National College of Ireland

Configuration Manual

MSc Research Project
Programme Name

Shubham Garg
Student ID: x19205295

School of Computing
National College of Ireland

Supervisor: Martin Alain

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shubham Garg
Student ID:	x19205295
Programme:	Programme Name
Year:	2021
Module:	MSc Research Project
Supervisor:	Martin Alain
Submission Due Date:	16/12/2021
Project Title:	Configuration Manual
Word Count:	604
Page Count:	8

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:	Shubham Garg
Date:	31st January 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

Shubham Garg
x19205295

1 Introduction

This paper is the manual of configuration which enlists the all the requirements is needed to run the whole code of the project on the local machine environment. this manual also includes the screenshot of the hardware requirements as well as the software requirements. This file also includes the pictures of code of the Exploratory Data Analysis, Data loading , Data pre-processing, Data cleaning, Model applied which are Long Short Term Memory.

2 Environment

In this section of this module we have explained the requirements of hardware as well as the software and their version to re run the code on the local environment.

2.1 Hardware requirements

To perform research code we need hardware of minimum 11th generation of intel core i5 of version 1135G7 with the processor speed of 2.40GHz., 16GB RAM of DDR4 version with the speed of 3200 MHz, with the operating system of windows 11 of version 64 bit and the hard drive of 512GB SSD.Figure 3

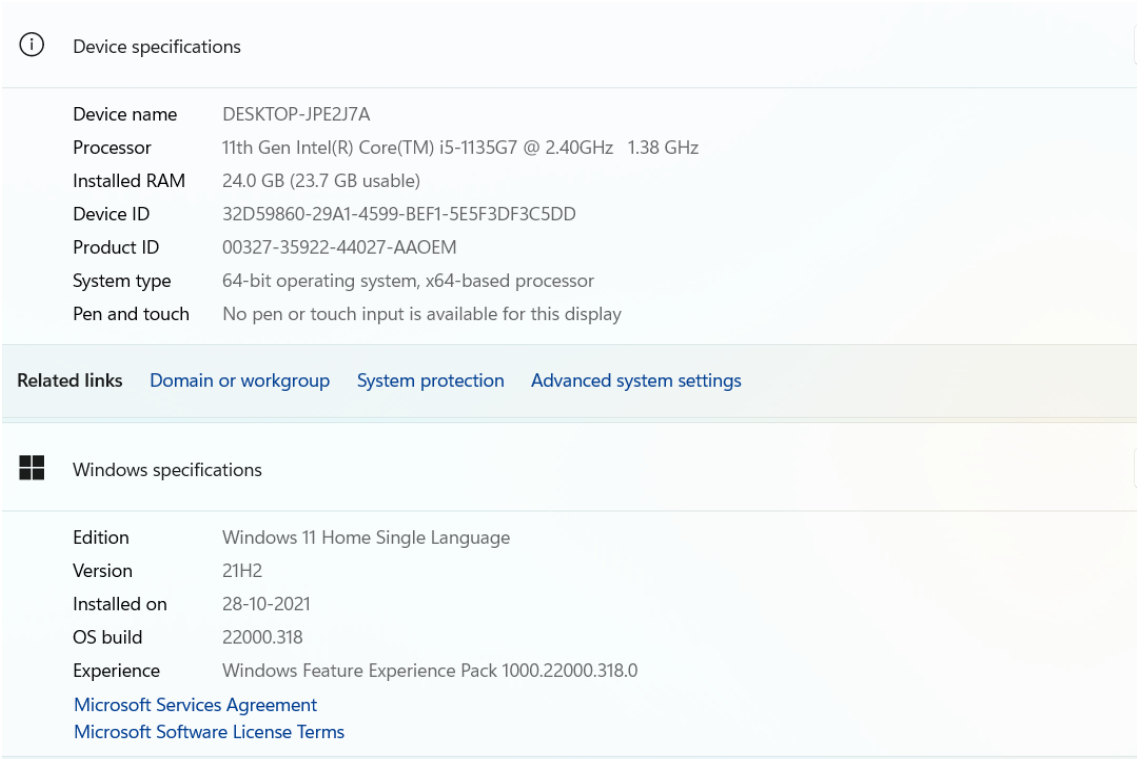

Figure 1: Hardware Requirements

2.2 Software Requirements

- Anaconda Navigator for Windows (Version 1.9.12)
- Jupyter Notebook (Version 6.0.3)
- Python (Version 3.9)

3 Data Collection

The data set for the research project is taken from Kaggle which is an open directory.
Link for dataset is `https://www.kaggle.com/eliasdabbas/flights-serps-and-landing-pages?select=flights_serp_scrape.csv`
`https://www.kaggle.com/eliasdabbas/search-engine-results-flights-tickets-keywords?select=flights_tickets_serp2019-08-15.csv` both are the same file but the second link has the data set which has been taken twice in every month from december 2018 to 1st of april 2020 what makes it the data set of 24 different csv with same script of code to scraping. it has 4000 rows of first 10 ranked pages for the same query. The rank has been changed every time.

4 Data Exploration

4.1 Importing The Libraries

There are many libraries which is required for processing the the code which has pre-defined functions shown below

```
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

Figure 2: Libraries

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from keras.callbacks import EarlyStopping
from keras.layers import Dropout
import re
import nltk
from nltk.corpus import stopwords
from nltk import word_tokenize
nltk.download('stopwords')
STOPWORDS = set(stopwords.words('english'))
from bs4 import BeautifulSoup
import plotly.graph_objs as go
#import plotly.plotly as py
import cufflinks
from IPython.core.interactiveshell import InteractiveShell
import plotly.figure_factory as ff
InteractiveShell.ast_node_interactivity = 'all'
from plotly.offline import iplot
cufflinks.go_offline()
cufflinks.set_config_file(world_readable=True, theme='pearl')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Figure 3: Libraries

4.2 Exploratory Data Analysis

4.2.1 Data description

	rank	totalResults	count	startIndex	searchTime	formattedSearchTime
count	4000.00000	4.000000e+03	4000.0	4000.0	4000.000000	4000.000000
mean	5.50000	5.538642e+07	10.0	1.0	0.321426	0.321350
std	2.87264	8.004667e+07	0.0	0.0	0.061791	0.062045
min	1.00000	9.940000e+04	10.0	1.0	0.162895	0.160000
25%	3.00000	9.217500e+06	10.0	1.0	0.278504	0.280000
50%	5.50000	2.300000e+07	10.0	1.0	0.314640	0.310000
75%	8.00000	7.097500e+07	10.0	1.0	0.354942	0.352500
max	10.00000	5.420000e+08	10.0	1.0	0.533649	0.530000

Figure 4: DATA DESCRIPTION

4.2.2 checking Null values

This code will check the null or Nan values in the dataset.

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(10,6))
sns.heatmap(df.isnull())
plt.title("Missing values?",fontsize = 16)
plt.show()
```

```
<Figure size 720x432 with 0 Axes><matplotlib.axes._subplots.AxesSubplot at 0x7f54f5c08790>Text(0.5, 1.0, 'Missing values
```

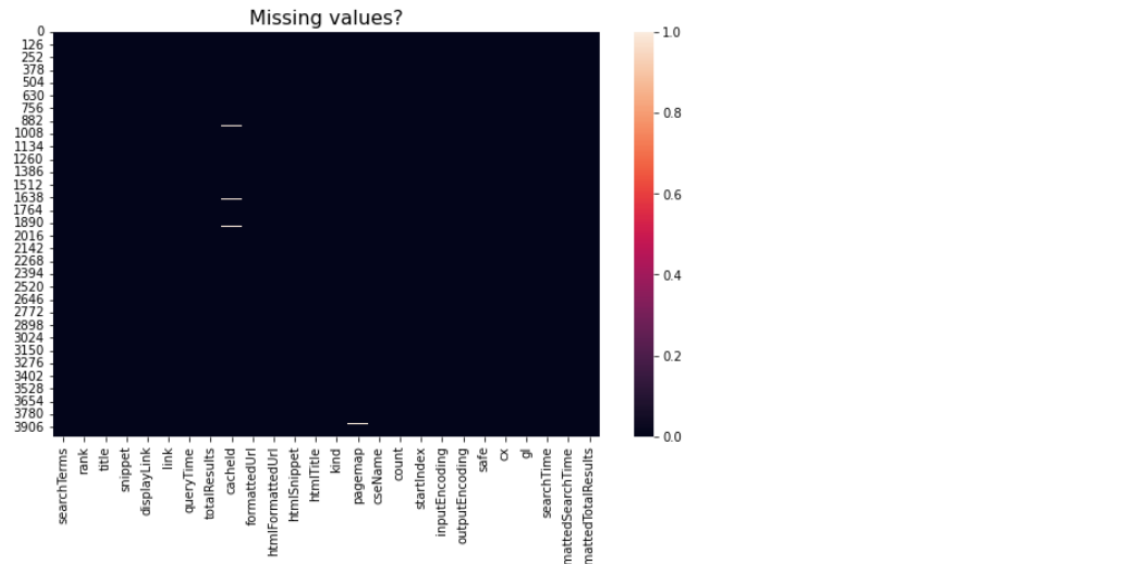

Figure 5: Null values

4.2.3 Percentage of traffic by rank

This is percentage of traffic on rank based regardless the websites.

```
traffic = {1: 0.33,
           2: 0.18,
           3: 0.12,
           4: 0.08,
           5: 0.06,
           6: 0.05,
           7: 0.04,
           8: 0.03,
           9: 0.02,
           10: 0.02}

# Display the percentage of traffic with colors
colors = []
for i in range(10):
    xc = round(0.7-0.05*i,2)
    c = (xc,xc,0.5)
    colors.append(c)
colors = colors + ["red"]

x = [str(t) for t in range(1,11)] +[">10"]
y = [traffic[key]*100 for key in traffic.keys()]
y += [100-sum(y)]

plt.figure(figsize=(8,6))
plt.bar(x,y, color = colors)
plt.title("Estimation of percentage of traffic by ranking\nin Google search results", fontsize = 16)
plt.xticks(x)
plt.xlabel("# rank")
plt.ylabel("% of traffic")
plt.show()
```

Figure 6: Percentage of traffic by rank

4.2.4 Percentage of traffic by websites

This code will give the percentage of traffic on the websites regardless the rank.

```
# Select the columns we will use
df = df[["searchTerms", "rank", "title", "snippet", "displayLink"]].copy()

# Calculate the percentage of the total traffic by website
df["traffic%"] = df["rank"].map(traffic)
byTraffic = pd.pivot_table(df, values = "traffic%", index = "displayLink", aggfunc = "sum").sort_values("traffic%", ascending = False)
nb_terms = df["searchTerms"].nunique()

byTraffic["traffic%"] = byTraffic["traffic%"].apply(lambda x: (100/2) * (x/nb_terms))

# Display the percentage of the total traffic by website
# Supposing that each term has an equal amount of traffic
# what isn't true. It is the only way get an estimation,
# because the dataset doesn't have the amount of traffic
# by search term.

# number of top websites to plot
nb_website = 20

# Display the result with colors
colors = []
for i in range(nb_website):
    x = round(0.7-0.02*i,2)
    c = (x,x,0.5)
    colors.append(c)

plt.figure(figsize=(10,6))
plt.title(f"Percentage of traffic on {nb_terms} search terms by website", fontsize = 16)
byTraffic["traffic%"].iloc[:nb_website].plot.bar(color = colors)
plt.xlabel("")
plt.ylabel("% of total traffic")
plt.show()
```

Figure 7: Percentage of traffic by websites

4.2.5 keyword concentration in titles

This code will get the percentage of the titles over the rank same code will be used for snippet column to know the concentration of key words in snippet.

```
# Calculate the % of keywords/search terms in the titles
df["%search_term_in_title"] = df["searchTerms"].apply(lambda x: len(x.split(" "))) / df["title"].apply(lambda x: len(x.split(" ")))
df["%search_term_in_title"] = 100 * df["%search_term_in_title"] # Convert the result in %

# Display the result
proc_searchterm_rank = pd.pivot_table(df, values = "%search_term_in_title", index = "rank", aggfunc = "mean").sort_index(ascending = False)
proc_searchterm_rank.plot.barh(figsize = (8,5), color = (0.32, 0.32, 0.5))
plt.legend("")
plt.xlabel("Exact keyword concentration in %", fontsize = 12)
plt.ylabel("# rank", fontsize = 14)
plt.title("Average keyword concentration in titles\nby rank", fontsize = 16)
```

Figure 8: keyword concentration in titles

4.2.6 concatenating all data

This function concatenate the all the data sets and formed one data set and make the new data frame.

```
# Import and merge all the csv files
lst_df_path = []
for dirname, _, filenames in os.walk('New_folder'):
    for filename in filenames:
        lst_df_path.append(os.path.join(dirname, filename))

# List the dates of the csv files
# Be aware that some months are missing
sorted([d[-14:-4] for d in lst_df_path ])
```

Figure 9: concatenating all data

4.2.7 Most visitors on websites over the time

this code helps to know the most visited websites as the ranks changes over the time so what will be the effect of it on the traffic of websites.

```
# Charge all the datasets
lst_df = []
for d in range(len(lst_df_path)):
    lst_df.append(pd.read_csv(lst_df_path[d]))

# Merge all the datasets
all_df = pd.concat(lst_df)
# Reset index
all_df = all_df.reset_index().drop("index",axis = 1)
# Cut the year and the month of the date
# 2019-03-25 12:25:42.277534+00:00 -> 2019-03
# Convert to datetime: datetime will automatically take the first day of each month
# 2019-03 -> 2019-03-01
all_df["queryTime"] = pd.to_datetime(all_df["queryTime"].apply(lambda x: x[:7]))
# Select the columns we will use
all_df = all_df[["searchTerms","rank","displayLink","queryTime"]].copy()
# Calculate the percentage of the total traffic by website
all_df["traffic"] = all_df["rank"].map(traffic)
avg_rank_bymonth = pd.pivot_table(all_df,
                                  values = "traffic",
                                  index = "displayLink",
                                  columns="queryTime",
                                  aggfunc="sum")

# Select the top 4 websites
avg_rank_bymonth = avg_rank_bymonth.loc[top4]
# Ajust the values to be in percentage
avg_rank_bymonth = avg_rank_bymonth*(100/2)/nb_terms
# For an unknown reason the values of some month are
# multiplicated by two
# Use an ad-hoc solution to fix it, dividing those
# values by two (if you find the issue, write me please)
min_avg = avg_rank_bymonth.sum(axis=0).min()
sum_avg = avg_rank_bymonth.sum(axis=0)
idx_change = sum_avg[sum_avg >= 2*min_avg].index
avg_rank_bymonth[idx_change] /= 2

# Transpose the dataframe to be able to plot it well
avg_rank_bymonth = avg_rank_bymonth.transpose()
# Rename the columns: keep skyscanner.com -> skyscanner
new_column_names = [c.split('.')[1] for c in avg_rank_bymonth.columns]
avg_rank_bymonth.columns = new_column_names

# Display the results
# Be aware that some months are missing
# even if it isn't obvious on the graphic
avg_rank_bymonth.plot(figsize = (12,8), lw = 5)
plt.title('% of the traffic for each website',fontsize = 16)
plt.ylabel('%')
plt.ylim(0, avg_rank_bymonth.max().max()+2)
plt.xlabel('')
plt.legend(fontsize = 14)
plt.show()
```

Figure 10: Most visitors on websites over the time

5 Pre Processing

5.0.1 Removing the impurities and stop words

this code help to remove all the symbols , numeric values, stop words etc. that menns it will remove everything except the text words.

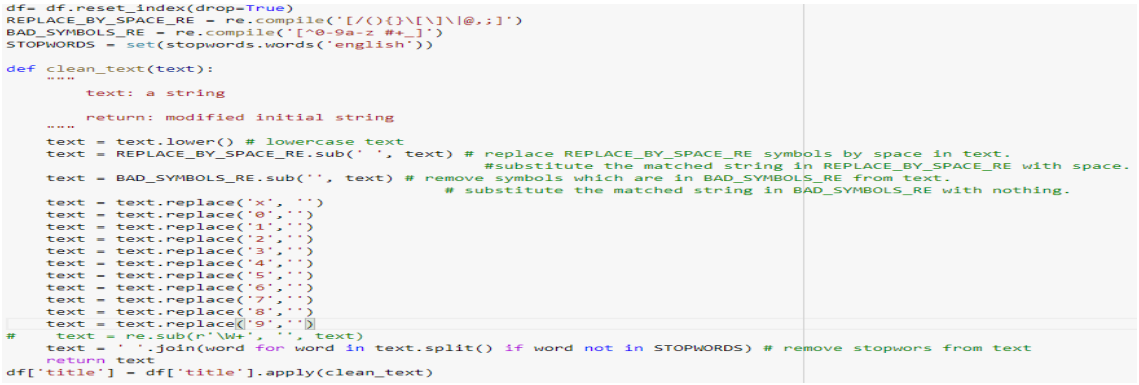

Figure 11: Removing the impurities and stop words

5.1 tokenizing the words

5.1.1 tokenizer function

Figure 12: tokenizer function

5.1.2 tokenizing the words in the data frame

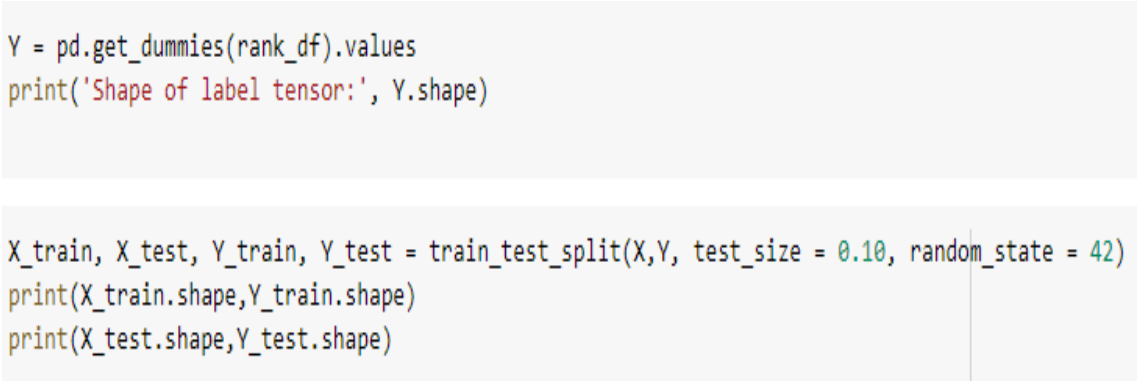

Figure 13: tokenizing the words in the data frame

5.2 splitting into test and train

This code split the data into the ratio of 90 and 10 in train and test data respectively

```
Y = pd.get_dummies(rank_df).values
print('Shape of label tensor:', Y.shape)
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.10, random_state = 42)
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

Figure 14: splitting into test and train

6 Modeling

this is the model which is applied on the data set for prediction the target variable "RANK" this model is then changed by adding and deleting the dense layer and by changing the weights.

```
model = Sequential()
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=X.shape[1]))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(10, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

Figure 15: modeling

Here the model has been fitted and will provide the result with the epochs history.

```
epochs = 50
batch_size = 25

history = model.fit(X_train, Y_train, epochs=epochs, batch_size=batch_size,validation_split=0.1,callbacks=[EarlyStopping(monitor='val_loss', patience=6, min_delta=0.0001)])
```

Figure 16: fitting the model