Report

Approach:

For this assignment we were asked to rank documents on the basis of similarity.

In part 1 we fetched the image from the url and applied some preprocessings (same that we applied while processing dump) then with the help of **Resnet18** we extracted relevant features and compare them to the features of images already extracted beforehand. On the baisis of cosine similarity we fetched the document respective to that image and ranked them on their scores. We ensured that top 3 documents retrieved are all different (in case of multiple images belonging to a doc).

In part 2 we fetched the text, preprocessed it which involved- tokenization, removal of stop word etc. and computed **tf-idf** (term frequency-inverse document frequency) for each term of each doc and stored the results. Now for a input text I preprocessed the text as before and computed the tf-idf. Now with every document I computed cosine similarity and ranked on the basis of scores. Top 5 documents were retrieved and returned.

In part 3 we computed the composite score which is average of the two scores and ranked the docs on the basis of these new scores.

From the observations: Top three of composite score ranking had two of them form the image retrieval methods which indicates Imge retieval performed better than text retrieval.

Reason can be:

1) Performance of Resnet Model in extracting features
2) Tf-Idf not being a very efficent method in comparing the text.
3) Data bias

Challenges:

I felt extracting features from images and computing TF-Idf scores for images and texts was a challengeing process. Also handling such large vectors for the purpose of cosine similarity was a challenge in itself.

Potential Improvements:

Using some other metrics like BM-25 or any other more powerful CNN will help.
Also instead of cosine similarity other similarity metrics will produce better results.

Methodologies:

- Features extracted were saved using pickel module locally in ExtractedFeatures.obj
- TF-IDF scores were computed and Term Frequency of each term and the IDF was also saved.
- Cosine Similarity: A simple Dot product was carried out to compute Cosine Similarity

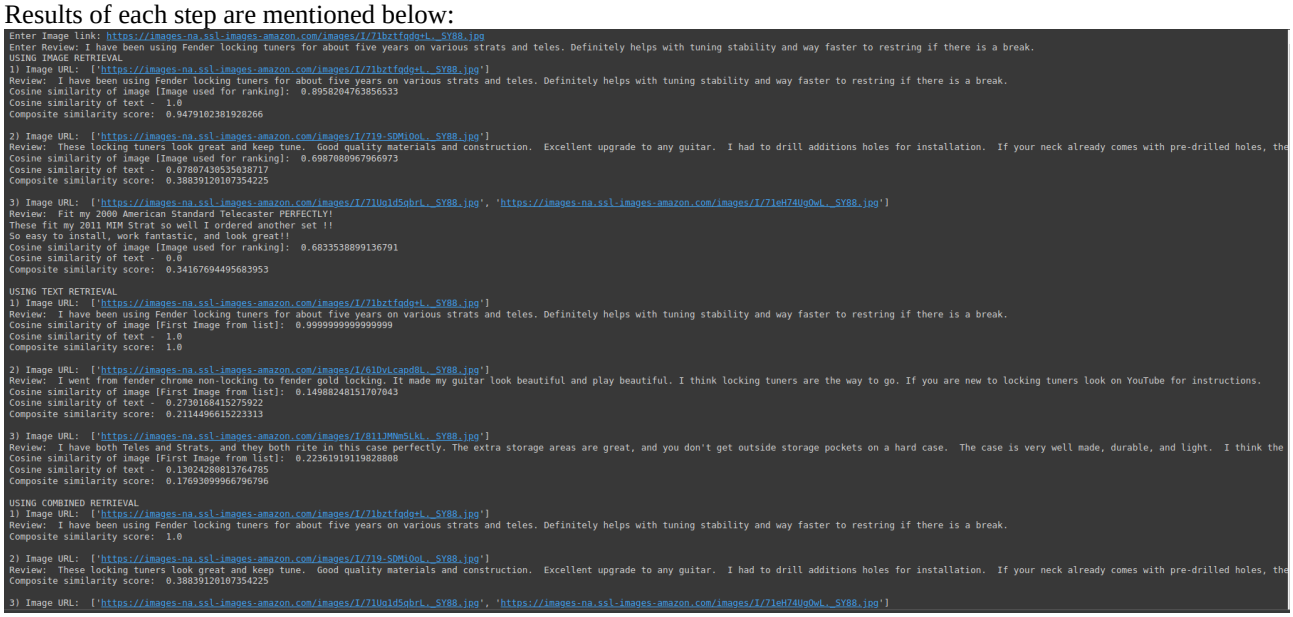
```
def cosinesim(tensor1, tensor2):
      tensor1 = np.array(tensor1)
      tensor2 = np.array(tensor2)
      len1 = len(tensor1)
      len2 = len(tensor2)
      if(len1 != len2):
            print("Invalid Vector Size")
            return 1
      else:
            sum = 0
            sum1 = 0
            sum2 = 0
            for x, y in zip(tensor1, tensor2):
                  sum += x * y
                  sum1 += x * x
                  sum2 += y * y
            if((sum1 == 0) or (sum2 == 0)): return 0
            return (sum / (math.sqrt(sum1) * math.sqrt(sum2)))
```

Assumptions:

- I have assumed that the image provided will be present in the database already.
- Since in the document some entries had multiple images while computing the respective image score for a text based retrieval. First Image of the list is taken for comparison (mentioned in assignment).
- Top 3 results of each retrieval method are presented.
- Normalization of extracted features
- Atleast three steps are done in both image and text preprocessing.

Results:

Results of each step are mentioned below:



References:

For Scraping dump files from the csv I have made use of ChatGpt and modified @anokas code according to my need:
Code: https://www.kaggle.com/anokas/py3-image-downloader-w-progress-bar
Thereby I don’t claim any ownership of the code.