National College of Ireland

Configuration Manual

MSc Research Project

Data Analytics

Zahra Fathima Sanaullah Shariff

Student ID: x20221207

School of Computing
National College of
Ireland

Supervisor: Christian Horn

National College of Ireland Project Submission Sheet School of Computing

Student Name:	Zahra Fathima Sanaullah Shariff
Student ID:	X20221207
Programme:	MSc in Data Analytics
Year:	2022
Module:	MSc Research Project
Supervisor:	Christian Horn
Submission Due Date:	15/08/2022
Project Title:	Configuration Manual
Word Count:	316
Page Count:	9

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:	Zahra Fathima S
Date:	15th August 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	Q
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	Q
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	Q

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

Product Matching for E-commerce Platform based on Text and Image Similarity using Deep Neural Network Architecture

1 Introduction

The configuration manual provides details about the components necessary for the implementation of the project to identify product matching using text and image similarity with deep neural networks. Brief steps are provided in this document to ensure the research project is implemented successfully.

2 System Configuration

For implementation, the following configuration and hardware requirements are required:

2.1 Requirement for Hardware

Hardware	Configurations
System	HP Intel Core i5
Operating System	Windows 10
RAM	12 GB
Hard Disk	15 GB
Processor	Intel(R) Core (TM) i5-6200U CPU @ 2.30GHz 2.40 GHz

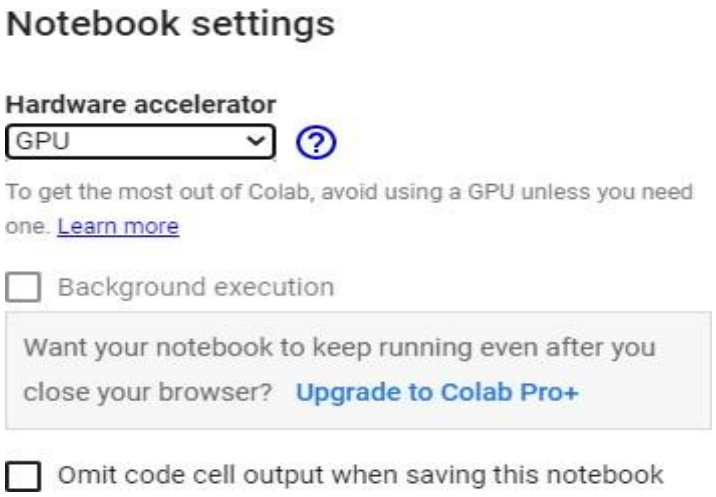
Figure 1 Hardware Details

2.2 Requirement for Software

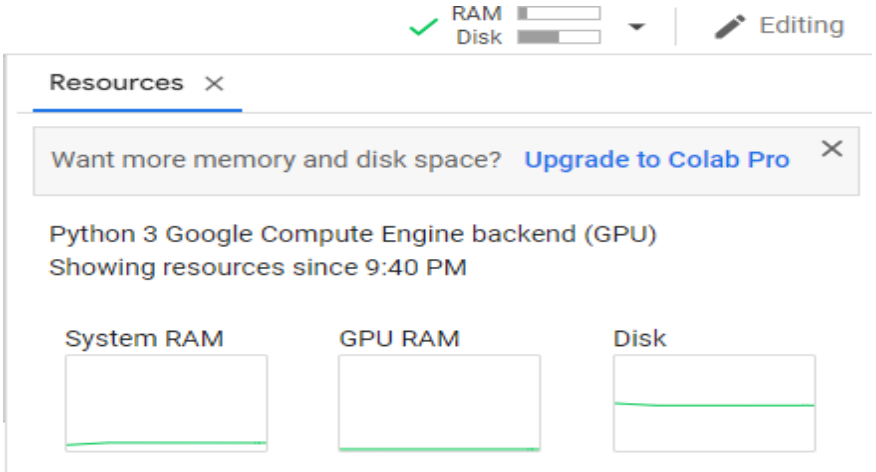
The details used for windows 10 operating system is given as below

Figure 2 Operating System Details

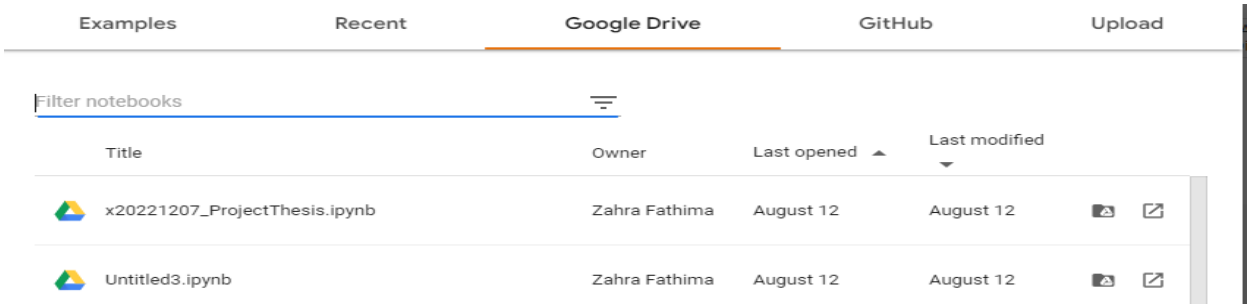
- The hardware accelerator is selected as GPU in Google Colab.



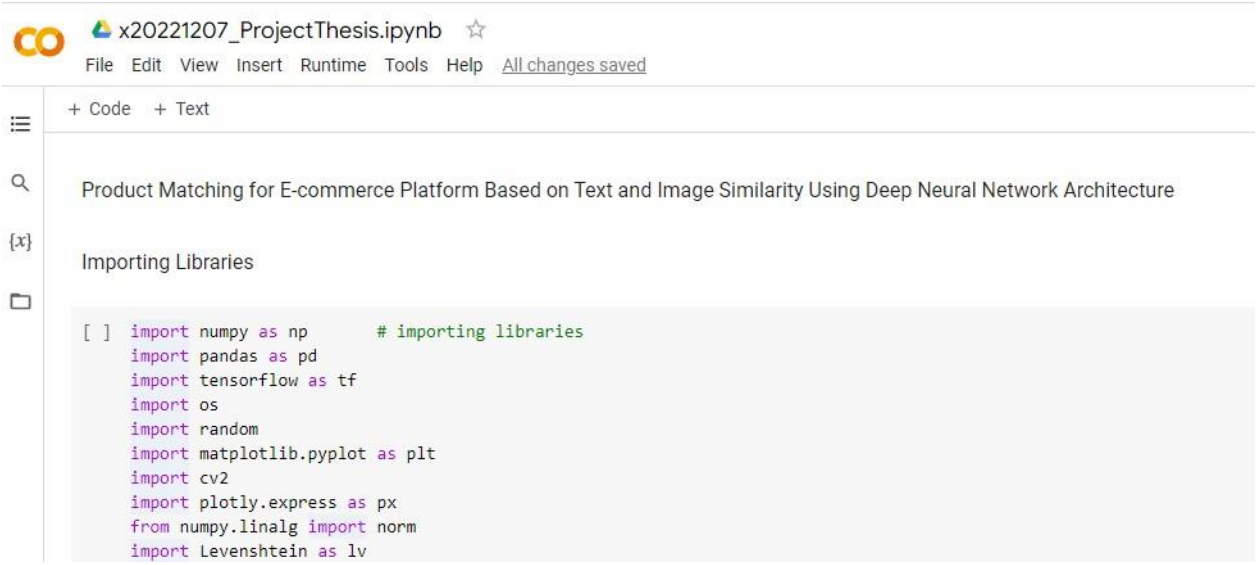
- In google colab, the disk, RAM and GPU RAM is provided below.



- Google drive is connected to Google colab storage.



- ipynb is the format of the notebook used to run the code



3 Project Implementation

3.1 Data Extraction

From Google drive location, the data is extracted.

3.2 Installing Required Libraries

```
import numpy as np       # importing libraries
import pandas as pd
import tensorflow as tf
import os
import random
import matplotlib.pyplot as plt
import cv2
import plotly.express as px
from numpy.linalg import norm
import Levenshtein as lv
```

3.3 Data Preparation

```
train_df = pd.read_csv('../input/shopee-product-matching/train.csv')    # reading train csv file
```

```
train_df.head(10)       # visualizing first ten rows of the data
```

```
print(len(train_df))   # calculating number of rows
```

```
34250
```

```
print(len(train_df['label_group'].unique()))    # calculating number of rows having unique label group
```

```
11014
```

```
labelGroup_df = train_df.groupby('label_group')    # assigning grouped data to new variable
```

Plotting images grouped by label group

```
%matplotlib inline
groupCount = 0
for groupName,groupDf in labelGroup_df:
    print(groupName)
    imgCount=0
    for index,row in groupDf.iterrows():
        print(row['title'])
        imagePath = '../input/shopee-product-matching/train_images/'+row['image']
        pil_im = cv2.imread(imagePath)
        plt.figure()
        plt.imshow(pil_im)
        plt.show()
        imgCount= imgCount+1
        if (imgCount==3):
            break

    groupCount = groupCount +1
    if (groupCount==10):
        break
```

3.4 Exploratory Data Analysis

An exploratory data analysis is any analysis that employs summary statistics and graphs to evaluate data in order to look for patterns, identify anomalies, test hypotheses, and confirm assumptions.

EDA

```
def plot(num):
    IMG_PATHS = "../input/shopee-product-matching/train_images/"
    sq_num = np.sqrt(num)
    assert sq_num == int(sq_num), "Number of Images must be a perfect Square!"

    sq_num = int(sq_num)
    image_ids = os.listdir(IMG_PATHS)
    random.shuffle(image_ids)
    fig, ax = plt.subplots(nrows=sq_num, ncols=sq_num, figsize=(10, 10))

    for i in range(sq_num):
        for j in range(sq_num):
            idx = i*sq_num + j
            ax[i, j].axis('off')
            img = cv2.imread(IMG_PATHS + '/' + image_ids[idx])
            img = img[:, :, ::-1]
            ax[i, j].imshow(img); ax[i, j].set_title(f'{image_ids[idx]}', fontsize=6.5)

    plt.show()
```

```
def plot_from_label(group):
    IMG_PATHS = "../input/shopee-product-matching/train_images/"
    image_list = train_df[train_df['label_group'] == group]
    image_list = image_list['image'].tolist()
    num = len(image_list)

    sq_num = np.sqrt(num)

    sq_num = int(sq_num)
    image_ids = os.listdir(IMG_PATHS)
    random.shuffle(image_ids)
    fig, ax = plt.subplots(nrows=sq_num, ncols=sq_num, figsize=(10, 10))

    path = [os.path.join(IMG_PATHS, x) for x in image_list]

    for i in range(sq_num):
        for j in range(sq_num):
            idx = i*sq_num + j
            ax[i, j].axis('off')
            img = cv2.imread(path[idx])
            img = img[:, :, ::-1]
            ax[i, j].imshow(img)

    plt.show()
```

```
def plot_from_title(title):
    IMG_PATHS = "../input/shopee-product-matching/train_images/"
    image_list = train_df[train_df['title'] == title]
    image_list = image_list['image'].tolist()
    num = len(image_list)

    sq_num = np.sqrt(num)
    sq_num = int(sq_num)

    image_ids = os.listdir(IMG_PATHS)
    random.shuffle(image_ids)
    fig, ax = plt.subplots(nrows=sq_num, ncols=sq_num, figsize=(10, 10))
    fig.suptitle(f"Product Name: {title}")
    path = [os.path.join(IMG_PATHS, x) for x in image_list]

    for i in range(sq_num):
        for j in range(sq_num):
            idx = i*sq_num + j
            ax[i, j].axis('off')
            img = cv2.imread(path[idx])
            img = img[:, :, ::-1]
            ax[i, j].imshow(img)

    plt.show()
```

Plotting Products Naively Let's start Image EDA by plotting the products just randomly from the dataset.

```
# Plot 16 random images
plot(16)
```

Plotting Products based on Image Label Group Let's take a little smarter approach by plotting products based on their label group.

Image Label Group: 1141798720 , 994676122

```
plot_from_label(1141798720)
```

```
plot_from_label(994676122)
```

Product Images with Same Name Now let's see some product images that have same name (title).

This will help us see how the images with same title can be different from each other.

Title: Koko syubbanul muslimin koko azzahir koko baju , Monde Boromon Cookies 1 tahun+ 120gr

```
plot_from_title("Koko syubbanul muslimin koko azzahir koko baju")
```

Product Name: Koko syubbanul muslimin koko azzahir koko baju

Top-15 Image Label Groups Let's see the top-15 image label groups (by the number of images) in this dataset.

```
import seaborn as sns
sns.set_palette("tab20")
top10_names = train_df['label_group'].value_counts().index.tolist()[:15]
top10_values = train_df['label_group'].value_counts().tolist()[:15]

plt.figure(figsize=(10, 10))
sns.barplot(x=top10_names, y=top10_values)
plt.xticks(rotation=45)
plt.xlabel("Label Group")
plt.ylabel("Image Count")
plt.title("Top-15 Label Groups by Image Count")
plt.show()
```

Top-5 Products from Images¶ Let's see the top-5 products (by count of titles) in this dataset using their provided images.

```
top5_products = train_df['title'].value_counts()[:5].index.tolist()
for title in top5_products:
    plot_from_title(title)
```

3.5 Data Preprocessing

Image Embedding:

A dense vector representation of an image is called an image embedding, and it is used for many tasks, such as classification, by using it as a lower-dimensional representation of the image.

```
def get_imageEmbeddings(model,imagePath):    # funtion for extracting image embeddings
    image = tf.keras.preprocessing.image.load_img(imagePath,target_size= size)  # pre-processing images
    input_arr = tf.keras.preprocessing.image.img_to_array(image)  # getting images into array
    input_arr = np.array([input_arr])
    img_embeddings = model(input_arr)      # calculating mean of image embeddings
    meanImgEmb1 = np.mean(img_embeddings,axis =0)
    meanImgEmb2 = np.mean(meanImgEmb1,axis=0)
    meanImgEmb = np.mean(meanImgEmb2,axis=0)
    return meanImgEmb
```

```
data_hub = {}
all_images = X_train["image"].values
all_text = X_train["title"].values
for img_name,text_meta in zip(all_images,all_text):
    img_path = "../input/shopee-product-matching/train_images/{}".format(img_name)

    data_hub[img_name] = {}
    data_hub[img_name]["mobilenet"] = get_imageEmbeddings(mobilenet,img_path)
    data_hub[img_name]["vgg"] = get_imageEmbeddings(vgg,img_path)
    data_hub[img_name]["resnet"] = get_imageEmbeddings(res,img_path)

    data_hub[img_name]["text"] = text_meta
```

3.6 Models

The models used are VGG-19, MobileNet-V2, ResNet-50. Cosine similarity is calculated for images and Levenshtein method is used for text to calculate similarity in all the models, and custom metric is calculated using the cosine and levenshtein method.

```
cosine_values_mobilenet = np.zeros([len(data_hub.keys())])
cosine_values_vgg = np.zeros([len(data_hub.keys())])
cosine_values_res = np.zeros([len(data_hub.keys())])

name_of_image = random_5_names[0]
for i,name in enumerate(data_hub.keys()):
    if name != name_of_image:
        cosine_values_mobilenet[i] = cosine_similarity(data_hub[name_of_image]["mobilenet"],data_hub[name]["mobilenet"])
        cosine_values_vgg[i] = cosine_similarity(data_hub[name_of_image]["vgg"],data_hub[name]["vgg"])
        cosine_values_res[i] = cosine_similarity(data_hub[name_of_image]["resnet"],data_hub[name]["resnet"])
    else:
        cosine_values_mobilenet[i]=np.inf
        cosine_values_vgg[i] =np.inf
        cosine_values_res[i] = np.inf
```

```
original_text = data_hub[name_of_image]["text"]
mobilenet_pred_text = []
vgg_pred_text = []
res_pred_text = []

temp = np.array(list(data_hub.keys()))
for i,j,k in zip(temp[mobilenet_closest], temp[vgg_closest],temp[res_closest]):
    mobilenet_pred_text.append(data_hub[i]["text"])
    vgg_pred_text.append(data_hub[j]["text"])
    res_pred_text.append(data_hub[k]["text"])

mobilenet_pred_le = []
vgg_pred_le = []
res_pred_le = []

for i in range(5):
    mobilenet_pred_le.append(lv.distance(original_text,mobilenet_pred_text[i]))
    vgg_pred_le.append(lv.distance(original_text,vgg_pred_text[i]))
    res_pred_le.append(lv.distance(original_text,res_pred_text[i]))
```

```
plt.figure(figsize=(15,5))
plt.subplot(1,3,1)
plt.bar(height=mobilenet_pred_le,x=range(5),color="r")
plt.plot([np.mean(mobilenet_pred_le) for i in range(5)],color="black")
plt.title("Mobilenet")

plt.subplot(1,3,2)
plt.bar(height=vgg_pred_le,x=range(5),color="g")
plt.plot([np.mean(vgg_pred_le) for i in range(5)],color="black")
plt.title("VGG19")

plt.subplot(1,3,3)
plt.bar(height=res_pred_le,x=range(5),color="b")
plt.plot([np.mean(res_pred_le) for i in range(5)],color="black")
plt.title("ResNet50")

plt.suptitle("Similarity Comparison ( Levenshtein Method Text)")
plt.show()
```

```
mb_cos_d = np.mean(cosine_values_mobilenet[mobilenet_closest])
print(mb_cos_d)
vgg_cos_d = np.mean(cosine_values_vgg[vgg_closest])
print(vgg_cos_d)
res_cos_d = np.mean(cosine_values_res[res_closest])
print(res_cos_d)
mb_le_d = np.mean(mobilenet_pred_le)/100
print(mb_le_d)
vgg_le_d = np.mean(vgg_pred_le)/100
print(vgg_le_d)
res_le_d = np.mean(res_pred_le)/100
print(res_le_d)
image_weight = 0.6
text_weight = 0.4
mb_custom1 = 0.6*(mb_cos_d)+0.4*(mb_le_d)
vgg_custom1 = 0.6*(vgg_cos_d)+0.4*(vgg_le_d)
res_custom1 = 0.6*(res_cos_d)+0.4*(res_le_d)
print("custom metric value of mobilenet model is ",mb_custom1 )
print("custom metric value of VGG-19 model is ",vgg_custom1 )
print("custom metric value of ResNet-50 model is ",res_custom1 )
```

Model Comparison:

```
data = {'Avg_cosine_sim':[mb_cos_d,vgg_cos_d,res_cos_d], # creating data contains metrics of evaluation
        'Avg_lev_dis':[mb_le_d,vgg_le_d,res_le_d],
        'Custom Metric':[mb_custom1,vgg_custom1,res_custom1],
        'Algo':['Mobilenet','Vgg19','ResNet50'] }
df = pd.DataFrame(data)
```

The name of the python notebook file is x20221207_ProjectThesis.ipynb