hotel name. Here we have initially striped the line in '- 'then we have split on 'Reviews-

```
#striping the text by '-' and spliting on Reviews,taking index 1
url['Company Symbol'] = url['hotel_url'].str.rstrip('-').str.split('Reviews-').str[1]
```

'

Then we have taken the index one part. In the next step er have again stripped in with'-' then split it on '.html' and have taken the string zero.

```
#striping the text by '-' and spliting on .html,taking index 0
url['hotel_name'] = url['Company Symbol'].str.rstrip('-').str.split('.html').str[0]
url.head(3)
```

	hotel_url	Company Symbol	hotel_name
112	Hotel_Review-g551609-d1126210-Reviews-Old_Bank...	Old_Bank_House_Hotel-Grouville_Jersey_Channel_...	Old_Bank_House_Hotel-Grouville_Jersey_Channel_...
406	Hotel_Review-g14126710-d319651-Reviews-Nakamur...	Nakamuraya_Ryokan-Chuo_Sapporo_Hokkaido.html	Nakamuraya_Ryokan-Chuo_Sapporo_Hokkaido
407	Hotel_Review-g14126710-d319651-Reviews-Nakamur...	Nakamuraya_Ryokan-Chuo_Sapporo_Hokkaido.html	Nakamuraya_Ryokan-Chuo_Sapporo_Hokkaido

```
#dropping hotel url and company symbol column
url=url.drop(['hotel_url','Company Symbol'], axis=1)
url.head(3)
```

	hotel_name
112	Old_Bank_House_Hotel-Grouville_Jersey_Channel_...
406	Nakamuraya_Ryokan-Chuo_Sapporo_Hokkaido
407	Nakamuraya_Ryokan-Chuo_Sapporo_Hokkaido


```
#dropping hotel url and company symbol column
url=url.drop(['hotel_url','Company Symbol'], axis=1)
url.head(3)
```

	hotel_name
112	Old_Bank_House_Hotel-Grouville_Jersey_Channel_...
406	Nakamuraya_Ryokan-Chuo_Sapporo_Hokkaido
407	Nakamuraya_Ryokan-Chuo_Sapporo_Hokkaido


```
#replacing '_' with ' '
url.hotel_name =url.hotel_name.str.replace('_',' ')
```

Merging all the chunks and convert to a large csv file:

```
#Reading all the CSVs and assign into dfs variables
df1 = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Hitesh Pickle/data_1')
df2= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Hitesh Pickle/data_2')
df3= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Hitesh Pickle/data_3')
df4= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Hitesh Pickle/data_4')
df5= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Hitesh Pickle/data_5')
df6= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Hitesh Pickle/data_6')
df7= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Hitesh Pickle/data_7')
df8= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Hitesh Pickle/data_8')
df9= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Hitesh Pickle/data_9')
df10= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Hitesh Pickle/data_10')
df11= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Daksh Pickle/df_11')
df12= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Daksh Pickle/df_12')
df13= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Daksh Pickle/df_13')
df14= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Daksh Pickle/df_14')
df15= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Daksh Pickle/df_15')
df16= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Daksh Pickle/df_16')
df17= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Daksh Pickle/df_17')
df18= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Daksh Pickle/df_18')
df19= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Daksh Pickle/df_19')
df20= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Daksh Pickle/df_20')
df21= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/csv_new/df_21')
df22= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/csv_new/df_22')
df23= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/csv_new/df_23')
df24= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/csv_new/df_24')
df25= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/csv_new/df_25')
df26= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/csv_new/df_26')
df27= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/csv_new/df_27')
df28= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/csv_new/df_28')
df29= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/csv_new/df_29')
df30= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/csv_new/df_30')
df31= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_31')
df32= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_32')
df33= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_33')
df34= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_34')
df35= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_35')
df36= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_36')
df37= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_37')
df38= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_38')
df39= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_39')
df40= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_40')
df41= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_41')
df42= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_42')
df43= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_43')
df44= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_44')
df45= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Saswata/df_45')
df46= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Madhurima/df_46')
df47= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Madhurima/df_47')
df48= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Madhurima/df_48')
df49= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Madhurima/df_49')
df50= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Madhurima/df_50')
df51= pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Madhurima/df_51')
```

Here we have read all the csv chunks and concat them to form a large csv file.

```
#Concat all the dataframe into single dataframe
df = pd.concat([df1, df2, df3, df4, df5, df6, df7, df8, df9, df10,df11, df12, df13, df14, df15, df16, df17, df18, df19, df20,df21, df22, df23, df24, df25, df26, df27, df28, df29, df30, df31, df32, df33, df34, df35, df36, df37, df38, df39, df40, df41, df42, df43, df44, df45, df4
display(df)
```

	author	date	rating	title	text	service	location	value	cleanliness	sleep quality	rooms	check in / front desk	business service (e.g., internet access)	hotel_name
0	Conners	2008-07-01T00:00:00	4.0	Lovely hotel	We had the weather so the hotel was a good exp...	5.0	5.0	5.0	5.0	NaN	5.0	5.0	5.0	Le Grand Large-Rivedoux Plage Ile de Re Charen...
1	Fisk123	2008-06-01T00:00:00	2.0	Too near the very busy road	I booked 3 nights at this hotel via an agency ...	2.0	1.0	2.0	2.0	NaN	1.0	3.0	2.0	Le Grand Large-Rivedoux Plage Ile de Re Charen...
2	Kendalswimmer	2008-09-01T00:00:00	2.0	Great location - appalling staff	There is nothing wrong with the hotel - the ro...	1.0	4.0	2.0	2.0	NaN	2.0	1.0	1.0	Gran Sol Hotel-Zahara de los Atunes Costa de l...
3	CasaCampana	2008-05-01T00:00:00	2.0	Lazy owner?	Just back from a week in this hotel. Wisely ch...	2.0	5.0	2.0	2.0	NaN	3.0	3.0	1.0	Gran Sol Hotel-Zahara de los Atunes Costa de l...
4	Phicus	2006-12-01T00:00:00	2.0	Great Beach, Less than expected from hotel!	After having seen the positive reviews we arri...	2.0	4.0	1.0	2.0	NaN	2.0	1.0	1.0	Gran Sol Hotel-Zahara de los Atunes Costa de l...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2874	intl_realist	2006-12-01T00:00:00	5.0	Great value	This hotel was a great bargain for under $100 ...	5.0	5.0	5.0	5.0	NaN	5.0	5.0	4.0	DoubleTree by Hilton Hotel Boston Bayside-Bost...
2875	Scrambler2	2006-10-01T00:00:00	5.0	Great suprise	This hotel chain is part of the Hilton family ...	4.0	3.0	4.0	4.0	NaN	4.0	4.0	4.0	DoubleTree by Hilton Hotel Boston Bayside-Bost...
2876	AZCat	2006-09-01T00:00:00	4.0	Good, inexpensive South Boston area hotel	First directions: hotel is adjacent to Bayside...	4.0	4.0	5.0	4.0	NaN	4.0	4.0	5.0	DoubleTree by Hilton Hotel Boston Bayside-Bost...
2877	jademarc88	2006-09-01T00:00:00	4.0	Good Value	Great location - especially with the complimen...	4.0	4.0	5.0	3.0	NaN	4.0	4.0	3.0	DoubleTree by Hilton Hotel Boston Bayside-Bost...
2878	dm1313	2006-08-01T00:00:00	4.0	Not bad at all!	I stayed here in August - at first i was put o...	5.0	3.0	4.0	5.0	NaN	5.0	5.0	5.0	DoubleTree by Hilton Hotel Boston Bayside-Bost...

563986 rows × 14 columns

Now we have csv files that has almost five lakhs sixty thousand rows and fourteen columns.

In the large csv file, we have done some pre-processing as some of the columns has nan values and author and hotel id were not present there. So, we have created the ids and remove the nan values.

```
1  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 563986 entries, 0 to 563985
Data columns (total 14 columns):
 #   Column                                  Non-Null Count   Dtype
---  ------                                  --------------   -----
 0   author                                  562655 non-null  object
 1   date                                    563986 non-null  object
 2   rating                                  563986 non-null  float64
 3   title                                   563986 non-null  object
 4   text                                    563986 non-null  object
 5   service                                 563972 non-null  float64
 6   location                                563984 non-null  float64
 7   value                                   563940 non-null  float64
 8   cleanliness                             563984 non-null  float64
 9   sleep quality                           1152 non-null    float64
 10  rooms                                   563979 non-null  float64
 11  check in / front desk                   563971 non-null  float64
 12  business service (e.g., internet access)  563708 non-null  float64
 13  hotel_name                              563986 non-null  object
dtypes: float64(9), object(5)
memory usage: 60.2+ MB
```

As we can see the author, rooms, check in / front desk, business service (e.g., internet access) sleep quality services have nan values.

```
1  df.isna().sum()
```

```
author                                    1331
date                                         0
rating                                       0
title                                        0
text                                         0
service                                     14
location                                     2
value                                       46
cleanliness                                  2
sleep quality                           562834
rooms                                        7
check in / front desk                       15
business service (e.g., internet access)   278
hotel_name                                   0
dtype: int64
```

Author has 1331 null values in the column. Most of the null values is present in the sleep quality.

For further operation we need to give the ids to the author and hotel_name.

We have done these things by using ‘pd.factorize’, that has generated unique ids for both the columns.

	hotel_name	hotel_id
0	Le Grand Large-Rivedoux Plage Ile de Re Charen...	0
1	Le Grand Large-Rivedoux Plage Ile de Re Charen...	0
2	Gran Sol Hotel-Zahara de los Atunes Costa de l...	1
3	Gran Sol Hotel-Zahara de los Atunes Costa de l...	1
4	Gran Sol Hotel-Zahara de los Atunes Costa de l...	1
...	...	...
562650	DoubleTree by Hilton Hotel Boston Bayside-Bost...	91302
562651	DoubleTree by Hilton Hotel Boston Bayside-Bost...	91302
562652	DoubleTree by Hilton Hotel Boston Bayside-Bost...	91302
562653	DoubleTree by Hilton Hotel Boston Bayside-Bost...	91302
562654	DoubleTree by Hilton Hotel Boston Bayside-Bost...	91302

562655 rows × 2 columns

	author	author_id
0	Conners	0
1	Fisk123	1
2	Kendalswimmer	2
3	CasaCampana	3
4	Phicus	4
...	...	...
562650	intl_realist	46957
562651	Scrambler2	197590
562652	AZCat	431502
562653	jademarc88	431503
562654	dm1313	73635

562655 rows × 2 columns

author	date	rating	title	text	service	location	value	cleanliness	sleep quality	rooms	check in / front desk	services_like_internet_access	hotel_name	hotel_id	author_id
Conners	2008-07-01T00:00:00	4.0	Lovely hotel	We had the weather so the hotel was a good exp...	5.0	5.0	5.0	5.0	NaN	5.0	5.0	5.0	Le Grand Large-Rivedoux Plage Ile de Re Charen...	0	0
Fisk123	2008-06-01T00:00:00	2.0	Too near the very busy road	I booked 3 nights at this hotel via an agency ...	2.0	1.0	2.0	2.0	NaN	1.0	3.0	2.0	Le Grand Large-Rivedoux Plage Ile de Re Charen...	0	1
Kendalswimmer	2008-09-01T00:00:00	2.0	Great location - appalling staff	There is nothing wrong with the hotel - the ro...	1.0	4.0	2.0	2.0	NaN	2.0	1.0	1.0	Gran Sol Hotel-Zahara de los Atunes Costa de l...	1	2
CasaCampana	2008-05-01T00:00:00	2.0	Lazy owner?	Just back from a week in this hotel. Wisely ch...	2.0	5.0	2.0	2.0	NaN	3.0	3.0	1.0	Gran Sol Hotel-Zahara de los Atunes Costa de l...	1	3
Phicus	2006-12-01T00:00:00	2.0	Great Beach, Less than expected from hotel!	After having seen the positive reviews we arri...	2.0	4.0	1.0	2.0	NaN	2.0	1.0	1.0	Gran Sol Hotel-Zahara de los Atunes Costa de l...	1	4

But as some values were empty in the author column the id is generated as -1 for those of the records. We have dropped the records where author id is -1.

```
i = df[(df.author_id == -1)].index
```

```
df.drop(i,inplace=True)
```

```
df[df.author_id==-1]
```

author	date	rating	title	text	service	location	value	cleanliness	sleep quality	rooms	check in / front desk	services_like_internet_access	hotel_name	hotel_id	author_id

Then we have saved the dataset as factorized.csv.

```
print(dataset.author_id.value_counts())
print(dataset.author.value_counts())
```

```
31849     89
7336      63
20802     46
22141     41
5763      38
          ..
171185     1
171183     1
171182     1
171180     1
431503     1
Name: author_id, Length: 431504, dtype: int64
Ozcfied                  89
BA-traveler              63
FrqTravlr                46
FarAndWide_Traveller     41
Ellemay                  38
                         ..
GlobalBiz                 1
Sunshine805919            1
hotsy                     1
arlu                      1
jademarc88                1
Name: author, Length: 431504, dtype: int64
```

```
print(dataset.hotel_id.value_counts())
print(dataset.hotel_name.value_counts())
```

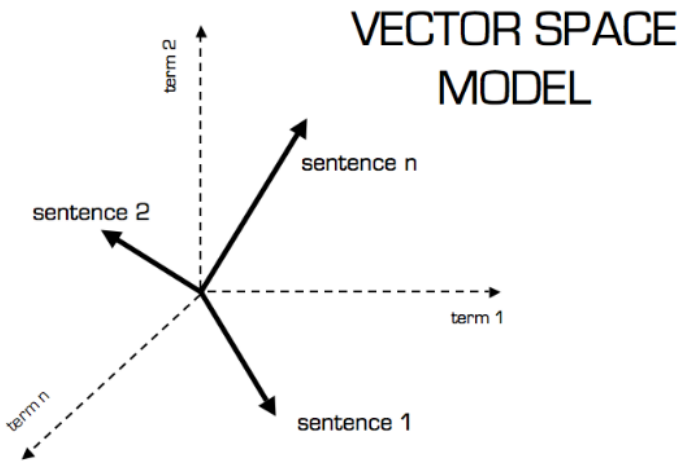
```
88945    493
24828    467
88905    436
24668    403
89407    394
        ...
67156      1
67155      1
41525      1
67152      1
45650      1
Name: hotel_id, Length: 91219, dtype: int64
Hotel Riu Cancun-Cancun Yucatan Peninsula                                    493
Grand Bahia Principe San Juan-Rio San Juan Maria Trinidad Sanchez Province Dominican R    467
Iberostar Costa Dorada-Puerto Plata Puerto Plata Province Dominican Republic             436
Hotel Riu Caribe-Cancun Yucatan Peninsula                                    403
Sensimar Resort Riviera Maya-Puerto Aventuras Yucatan Peninsula              394
                                                                            ...
Epsom Motel-Epsom North Island                                                 1
Hotel Residenza G R A 21-Rome Lazio                                            1
PLAZAHOTEL Hanau-Hanau Hesse                                                   1
Niforeika Beach-Niforeika Achaea Region West Greece                            1
Holiday Inn Express Seaford Route 13-Seaford Delaware                          1
Name: hotel_name, Length: 91219, dtype: int64
```

Now we have author ids and hotel ids for unique authors and hotels.

Text Pre-processing:

These various text pre-processing steps are widely used for dimensionality reduction.

In the vector space model, each word/term is an axis/dimension. The text/document is represented as a vector in the multi-dimensional space.



The number of unique words means the number of dimensions.

Some key concept of text pre-processing: -

Tokenization: text is split into smaller units. We can use either sentence tokenization or word tokenization based on our problem statement.

Stop words: - I, me, my, myself, we, our, ours, ourselves, you, you're etc. are called stop words.

Lemmatization: - It stems the word but makes sure that it does not lose its meaning. Lemmatization has a pre-defined dictionary that stores the context of words and checks the word in the dictionary while diminishing. Examples- Going>Go,Geese>Goose

For text pre-processing part we have taken text column from each of the chunks. As we have mentioned before, every chunk has different no of records after chunks pre-processing.

```
[ ]  1  # For each review segregating each sentence into new column
     2  #The avarage sentence per review is 14
     3  new = data_text["text"].str.split(".", n = 50, expand = True)
```

Initially we were trying to proceed with raw data column but we were facing some issues related to resources so we have divided the reviews into fifty columns as the average number of sentences in reviews is around 14, to be in safe side we have taken fifty.

Again, we have stacked all the sentences with id column.

```
1  data_text.head()
```

	id	text
0	0	if you"re looking for a clean but very average...
1	0	however, do not expect a true 4 star hotel! w...
2	0	The location is pretty good-a bit off the bea...
3	0	the hotel is very clean (but as someone point...
4	0	The pool is very nice, but the rooms are smal...

Some rows are none so we have removed the none values with regular expression function.

```
# Remove Emails
data = [re.sub('\S*@\S*\s?', '', sent) for sent in data]
```

We have removed if any emails are there.

```
# Remove new line characters
data = [re.sub('\s+', ' ', sent) for sent in data]
```

Here we have removed the new line characters.

```
# Remove distracting single quotes
data = [re.sub("\'", "", sent) for sent in data]
```

Here we have removed if there is any quotations.

```
#Removing https link
data = [re.sub('http\S+', '', sent) for sent in data]
```

Here we have removed the hyperlinks contained sentences.

```
#Removing alpha-numeric
data = [re.sub('[^a-z0-9A-Z_]',' ', sent) for sent in data]
```

Here we have removed alpha numeric characters.

```
#Removing exclamation marks
data = [re.sub('!{2,}','!', sent) for sent in data]
```

We have removed exclamation marks if there are one or more.

```
#Removing double space
data = [re.sub(' {2,}',' ', sent) for sent in data]
```

Here we have removed spaces if there are more than empty spaces.

```
#Removing single or double letter
data = [re.compile(r'\W*\b\w{1,2}\b').sub('',sent)for sent in data]
```

Here we have removed the single and double letter characters we found during pre-processing.

Topic Modelling:

Topic modelling is an unsupervised method of classifying documents, similar to clustering of numerical data, to find natural groupings of elements (topics) even when you don't know what you are looking for.

A document can be a part of multiple topics, kind of like in fuzzy clustering (soft clustering) in which each data point belongs to more than one cluster.

Importance:

a. discovering the hidden themes in the collection.
b. classifying the documents into the discovered themes.
c. using the classification to organize/summarize/search the documents

Documents can be part of multiple topics, similar to fuzzy clustering (soft clustering) where each data point belongs to multiple clusters.

```
def sent_to_words(sentences):
    for sentence in sentences:
        yield(gensim.utils.simple_preprocess(str(sentence), deacc=True))  # deacc=True removes punctuations
```

We have created a function 'sent_to_word' where we will give the sentences.

gensim.utils.simple_preprocess(str(sentence), deacc=True)) Convert a document into a list of tokens. This lowercases, tokenizes, de-accents (optional), the output are final tokens = unicode strings, that won't be processed any further. deacc=True removes punctuations.

```
#converting the words into list
data_words = list(sent_to_words(data))

print(data_words[:1])
```

```
[['you', 'looking', 'for', 'clean', 'but', 'very', 'average', 'place', 'stay', 'zermatt', 'this', 'was', 'our', 'only', 'option', 'given', 'busy', 'xmas', 'time', 'this', 'fine', 'place']]
```

Above is the output after tokenization and lowering case of words and removing punctuations.

A bigram is two words that often appear together in a document. Trigrams are three common words.

Some examples in our example are: 'front_bumper', 'oil_leak', 'maryland_college_park' etc.

```
# Build the bigram and trigram models
bigram = gensim.models.Phrases(data_words, min_count=5, threshold=100) # higher threshold fewer phrases.
trigram = gensim.models.Phrases(bigram[data_words], threshold=100)
```

Gensim's phrase model can create and implement bigrams, trigrams, quadgrams, and more. Two important arguments to the phrase are min_count and threshold. The higher the values of these parameters, the more difficult it is to combine words into bigrams.

```
print(trigram_mod[bigram_mod[data_words[0]]])
```

```
/local/lib/python3.7/dist-packages/gensim/models/phrases.py:598: UserWarning: For a faster implementation, use the gensim.models.phrases.Phraser class
rnings.warn("For a faster implementation, use the gensim.models.phrases.Phraser class")
u', 'looking', 'for', 'clean', 'but', 'very', 'average', 'place', 'stay', 'zermatt', 'this', 'was', 'our', 'only', 'option', 'given', 'busy', 'xmas', 'time', 'this', 'fine', 'place']
```

In our case we do not find any combined words in the first token but maybe there are some.

The bigram model is ready. Let's remove stopwords, create bigrams and lemmatizations, and define a function to call them in order.

```
# Define functions for stopwords, bigrams, trigrams and lemmatization
def remove_stopwords(texts):
    return [[word for word in simple_preprocess(str(doc)) if word not in stop_words] for doc in texts]
```

This function is used to remove the stop words from the tokens.

```
def make_bigrams(texts):
    return [bigram_mod[doc] for doc in texts]
```

This function will use the bigram model we trained.

```
#filtering with noun,adj,verb,adverb,proper noun
def lemmatization(texts, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV','PROPN']):
```

This function has been used for lemmatization and we have filtered with Noun, Adjective, Verb, Adverb, Proper Noun.

```
    texts_out = []
    for sent in texts:
        doc = nlp(" ".join(sent))
        texts_out.append([token.lemma_ for token in doc if token.pos_ in allowed_postags])
    return texts_out
```

The tokens with checked with allowed parts of speech after lemmatization then it will be appended to texts_out list.

```
[['look', 'clean', 'average', 'place', 'stay', 'zermatt', 'option', 'give', 'busy', 'xmas', 'time', 'fine', 'place']]
```

The above is the output after applying all the functions.

The two main inputs to the LDA topic model are the dictionary(id2word) and the corpus.

Token – A token means a 'word'.

Document – A document refers to a sentence or paragraph.

Corpus – It refers to a collection of documents as a bag of words (BoW)

To work with text documents, Gensim also needs words, or tokens, which must be converted into unique IDs. To accomplish this, we provide the functionality of a Dictionary object that maps each word to a unique integer ID. To do this, we convert the input text to a dictionary and pass it to the corpora.Dictionary() object.

```
# Create Dictionary
id2word = corpora.Dictionary(data_lemmatized)
```

Here the lemmatized list of words are provided to get unique ids.

```
# Create Corpus
texts = data_lemmatized

# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]
```

The corpus is created which is basically collections of all the lemmatized words. The produced corpus shown above is a mapping of (word_id, word frequency).

Latent Dirichlet Allocation (LDA):

This is one of the most popular topic modelling techniques. Every document consists of different words and every topic has different words associated with it. The purpose of LDA is to find the topic a document belongs to base on the words in the document.

```
Doc1: word1, word3, word5, word45, word11, word 62, word88 ...
Doc2: word9, word77, word31, word58, word83, word 92, word49 ...
Doc3: word44, word18, word52, word36, word64, word 11, word20 ...
Doc4: word85, word62, word19, word4, word30, word 94, word67 ...
Doc5: word19, word53, word74, word79, word45, word 39, word54 ...
```

We have 5 documents each containing the words listed in front of them (ordered by frequency of occurrence).

Each row in the table represents a different topic and each column a different word in the corpus. Each cell contains the probability that the word(column) belongs to the topic(row).

	Word1	word2	word3	word4	.....
Topic1	0.01	0.23	0.19	0.03	
Topic2	0.21	0.07	0.48	0.02	
Topic3	0.53	0.01	0.17	0.04	

Find the Word that represent a topic:

We can sort the words with respect to their probability score.

a. If x = 10, we'll sort all the words in topic1 based on their score and take the top 10 words to represent the topic or,

   you can set the score threshold. All words in the topic with scores above the threshold can be stored as representatives in order of score.

Example Let's say we have 2 topics that can be classified as room related and food related. A topic has probabilities for each word, so words such as sleep, room, area, shower, large etc will have a higher probability in the room_related topic than in the food_related one.

```
(7,
 '0.041*"room" + 0.017*"towel" + 0.014*"large" + 0.013*"area" + 0.011*"feel" '
 '+ 0.010*"shower" + 0.010*"size" + 0.010*"bathroom" + 0.009*"make" + '
 '0.008*"change"'),
```

The food_related topic, likewise, will have high probabilities for words such as restaurant, hotel, location, great, good, food, bar etc.

```
(1,
 '0.044*"restaurant" + 0.040*"hotel" + 0.034*"location" + 0.033*"great" + '
 '0.029*"good" + 0.022*"food" + 0.017*"area" + 0.015*"bar" + 0.013*"close" + '
 '0.013*"shop"'),
```

Assumptions:

a. Each document is simply a collection of words, or a "word bag". Therefore, word order and grammatical roles of words (subject, object, verb, etc.) are not considered in the model.

b. Words like am/is/are/of/a/the/but/… do not contain information about the "topic" and can be removed from the document as a pre-processing step. In fact, you can remove words in at least 80-90% of your documents without losing any information.
c. We know beforehand how many topics we want. 'k' is pre-decided.

Workings:

a. The words that belong to a document, that we already know.
b. The words that belong to a topic or the probability of words belonging into a topic, that we need to calculate.
c. Go through each document and randomly assign each word in the document to one of k topics (k is chosen beforehand).
d. For each document d, go through each word w and compute:
   a.
   1. P1- p (topic t | document d): the proportion of words in document d that are assigned to topic t. Tries to capture how many words belong to the topic t for a given document d. Excluding the current word.
   2. If a lot of words from d belongs to t, it is more probable that word w belongs to t.
   3. (#words in d with t +alpha/ #words in d with any topic+ k*alpha), alpha is a hyper parameter.

   b.
   1. P2- p(word w| topic t): the proportion of assignments to topic t over all documents that come from this word w. Tries to capture how many documents are in topic t because of word w.
   c. The current topic – word assignment is updated with a new topic with the probability, product of P1 and P2.

i) LDA represents documents as a mixture of topics. Similarly, a topic is a mixture of words.

ii) If a word, has high probability of being in a topic, all the documents having w will be more strongly associated with topic, t, as well.

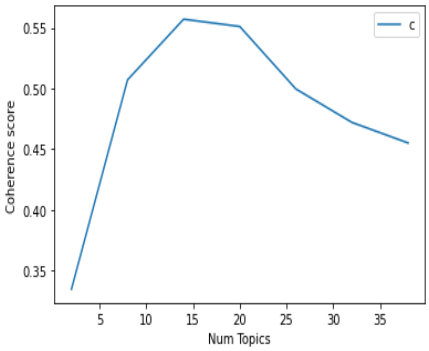
iii) Similarly, if word, w, is not very probable to be in topic, t, the documents which contain the word, w, will be having very low probability of being in topic, t, because rest of the words in document, d, will belong to some other topic. hence d will have a higher probability for those topics.

```
coherence_values = []
model_list = []
for num_topics in range(start, limit, step):
    model=gensim.models.LdaMulticore(random_state=100, chunksize=1500,corpus=corpus, id2word=dictionary, num_topics=num_topics,alpha=0.1,)
    model_list.append(model)
    coherencemodel = gensim.models.CoherenceModel(model=model, texts=texts, dictionary=dictionary, coherence='c_v')
    coherence_values.append(coherencemodel.get_coherence())

return model_list, coherence_values
```

LDA Model:

Here chunksize is the number of documents to be used in each training chunk.



The topic coherence measure scores a single topic by measuring the degree of semantic similarity between high-scoring words within the topic. These measures help distinguish between topics that are semantically interpretable and topics that are artifacts of statistical inference.

We calculated the coherence score with number of topic and we get that the number of topics can be 16 where coherence score is more than 0.55.

```
[(0,
  '0.075*"breakfast" + 0.035*"food" + 0.024*"good" + 0.017*"make" + '
  '0.013*"excellent" + 0.012*"restaurant" + 0.012*"dinner" + 0.012*"serve" + '
  '0.011*"eat" + 0.011*"morning"'),
```

From topic 1 we can see that it is talking about the breakfast.

```
(1,
 '0.026*"minute" + 0.026*"walk" + 0.025*"hotel" + 0.012*"easy" + 0.011*"take" '
 '+ 0.010*"bus" + 0.010*"airport" + 0.010*"get" + 0.010*"drive" + '
 '0.009*"charge"'),
```

Topic 2 is talking about the location of the hotel.

```
(2,
 '0.012*"fruit" + 0.012*"coffee" + 0.009*"cereal" + 0.009*"juice" + '
 '0.009*"breakfast" + 0.009*"car" + 0.007*"bread" + 0.007*"egg" + '
 '0.006*"also" + 0.006*"taste"'),
```

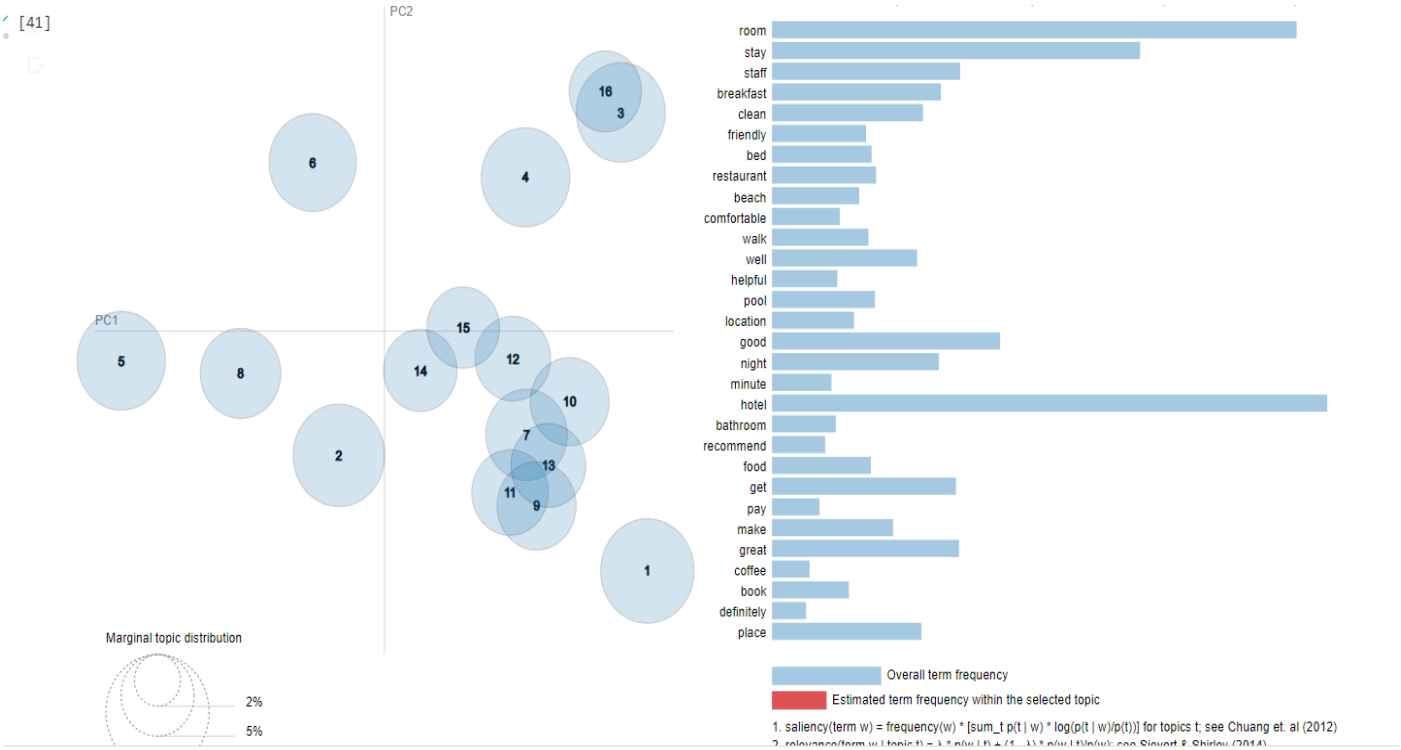
Topic 3 – Breakfast or food.

```
(3,
 '0.033*"pool" + 0.025*"area" + 0.018*"nice" + 0.017*"good" + 0.017*"stay" + '
 '0.016*"room" + 0.013*"hotel" + 0.012*"enjoy" + 0.012*"really" + '
 '0.012*"great"'),
```

Topic – 4 is talking about the hotel and so on.

The values associated with the word is saying about the weightage of the word on the topic.

The weights reflect how important a keyword is to that topic.



i. Each bubble on the left-hand side plot represents a topic. The larger the bubble, the more prevalent is that topic.

ii. A good topic model will have fairly big, non-overlapping bubbles scattered throughout the chart instead of being clustered in one quadrant.

Dominant topic in each sentence:

One practical application of topic modeling is to determine what topic a given document is about.

To find out, we find the topic number that has the highest percentage contribution in that document. The format_topics_sentences() function below nicely aggregates this information into a presentable table.

```
def format_topics_sentences(ldamodel=lda_model, corpus=corpus, texts=data):
    # Init output
    sent_topics_df = pd.DataFrame()

    # Get main topic in each document
    for i, row in enumerate(ldamodel[corpus]):
        row = sorted(row, key=lambda x: (x[1]), reverse=True)
        # Get the Dominant topic, Perc Contribution and Keywords for each document
        for j, (topic_num, prop_topic) in enumerate(row):
            if j == 0:  # => dominant topic
                wp = ldamodel.show_topic(topic_num)
                topic_keywords = ", ".join([word for word, prop in wp])
                sent_topics_df = sent_topics_df.append(pd.Series([int(topic_num), round(prop_topic,4), topic_keywords]), ignore_index=True)
            else:
                break
    sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution', 'Topic_Keywords']
```

	Document_No	Dominant_Topic	Topic_Perc_Contrib	Keywords	Text
0	0	1.0	0.5833	restaurant, hotel, location, great, good, food...	Lodge the middle where
1	1	14.0	0.5415	breakfast, room, coffee, free, nice, morning, ...	The lodge diner was GREAT the little gift sho...
2	2	13.0	0.3081	room, get, check, hotel, night, even, day, go,...	The check and out was normal
3	3	6.0	0.7321	beach, pool, walk, minute, take, water, see, g...	Pool was salt water pool
4	4	3.0	0.4230	room, clean, bed, comfortable, bathroom, nice,...	Very nice
5	5	3.0	0.7727	room, clean, bed, comfortable, bathroom, nice,...	Rooms were same most hotels worked fine both ...
6	6	10.0	0.4132	hotel, star, guest, airport, hour, noise, reso...	The only thing wished knew about were the tra...
7	7	6.0	0.3176	beach, pool, walk, minute, take, water, see, g...	They run right behind the lodge and they run ...
8	8	14.0	0.2746	breakfast, room, coffee, free, nice, morning, ...	You can get free ear plugs the desk
9	9	5.0	0.5833	stay, hotel, place, recommend, back, time, gre...	was pleasant stay

Most representative document for each topic

Topic keywords alone may not be enough to understand what the topic is about. So, to better understand a topic, you can find the document that a particular topic contributed the most and read this document to infer that topic.

```
# Group top 5 sentences under each topic
sent_topics_sorteddf = pd.DataFrame()

sent_topics_outdf_grpd = df_topic_sents_keywords.groupby('Dominant_Topic')

for i, grp in sent_topics_outdf_grpd:
    sent_topics_sorteddf = pd.concat([sent_topics_sorteddf,
                                      grp.sort_values(['Perc_Contribution'], ascending=[0]).head(1)],
                                     axis=0)

# Reset Index
sent_topics_sorteddf.reset_index(drop=True, inplace=True)

# Format
sent_topics_sorteddf.columns = ['Topic_Num', "Topic_Perc_Contrib", "Keywords", "Text"]

# Show
sent_topics_sorteddf.head()
```

	Topic_Num	Topic_Perc_Contrib	Keywords	Text
0	0.0	0.8972	well, place, old, hotel, travel, sleep, keep, ...	The childrens playground was also very small ...
1	1.0	0.9148	restaurant, hotel, location, great, good, food...	Wawel Schindlers The main square Rynek Glowny...
2	2.0	0.9148	hotel, pay, night, price, much, find, stay, ro...	did have queue for while check but did arrive...
3	3.0	0.9038	room, clean, bed, comfortable, bathroom, nice,...	The room interiors are very well designed the...
4	4.0	0.9038	hotel, book, area, really, get, breakfast, dri...	Calling the property directly received rate a...

Topic distribution across documents:

Finally, we want to understand the volume and distribution of topics in order to judge how

	Dominant_Topic	Topic_Keywords	Num_Documents	Perc_Documents
0.0	1.0	restaurant, hotel, location, great, good, food...	23895.0	0.1748
1.0	14.0	breakfast, room, coffee, free, nice, morning, ...	9325.0	0.0682
2.0	13.0	room, get, check, hotel, night, even, day, go,...	5990.0	0.0438
3.0	6.0	beach, pool, walk, minute, take, water, see, g...	10408.0	0.0762
4.0	3.0	room, clean, bed, comfortable, bathroom, nice,...	4763.0	0.0349
5.0	3.0	room, clean, bed, comfortable, bathroom, nice,...	13272.0	0.0971
6.0	10.0	hotel, star, guest, airport, hour, noise, reso...	8046.0	0.0589
7.0	6.0	beach, pool, walk, minute, take, water, see, g...	6353.0	0.0465
8.0	14.0	breakfast, room, coffee, free, nice, morning, ...	7981.0	0.0584
9.0	5.0	stay, hotel, place, recommend, back, time, gre...	6262.0	0.0458
10.0	5.0	stay, hotel, place, recommend, back, time, gre...	5395.0	0.0395
11.0	0.0	well, place, old, hotel, travel, sleep, keep, ...	4648.0	0.0340
12.0	15.0	stay, make, day, hotel, good, week, night, tri...	5169.0	0.0378
13.0	0.0	well, place, old, hotel, travel, sleep, keep, ...	9701.0	0.0710
14.0	5.0	stay, hotel, place, recommend, back, time, gre...	8692.0	0.0636
15.0	3.0	room, clean, bed, comfortable, bathroom, nice,...	6770.0	0.0495
16.0	11.0	good, hotel, value, also, perfect, station, mo...	NaN	NaN

widely it was discussed. The below table exposes that information.

Data Visualization:

Hotel ratings are often used to classify hotels according to their quality. From the initial purpose of informing travellers on basic facilities that can be expected, the objectives of hotel rating have expanded into a focus on the hotel experience as a whole. Below are the ratings given by the user based on various factors such as service, location, value, cleanliness, business service (e.g., internet service), check in/ front desk etc.

Figure:1 below shows the rating provided by authors based on their experience in hotels in the above dataset.

```
5.0    270015
4.0    147944
3.0     54220
2.0     48288
1.0     43519
Name: rating, dtype: int64
```

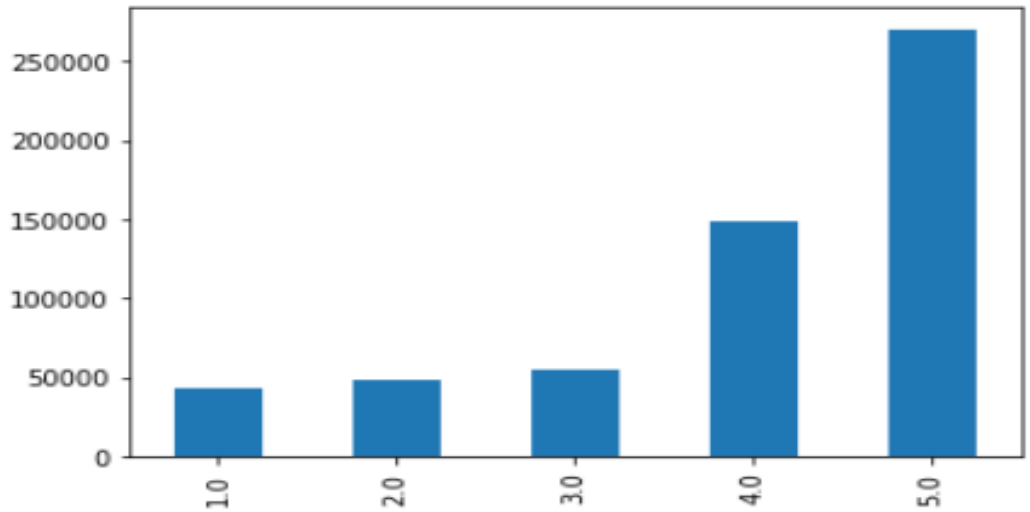


Figure 1: Hotel ratings given by authors

From Figure 1, we can say that

- Among the total number of authors (563986), maximum authors (74.1%) gave ratings between 4.0 and 5.0 i.e., most people liked the Hotel during their stay.
- There are authors (91807) who were not satisfied with the Hotel during their stay as they game them rating between 1.0-2.0.

Figure:2 below shows the rating provided by authors based on the service provided by the Hotels. From this we can saw that:

```
5.0    296340
4.0    124389
3.0     69951
1.0     39398
2.0     33894
Name: service, dtype: int64
```

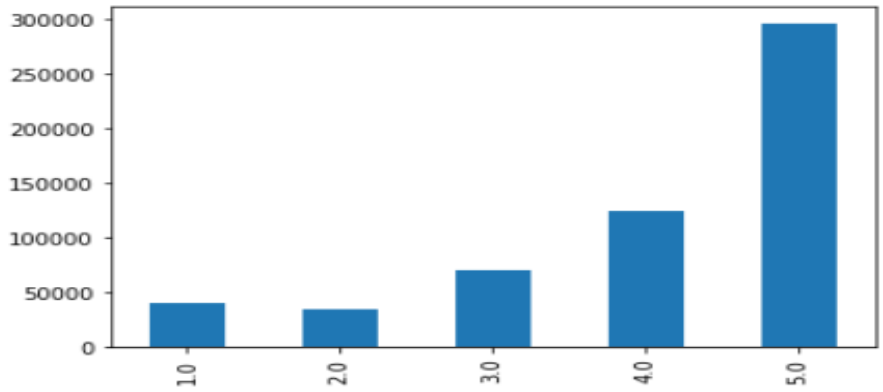


Figure 2: Hotel's rating based on service

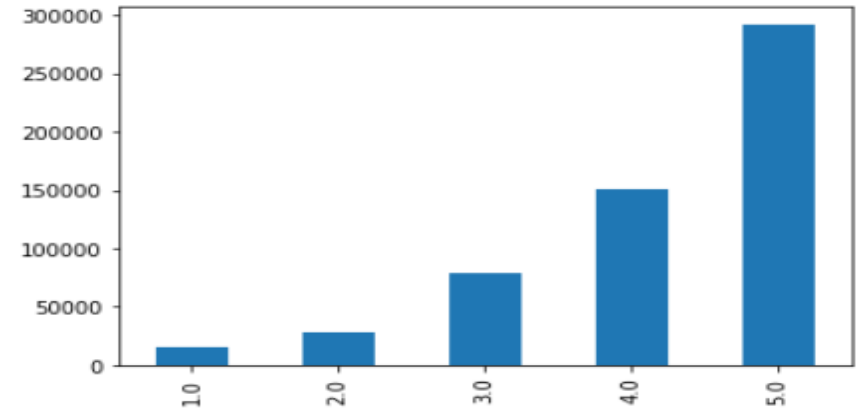
- Majority of authors are satisfied with the hotel service and rated 5.0

• There are hotels which have offered poor service to the authors and thus have been rated with 1.0 & 2.0, i.e.,

there are hotels which doesn't provide good service to their customers.

- If a customer wants a hotel with good service, he should be recommended with hotels

```
5.0    292059
4.0    150691
3.0     78393
2.0     27965
1.0     14876
Name: location, dtype: int64
```



with 5.0 ratings.

figure 3: Rating based on Hotel's Location

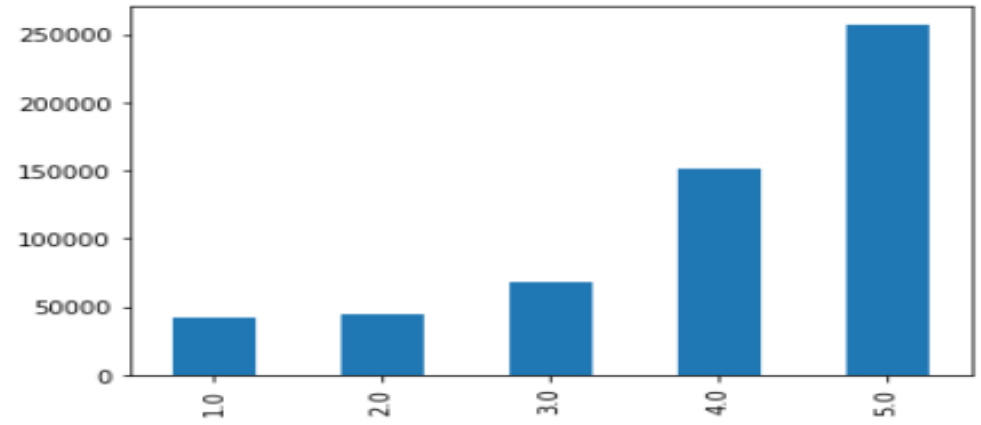
Location plays an important role in selecting a hotel. Mostly people prefer locations which are attractive, and locations usually located near to different transport links. In Figure 3, we can clearly see how's user ratings vary with respect to location.

• The hotel with good ambience and location will get 5.0 and the hotel with not so favourable location will get 1.0. In our dataset, more than 50% customers gave 5.0 rating based on location.

• This will help people in recommending good hotels based on location preference.

Figure 4, shows the rating of the hotel based on its value, Finding the best value of hotel's

```
5.0    257225
4.0    151066
3.0     67938
2.0     45087
1.0     42624
Name: value, dtype: int64
```

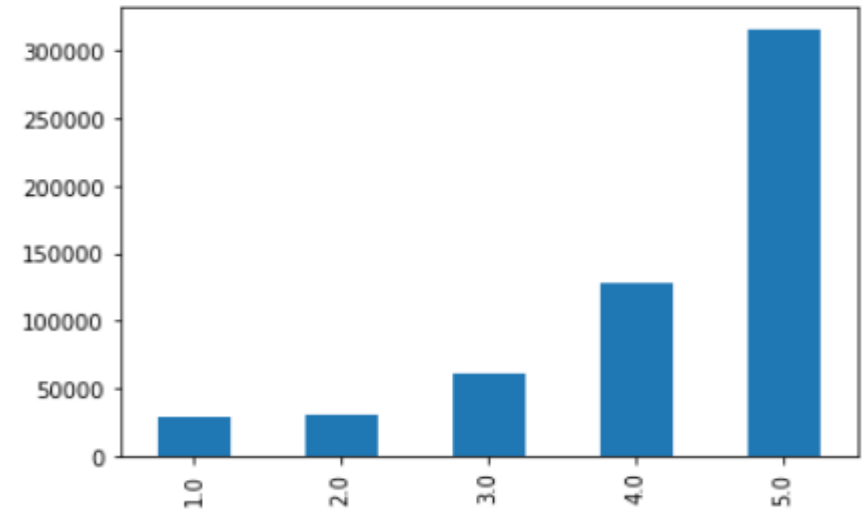


room is an important factor while Figure 4: Rating based on Hotel's value

selecting a hotel. The best hotel for your stay won't be the one with the lowest price tag, but it likely won't be the one with the highest price either! Instead of just looking at the cost, consider the overall value of the stay you'll be having. The hotel that offers the best total value of price, amenities, and location for your upcoming stay is the one you should book.

Thus, the ratings help in recommending the best value of hotel's room.

```
5.0    315954
4.0    127714
3.0     61044
2.0     30013
1.0     29259
Name: cleanliness, dtype: int64
```



As in the figure we can see that

There are hotels where customer gave 1.0(5.1%) and 2.0(5.3%). If a customer is looking for hotel in terms of cleanliness, then he shouldn't be recommended these hotels.

```
5.0    287484
4.0    125795
3.0     83929
2.0     34801
1.0     31962
Name: check in / front desk, dtype: int64
```

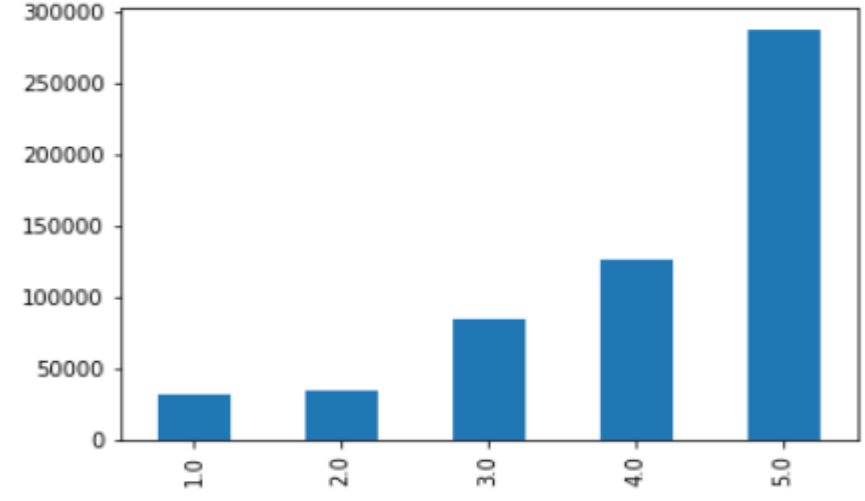


Figure 6: Rating based on Check in/ front desk

A Hotel Front Desk Agent is a professional who is responsible for greeting guests in an efficient and courteous manner. There behaviour is also an important factor in selecting a hotel. Figure 6, depicts the ratings in terms of behaviour of front desk agent at the time of check in.

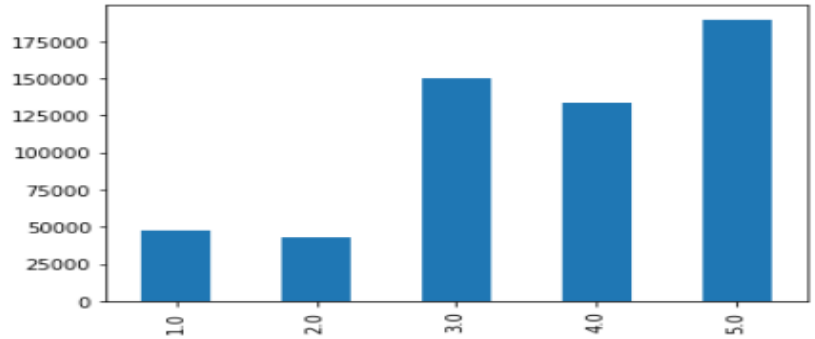
From figure 6, we can depict that there are some hotels where customers find the front desk agents' behaviour inappropriate in which they gave 1.0 and 2.0 ratings.

There are hotels which got 5.0 in terms of front desk agent behaviour, 50.94%.

The hotels which got 4.0 in terms of front desk agent behaviour are 22.30%.

Another important factor in deciding the best hotel is business service such as internet access. Hotel WIFI is a vital amenity for the guests. As such, it is important to have various hotel WIFI access points strategically located throughout your hotel. Figure 7, shows the rating

```
5.0    189649
3.0    150534
4.0    133736
1.0     47117
2.0     42672
Name: business service (e.g., internet access), dtype: int64
```



given by customers based on the business service provided by the hotel.

As we can see that there are many hotels that doesn't pay attention to these services as the customers have given low ratings to these hotels.

From figure 7, we can see that only 33.6% customers gave 5.0 rating, 23.7% customers gave 4.0 and 39.63% customers gave rating below 4.0 If someone is looking for a hotel that has internet access or WIFI service available, he should be recommended with hotel which are given 4.0 or 5.0 ratings.

As we can see in Figure 8, the length of sentences in each review given

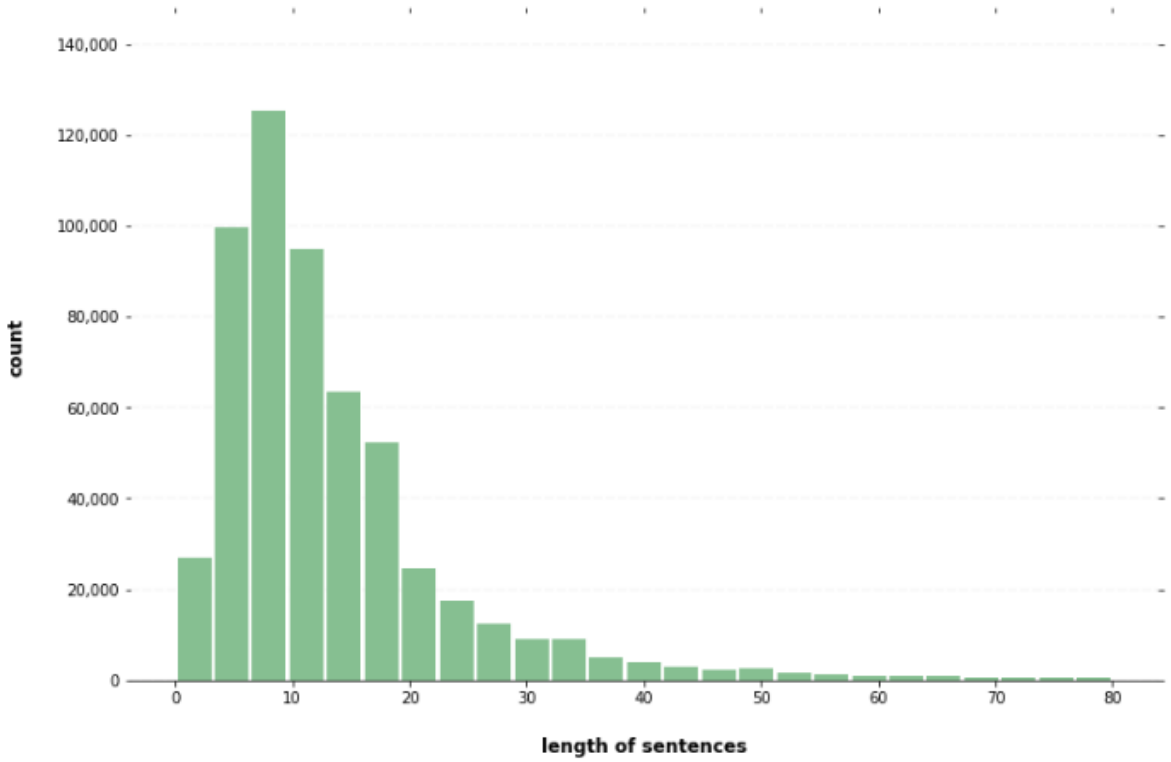


Figure 8: length of sentences in each review

As we can see that in most of the reviews are within the range of 20 sentences. There some guests whose reviews have more than 20 sentences.

Recommendation system using Alternating Least Square

Recommender systems predicts the user's interests and recommend product items that are quite likely interesting for them. They are among the most powerful machine learning systems that online retailers, e-commerce, travel, hotel websites etc. implement in order to drive sales.

Recommendation system has become an integral part of many online platforms given its use in increasing sales and customer retention. Amazon, Netflix, Facebook, LinkedIn and many more make use of recommendation system on their platform to recommend products, movies, people, posts respectively to users.

We will be using the Collaborative filtering technique in Pyspark for creating a recommendation system. Apache Spark ML implements alternating least squares (ALS) for collaborative filtering, algorithm for making recommendations.

Collaborative filtering:

Collaborative filtering is a method of making predictions (filtering) about the interests of a user by collecting the taste, likes, dislikes and various other information from many users.

For example, if a person A likes item 6, 7, 8 and B like 7, 8, 9 then they have similar interests and A should like item 9 and B should like item 6.

PySpark

PySpark is the Python API for Apache Spark, an open source, distributed computing framework and set of libraries for real-time, large-scale data processing.

So, we followed the following steps to build our recommendation system -

- We first installed the PySpark in our Google Colab notebook.
- Then we initialised a spark session which is the main entry point for Dataframe with the name Recommendations.

- After this we loaded our dataset named 'ratings' which contains a csv file named ALS

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName('Recommendations').getOrCreate()
```

in Apache Spark.

```
ratings = spark.read.csv("/content/drive/MyDrive/als/ALS.csv",header=True)
```

```
+---------+------+--------+
|author_id|rating|hotel_id|
+---------+------+--------+
|        0|   4.0|       0|
|        1|   2.0|       0|
|        2|   2.0|       1|
|        3|   2.0|       1|
|        4|   2.0|       1|
|        5|   1.0|       2|
|        6|   2.0|       2|
|        7|   2.0|       2|
|        8|   3.0|       2|
|        9|   4.0|       2|
+---------+------+--------+
only showing top 10 rows
```

And our dataset looked like this –

- After this we checked for the sparsity of the matrix which we are going to feed the ALS algorithm. The matrix was a very sparse matrix which had a sparsity of 99.998570540% which is normally seen in ALS data frame as the matrix contains rows which is basically a unique id i.e., 431504 in number, of the users who have reviewed

the hotels, and columns contains the unique id i.e., 91219 in number, for each hotel. We have a mapping for each user id and hotel id pair.

However, naturally not all the users would have visited all the hotels. Hence the matrix is very sparse.

The sparse matrix is then represented by the product of the user aspect matrix and item aspect matric.

- Aspects are the latent or hidden variables and it is an hyperparameter for ALS.

- Now, let's say that you were omniscient and you knew the absolute truth and you knew that in fact that the hotel ratings could be perfectly predicted by just 3 hidden factors, sex, age and income. In that case the 'rank' or 'aspects' of your run should be 3. Of course, you don't know how many underlying factors, if any, drive your data so you have to guess. The more you use, the better the results are up to a point.

Algorithm : *Alternating Least Squares (ALS)*

Procedure ALS (x_u ,y_i)
Initialization $x_u \leftarrow 0$
Initialization matrix y_i *with random values*
Repeat
    Fix y_i , *solve* x_u *by minimizing's the objective function (the sum of squared errors)*
    Fix y_i *solve* y_i *by minimizing the objective function similarly*
Until reaching the maximum iteration
Return x_u ,y_i
End procedure

Matrix factorization is the solution of this sparse matrix problem. There are two k dimensional vectors which are referred to as "factors".

– x_u is k dimensional vectors summarizing's every user u.

– y_i is k dimensional vectors summarizing's every item i.

Let,

$r_{ui} \approx x^T u Y_i$ (1)

$x_u = x_1, x_2, ..., x_n R_k$ (2)

$y_i = y_1, y_2, ..., y_n R_k$ (3)

•Equation 1 can be formulated as an optimization problem to find:

argminX rui $(r_{ui} - x^T u y_i) 2 + \lambda * (X_u k_{xu} k^2 + X_i k y_i k^2)$ (4)

Here, λ is the regularization factor, which is used to solve overfitting problem, is referred to as weighted-λ -regularization. The value of λ can be tuned to solve over-fitting whereas default value is 1. Let, the set of variables x_u is constant then the objective function of y_i is convex and then the set of variables yi is constant then the objective function of x_u is convex. Hence the optimize value of x_u and y_i can be found repeating the aforementioned approach until the convergence. This is called ALS (Alternating Least Squares).

- Importing the necessary modules.

```
# Import the required functions
from pyspark.ml.evaluation import RegressionEvaluator
from pyspark.ml.recommendation import ALS
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
```

- Regression Evaluator - Evaluator for Regression, which expects input columns prediction, label and an optional weight column.

- ALS - attempts to estimate the ratings matrix R as the product of two lower-rank matrices, X and Y, i.e. X * Y_t = R. Typically these approximations are called 'factor' matrices. The general approach is iterative. During each iteration, one of the factor matrices is held constant, while the other is solved for using least squares. The newly-solved factor matrix is then held constant while solving for the other factor matrix.

- ParamGridBuilder - for a param grid used in grid search-based model selection.

- K-fold cross validation performs model selection by splitting the dataset into a set of non-overlapping randomly partitioned folds which are used as separate training and test datasets e.g., with k=3 folds, K-fold cross validation will generate 3 (training, test) dataset

pairs, each of which uses 2/3 of the data for training and 1/3 for testing. Each fold is used as the test set exactly once.

• After this we have randomly split the dataset i.e., 80% for training and 20% for testing.

• Post this we created an ALS model whose inputs were author_id, hotel_id, rating columns.

• We also built a parameter grid which will be used to search the parameters for our best model which will have the lowest rmse.

- The parameters used were -

- 'als.rank' or hidden factors as 10, 20, 25, 50, 60, 70, 80, 100, 150, 200.

- 'regParam' or regularisation parameter as 0.05 and 0.1.

- 'Maxiter' or number of iterations to default 10.

- After this we used CrossValidator as 'cv' whose inputs are our ALS model, param_grid, evaluator and 3-fold cross validation.

```
# Build cross validation using CrossValidator
cv = CrossValidator(estimator=als, estimatorParamMaps=param_grid, evaluator=evaluator, numFolds=3)
```

- Then we fitted our model on the training data comprising of 80% of our original data and the best parameters received for our model are as follows-

```
<class 'pyspark.ml.recommendation.ALSModel'>
**Best Model**
  Rank: 150
  MaxIter: 10
  RegParam: 0.1
```

- Using the parameters of the best model, we will see that how our best model behaves on the test dataset which is 20% of our total data points

```
# View the predictions
test_predictions = best_model.transform(test)
RMSE = evaluator.evaluate(test_predictions)
print(RMSE)
```

```
1.8977240281576524
```

After that we have made predictions on the test data and the results are as shown below

author_id	rating	hotel_id	prediction
4	2.0	1	1.318638
17	3.0	3	2.6005383
79	4.0	16	2.577865
80	5.0	16	3.6243682
96	4.0	20	0.6976968
103	5.0	20	3.6473932
134	4.0	26	2.3668191
157	4.0	31	2.2867167
170	5.0	34	3.7201471
201	5.0	41	3.5378232
213	5.0	44	3.0247502
224	5.0	47	3.7104597
407	5.0	61	2.8062346
428	4.0	65	3.9634352
559	5.0	72	2.0168598
584	4.0	78	2.3852117
616	4.0	81	2.3601587
742	4.0	93	3.501234
764	4.0	96	3.6889563
763	5.0	96	3.2983668

only showing top 20 rows

Apache spark is well suited for applications which require high speed query of data, transformation and analytics results. Therefore, the recommendation system developed in this project is implemented on Apache Spark. Also, the matrix factorization using Alternating Least Squares (ALS) algorithm which is a type of collaborative filtering is used to solve overfitting issues in sparse data and increases prediction ac-curacy. The overfitting problem arises in the data as the user-item rating matrix is sparse. In this project a recommendation system for hotels using alternating least squares (ALS) matrix factorization method on Apache Spark is developed. The results shows that the RMSE value is significantly reduced using ALS matrix factorization method and the RMSE is 1.897. Consequently, it is shown that the ALS algorithm is suitable for training explicit feedback data set where users provide ratings for items. This will in turn bring enormous benefit to the hotel industry as they will be able to map the customer preferences before selection of a hotel room for stay.

Hotel Recommendation based on Similarities in Reviews

Content-based filtering:

For recommendation systems, especially content-based recommendation systems, similarity between recommended products is an important metric.

Recommendation systems work based on similarities between content or users accessing content.

There are several ways to measure the similarity between two items. The recommendation system uses this similarity matrix to recommend the closest similar products to the user.

Content-based filtering is a common technique in recommendation or recommender systems. The contents and features of things that you like are called “contents”.

Here the system uses your characteristics and preferences to recommend things you might like. We use information provided over the internet and information we can gather to curate our recommendations accordingly.

The purpose of content-based filtering is to categorize products using specific keywords, understand customer preferences, search a database for those terms, and recommend similar ones.

This type of recommendation system relies heavily on user input. Common examples include Google and Wikipedia.

Pairwise metrics, Affinities and Kernels:

There is no single right answer as to which similarity metric to choose. It depends on a number of factors, including the model chosen and the nature and distribution of the data. However, understanding how each similarity metric works from a geometric perspective makes the process of choosing a metric more transparent.

Nucleus is a measure of similarity. H. s(a, b) > s(a, c) if objects a and b are considered more "similar" than objects a and c. The kernel must also be positive and positive semidefinite.

This kernel is used to compute the similarity of documents represented as tf-idf vectors.

pairwise.pairwise_kernels allows you to compute kernels between X and Y using different kernel functions.

Cosine Similarity:

Cosine similarity is a metric, helpful in determining, how similar the data objects are irrespective of their size. In cosine similarity, data objects in a dataset are treated as a vector.

• The cosine similarity is beneficial because even if the two similar data objects are far apart by the Euclidean distance because of the size, they could still have a smaller angle between them. Smaller the angle, higher the similarity.

• When plotted on a multi-dimensional space, the cosine similarity captures the orientation (the angle) of the data objects and not the magnitude.

Linear Kernel:

The linear kernel used by scikit-learn's Gaussian process is provided as Dot_Product_kernel.

If you want the dot product of two vectors to check their similarity, you can use a linear kernel that returns the dot product of two vectors (with check for sparse input).

If x and y are column vectors, their linear kernels are:

$$k(x, y) = x^{\top} y$$

Methodology:

from sklearn.metrics.pairwise import linear_kernel

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.feature_extraction.text import TfidfVectorizer

TfIdfVectorizer – converts the text / BoW / corpus into tfidf matrix with relevance / importance of each term in all the documents.

The purpose of using tf-idf is to reduce the impact of tokens that occur very frequently in a given corpus. As such, it is empirically less informative than features that occur in a small fraction of the training corpus.

Using just the word count as a feature value for a word does not really reflect the meaning of that word in the document.

For example, if a word occurs frequently in all documents of a corpus, its count in different documents does not help distinguish between different documents.

On the other hand, if a word only appears in a few documents, its count in those documents helps distinguish it from the rest.

Therefore, the meaning of a word i.e. document feature values depend not only on the number of occurrences in the document, but also on its overall presence in the corpus.

This notion of word importance within a document is captured by a scheme known as the term frequency inverse document frequency (tf-idf) weighting scheme.

TfidfVectorizer(analyzer='word', ngram_range=(1, 3), min_df=0, stop_words='english')

"analyzer" - Whether the feature consists of word n-grams.

'ngram_range' - Lower and upper bounds of the range of n-values of the various n-grams to extract. All values of n where min_n <= n <= max_n are used. For example, an ngram_range of (1, 1) means unigrams only, (1, 2) means unigrams and bigrams, and (2, 2) means bigrams only.

Here we use a series of unigrams, bigrams and trigrams.

min_df - Minimum Document Frequency - When building the vocabulary, ignore terms whose document frequency is strictly below the specified threshold. This value is also called the cut-off in the literature. If the float is in the range [0,0,1,0], the parameter represents a document fraction (integer absolute count).

min_df is sometimes used to limit the vocabulary size so that only terms appearing in at least 10%, 20%, etc. of the documents are learned.

tf.fit_transform(df['pre_process'])

returns document-term matrix, with tf-idf values

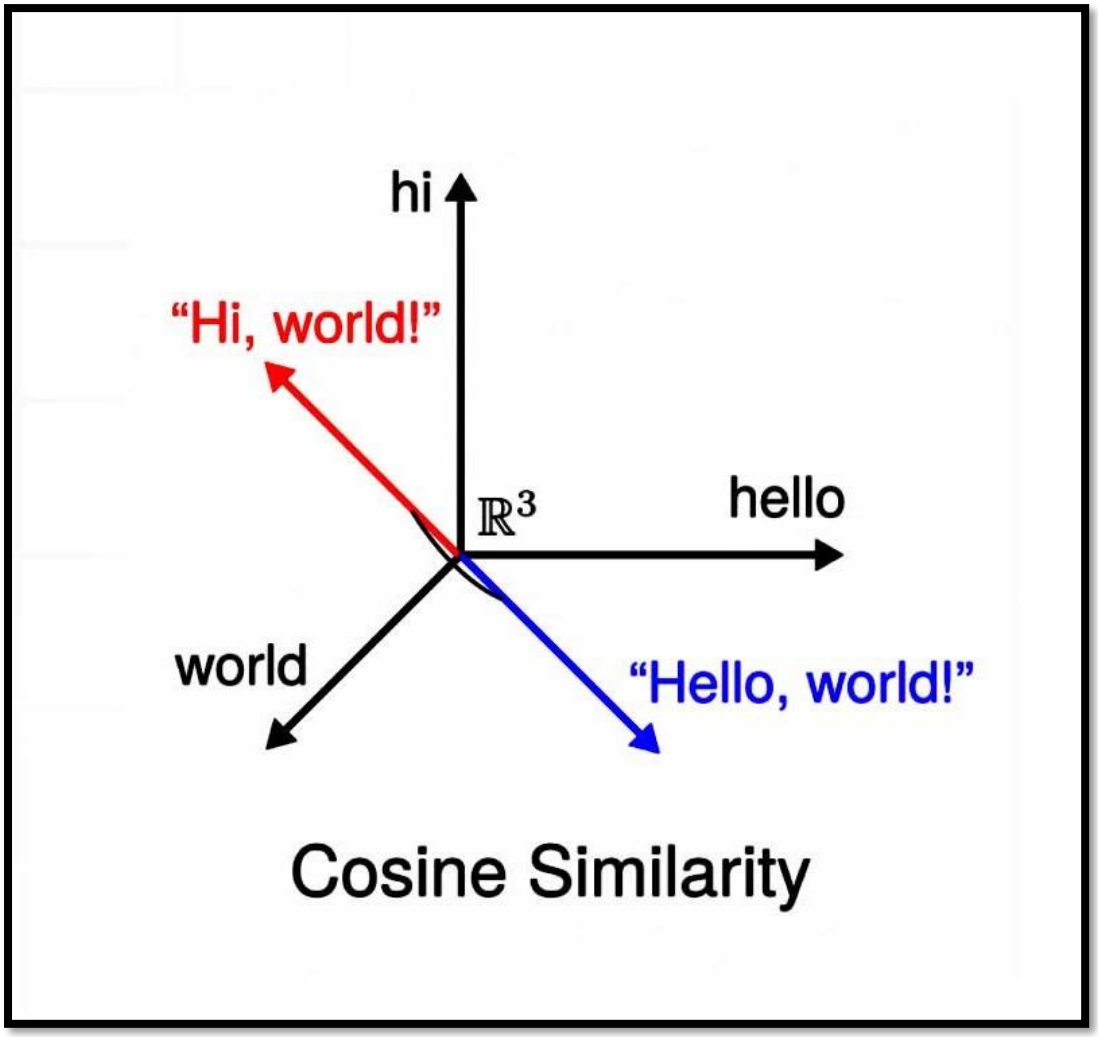
Applied on the text column containing reviews.

cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)

• Computes the linear kernel of the tf-idf matrix (formed above) by itself

• This returns an array of lists

• The inner product (linear kernel) is the 0 and Gives values between 1, these are taken from the tf-idf matrix and can be positive or negative based on the similarity/cosine similarity.



Finding the similarity quotient between the reviews based on orientation of the vectors or the structure of the text/reviews, how close or far apart they are.

creating a Series with the similarity scores in descending order

score_series = pd.Series(cosine_similarities[idx]).sort_values(ascending = False)

getting the indexes of the 51 most similar hotels except itself

top_51_indexes = list(score_series.iloc[1:51].index)

range as 1:51 due to repeating hotel names

populating the list with the names of the top 10 matching hotels

for i in top_51_indexes:

recommended_hotels.append(list(df.index)[i])

Picked the names of the top 10 hotels most similar to the given input name of the hotel, based on the highest similarity scores.

Input –

set(recommendations('GuestHouse Inn Bellingham-Bellingham Washington'))

{'Ambassador Hotel Zermatt-Zermatt Canton of Valais Swiss Alps',

'Baymont by Wyndham Galloway Atlantic City Area-Galloway New Jersey',

'Beijing Downtown Travelotel-Beijing',

'Ghost City Inn-Jerome Arizona',

'GuestHouse Inn Bellingham-Bellingham Washington',

'Holiday Inn Ontario Airport-Ontario California',

'Homewood Suites by Hilton Longview-Longview Texas',

'Inn at Mount Snow-Dover Vermont',

'Staymor Hotel-Blackpool Lancashire England',

'Venice Suites-Los Angeles California'}

Limitations:

We faced a lot of challenges to proceed in this project. Initial problem was with the data. The data was too large to work with colab notebook or Jupiter notebook. We mitigate the challenge by dividing the data into chunks and using colab pro.

Even the chunks were so huge we had to select the certain number of records from the chunks.

Even after merging all the selected records from each chunk, we could not proceed for topic modelling for the large dataset. We had to work with the chunks for topic modelling and recommendation based on similarity.

So, there were information loss. The limitations were with the infrastructure.

The memory and RAM were not sufficient to work with such huge data.

Future Scope:

UV matrix factorization:

We can build collaborative recommendation system by using UV matrix factorization with Tensor Flow.

We can use non-Negative matrix factorization to build collaborative recommendation system.

Hybrid Model:

We can build hybrid model, which is basically both collaborative and content-based recommendation system.

Topic based recommendation:

We had done topic modelling and based on the topics we can build recommendation system that can recommend hotels based on the topic searched by the user.

Reference:

https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/

https://towardsdatascience.com/text-preprocessing-in-natural-language-processing-using-python-6113ff5decd8

https://towardsdatascience.com/topic-modelling-f51e5ebfb40a#:~:text=Topic%20modeling%20refers%20to%20the,captured%20by%20those%20imaginary%20topics
.

https://towardsdatascience.com/build-recommendation-system-with-pyspark-using-alternating-least-squares-als-matrix-factorisation-ebe1ad2e7679

https://medium.com/@patelneha1495/recommendation-system-in-python-using-als-algorithm-and-apache-spark-27aca08eaab3

https://www.machinelearningplus.com/nlp/topic-modeling-gensim-python/

https://medium.com/analytics-vidhya/topic-modeling-using-gensim-lda-in-python-48eaa2344920

https://towardsdatascience.com/using-cosine-similarity-to-build-a-movie-recommendation-system-ae7f20842599

https://www.researchgate.net/publication/327842138_An_Improved_ALS_Recommendation_Model_Based_on_Apache_Spark_Second_International_Conference_ICSCS_2018_Kollam_India_April_19-20_2018_Revised_Selected_Papers

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0255-7