CSE508 Information Retrieval
Assignment-2
Megha 2021337

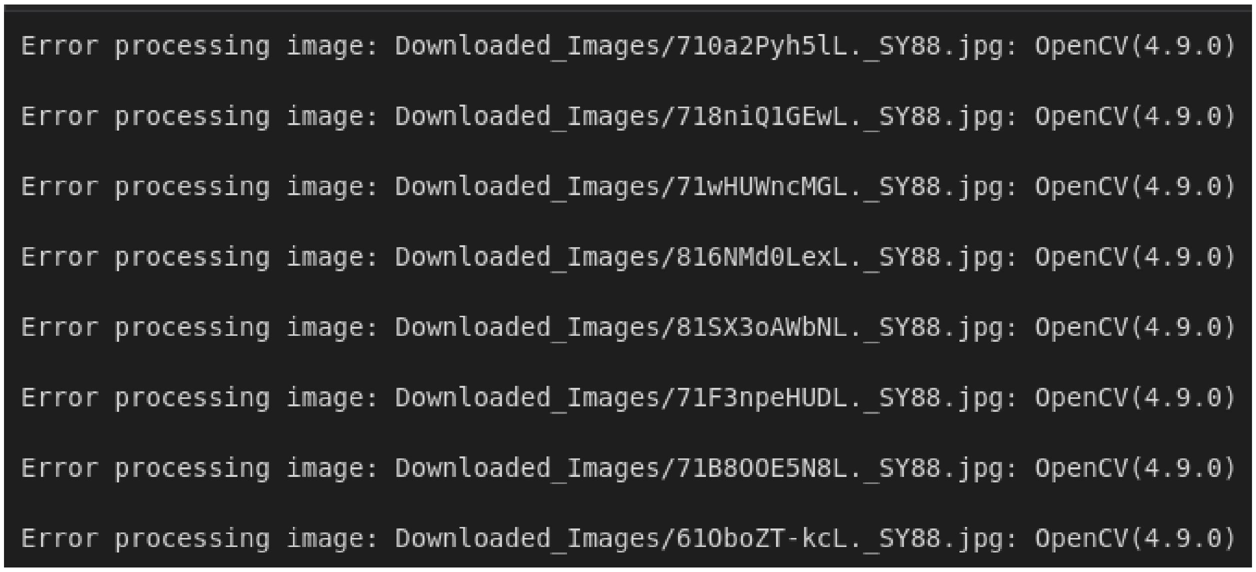
1. Image Feature Extraction [25]

Step 1 : Downloaded all the images from A2_Data.csv and saved them in a new folder 'Downloaded_Images'. Then created a new .csv file and modified the image column.

```
df.to_csv('A2_Downloaded_Images.csv', index=False)
```

Step 2 : Basic Pre-processing
Vgg16 uses 224x224 image size.
Convert all Images to RGB and replace the old images with new processed ones.

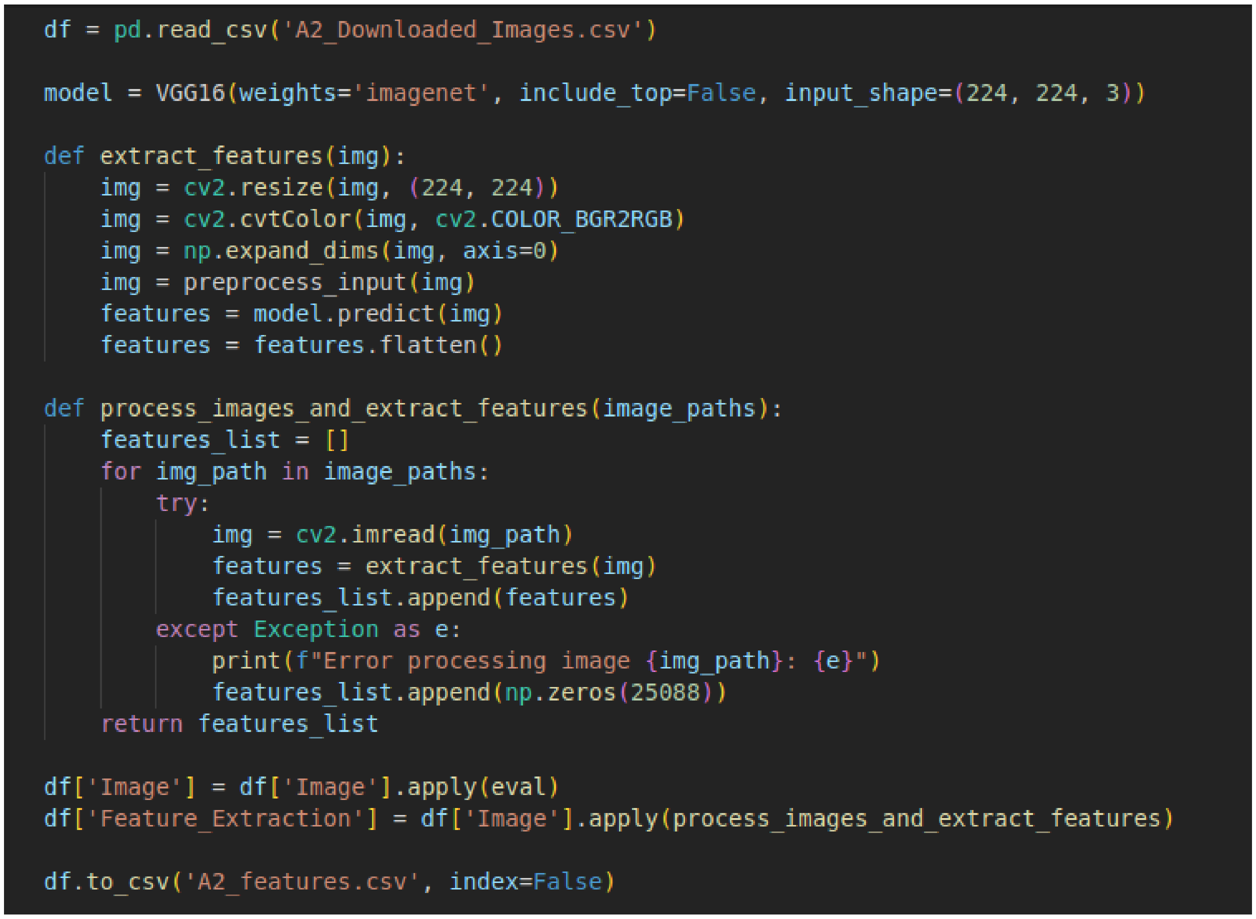
Images that are not found are :



```
Error processing image: Downloaded_Images/710a2Pyh5lL._SY88.jpg: OpenCV(4.9.0)

Error processing image: Downloaded_Images/718niQ1GEwL._SY88.jpg: OpenCV(4.9.0)

Error processing image: Downloaded_Images/71wHUWncMGL._SY88.jpg: OpenCV(4.9.0)

Error processing image: Downloaded_Images/816NMd0LexL._SY88.jpg: OpenCV(4.9.0)

Error processing image: Downloaded_Images/81SX3oAWbNL._SY88.jpg: OpenCV(4.9.0)

Error processing image: Downloaded_Images/71F3npeHUDL._SY88.jpg: OpenCV(4.9.0)

Error processing image: Downloaded_Images/71B8OOE5N8L._SY88.jpg: OpenCV(4.9.0)

Error processing image: Downloaded_Images/61OboZT-kcL._SY88.jpg: OpenCV(4.9.0)
```

Going to '**A2_Downloaded_Images.csv**' and dropping the rows having these images.

Step 3 : Extracting Features using VGG16 convolutional network

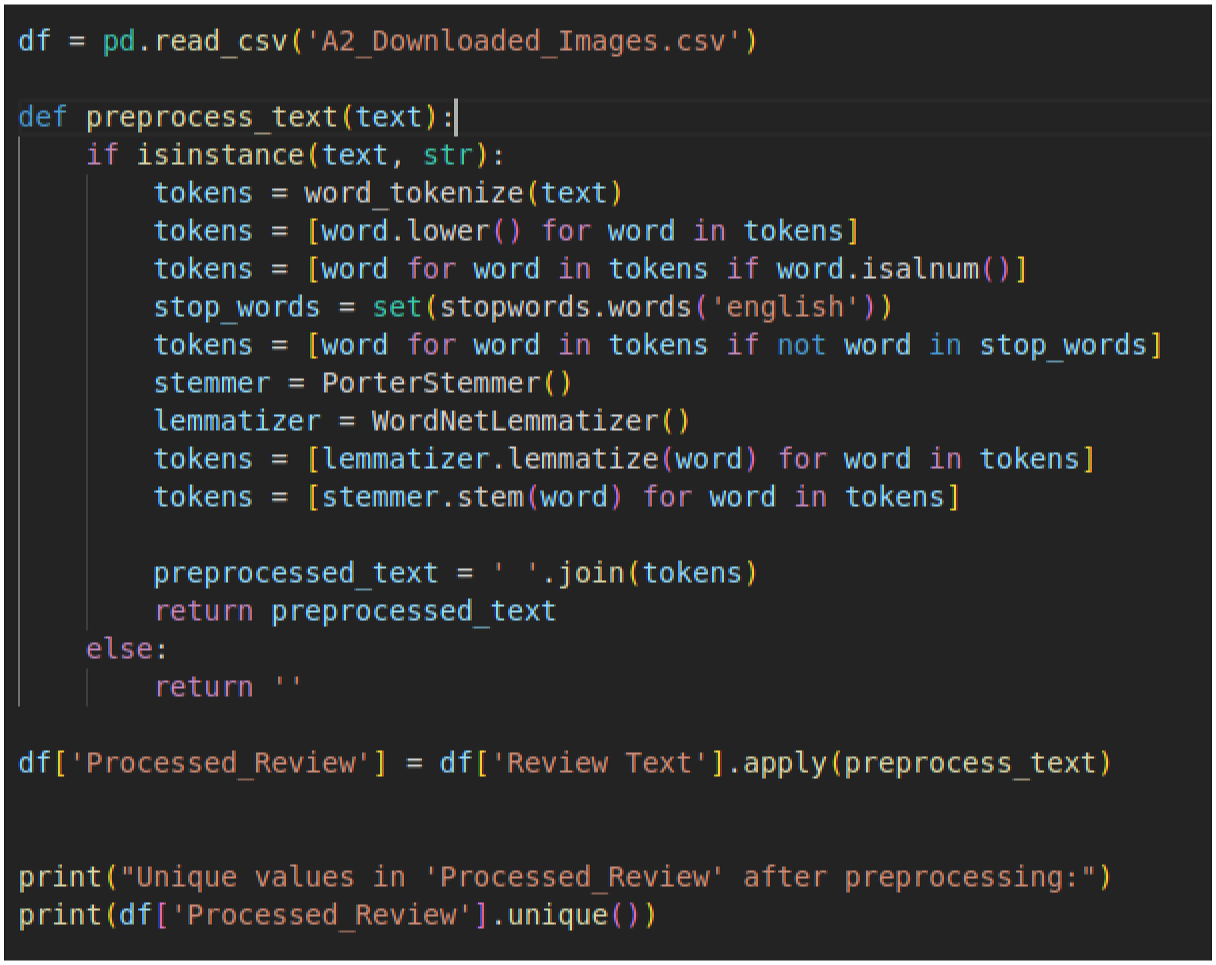


```python
df = pd.read_csv('A2_Downloaded_Images.csv')

model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

def extract_features(img):
    img = cv2.resize(img, (224, 224))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)
    features = model.predict(img)
    features = features.flatten()

def process_images_and_extract_features(image_paths):
    features_list = []
    for img_path in image_paths:
        try:
            img = cv2.imread(img_path)
            features = extract_features(img)
            features_list.append(features)
        except Exception as e:
            print(f"Error processing image {img_path}: {e}")
            features_list.append(np.zeros(25088))
    return features_list

df['Image'] = df['Image'].apply(eval)
df['Feature_Extraction'] = df['Image'].apply(process_images_and_extract_features)

df.to_csv('A2_features.csv', index=False)
```

Step 4 : Normalizing extracted features

2. Text Feature Extraction [25]

Steps : Lower Casing, Tokenization,removing punctuations, Stop Word Removal, Stemming and Lemmatization



```python
df = pd.read_csv('A2_Downloaded_Images.csv')

def preprocess_text(text):
    if isinstance(text, str):
        tokens = word_tokenize(text)
        tokens = [word.lower() for word in tokens]
        tokens = [word for word in tokens if word.isalnum()]
        stop_words = set(stopwords.words('english'))
        tokens = [word for word in tokens if not word in stop_words]
        stemmer = PorterStemmer()
        lemmatizer = WordNetLemmatizer()
        tokens = [lemmatizer.lemmatize(word) for word in tokens]
        tokens = [stemmer.stem(word) for word in tokens]

        preprocessed_text = ' '.join(tokens)
        return preprocessed_text
    else:
        return ''

df['Processed_Review'] = df['Review Text'].apply(preprocess_text)


print("Unique values in 'Processed_Review' after preprocessing:")
print(df['Processed_Review'].unique())
```

TF-IDF calculation from scratch

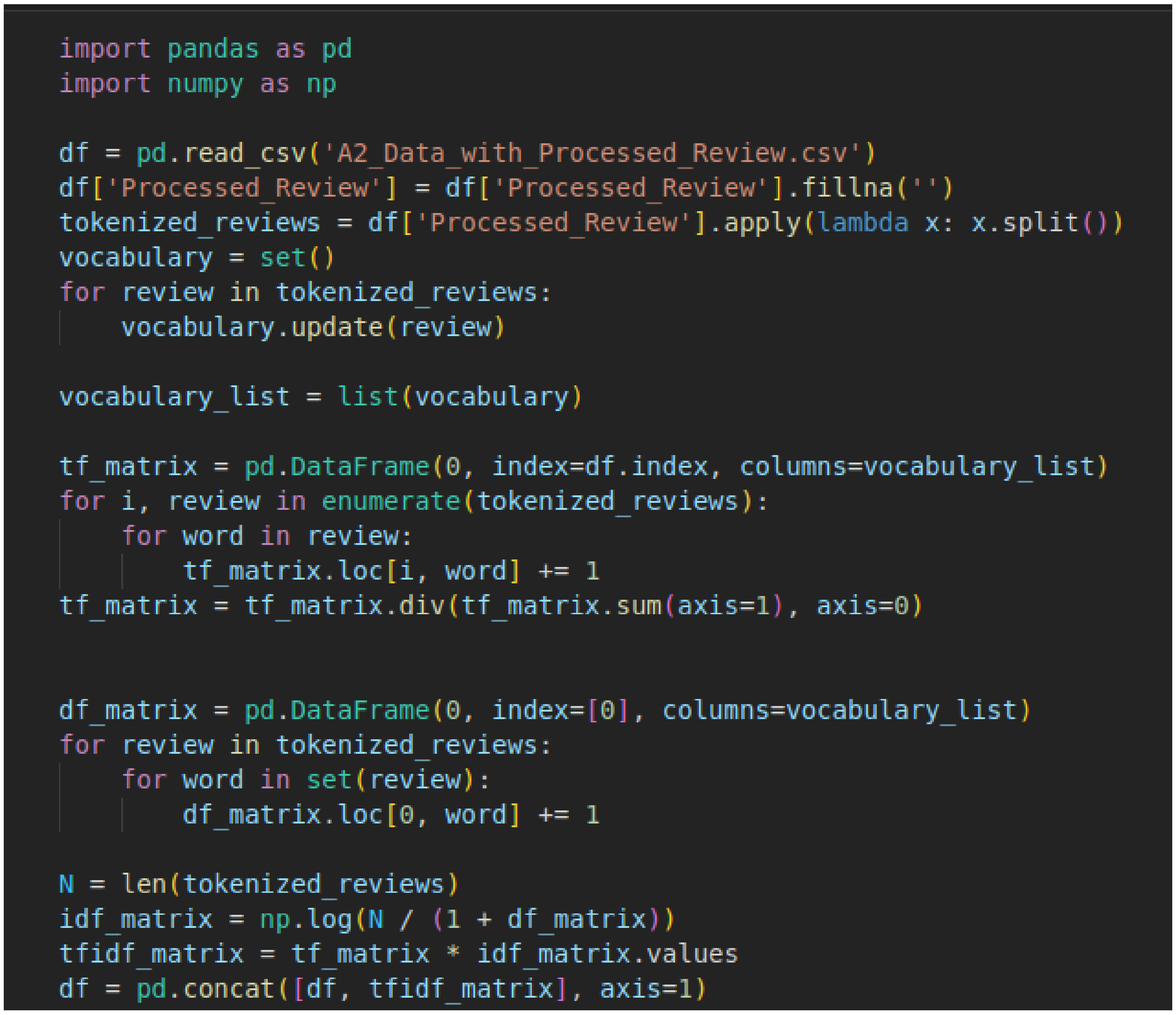
Term Frequency (TF) Calculation:
Tokenize the reviews:
Count the frequency of each word in each review:
Normalize the counts: This gives the TF value for each word in each review.

Document Frequency (DF) Calculation:
Counting the number of documents containing each word:

Inverse Document Frequency (IDF) Calculation:
Calculate IDF for each word: Apply the IDF formula, which involves taking the logarithm of the total number of documents divided by the document frequency of each word, with 1 added to the denominator to avoid division by zero.

TF-IDF Calculation:

Multiply TF by IDF for each word: Multiply the TF value of each word in each review by its IDF value.



```
import pandas as pd
import numpy as np

df = pd.read_csv('A2_Data_with_Processed_Review.csv')
df['Processed_Review'] = df['Processed_Review'].fillna('')
tokenized_reviews = df['Processed_Review'].apply(lambda x: x.split())
vocabulary = set()
for review in tokenized_reviews:
    vocabulary.update(review)

vocabulary_list = list(vocabulary)

tf_matrix = pd.DataFrame(0, index=df.index, columns=vocabulary_list)
for i, review in enumerate(tokenized_reviews):
    for word in review:
        tf_matrix.loc[i, word] += 1
tf_matrix = tf_matrix.div(tf_matrix.sum(axis=1), axis=0)


df_matrix = pd.DataFrame(0, index=[0], columns=vocabulary_list)
for review in tokenized_reviews:
    for word in set(review):
        df_matrix.loc[0, word] += 1

N = len(tokenized_reviews)
idf_matrix = np.log(N / (1 + df_matrix))
tfidf_matrix = tf_matrix * idf_matrix.values
df = pd.concat([df, tfidf_matrix], axis=1)
```