EXPLICABLE CLOTHING RECOMMENDATION USING JOINT MATCHING TECHNIQUE

Report submitted to the SASTRA Deemed to be University as the requirement for the course

CSE300 - MINI PROJECT

Submitted by

BHARATHI S
(123015018, B. TECH Information technology)

DIVYA NARASIMHAN
(123003061, B. TECH Computer Science and Engineering)

KEERTHANA S
(123015047, B. TECH Information technology)

January 2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING

THANJAVUR– 613 401

Bonafide Certificate

This is to certify that the report titled "**EXPLICABLE CLOTHING RECOMMENDATION USING JOINT MATCHING TECHNIQUE**" submitted as a requirement for the course, CSE300: **MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Bharathi S(123015018), Divya Narasimhan (123003061),Keerthana S(123015047)**during the academic year 2021-22, in the School of Computing, under my supervision.

Signature of Project Supervisor **:** N. Senthil Selvan

Name with Affiliation **:** Mr Senthil Selvan

School of Computing, SASTRA Deemed University

Date **: 29-06-2022**

Mini Project *Viva voce* held on 29-06-2022

Examiner 1 **Examiner 2**

Acknowledgements

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan,** Dean**,** Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli,** Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. A. Umamakeswari**, Dean, School of Computing, **Dr. V. S. Shankar Sriram**, Associate Dean, Department of Computer Science and Engineering, **Dr. R. Muthaiah**, Associate Dean, Department of Information Technology and Information & Communication Technology and **Dr. B. Santhi**, Associate Dean, Department of Computer Application.

Our guide **Mr Senthil Selvan**, was a driving force behind this whole idea from the start. His deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which madethis project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing me an opportunity to showcase my skills through project.

Abstract

In this modern era, fashion plays a very holistic role in people's lives. While the outfits we wear can majorly impact our first impressions, it can also be stressful every day to find trendy, classy but appropriate outfits. For impulsive shoppers of the generation, buying shreds of clothing might be easy, but putting the pieces together to form an outfit might not. This project proposes the idea of creating a whole outfit with pieces of clothing through an outfit recommendation and validation system for the outfits with the help of abstract comment generation. The proposed idea is presented in the form of a novel neural network framework -Neural Outfit Recommendation (NOR). The outfit recommendation here uses a convolutional neural network with a mutual attention mechanism that compares the visual features of separate pieces of clothing and calculates the rating to predict the combination of various outfits. In order to determine the effectiveness of the recommendation, the evaluation metrics used are Mean Average Precision, Mean Reciprocal Rank (MRR) and Area under the ROC curve. Abstract Comment Generation uses a gated recurrent neural network with a cross-modality attention mechanism that also takes user comments into account to modify and generate new comments regarding the outfits from its former learnings. Rogue is the classical evaluation metric used in the text generation sector here. This project uses ExpFashion generated from FashionVC, Fashion -136K, WoW and Street2Shop as the datasets.

Keywords:

Outfit recommendation, Abstractive Comment Generation, mutual attention mechanism, cross-modality attention mechanism, Neural Outfit Recommendation, gated recurrent neural network, Mean Average Precision, mean reciprocal rank(MRR), Area under the ROC curve and Rogue.

Table of Contents

CHAPTER 1

SUMMARY OF THE BASE PAPER

Explainable Outfit Recommendation with Joint Matching Outfit Matching and Comment Generation

As mentioned in the abstract before, the basic proposal of the paper is a neural network framework specifically a Neural Outfit Recommender (NOR) that simultaneously produces matching outfits and generates active comments.

Given a top or a bottom from the given set as input, the NOR network's task is to recommend a ranked list of bottoms or tops from a candidate list respectively. The comment generation task is to generate a natural sounding comment for each recommended outfit.

Coming to the core components of the model/network, there are 3 significant ones: i) Top and Bottom image encoder ii) Mutual attention and Matching decoder iii) Generation Decoder.

1. **Top and bottom image encoder**:
    - Extracts Visual Features from images including a pair (t,b) and transforms visual features to the latent representations of t and b respectively.
        - A mutual attention mechanism is used here to guarantee that the encoder can encode the compatibility between t and b into their latent representations.
2. **Matching encoder:**
    - It is a Multi-Layered Perceptron that evaluates the matching score between t and b.
3. **Generation decoder**
    - It is a Gated Recurrent Unit used to Translate the combination of the latent representation of a top and the latent representation of the bottom into a sequence of words as comments

Coming to the evaluation metrics that's used, MAP,MRR,AUC are some significant metrics used in outfit matching and ROUGE and BLEU are some metrics used for Comment Generation.

CHAPTER 2

MERITS AND DEMERITS OF THE BASE PAPER

2.1 Literature Review:

Paper Details	Methodology Used	Observations
Neurostylist: Neural compatibility modeling for clothing matching	Dual auto-encoder network	The matching of the tops and bottoms are of low efficiencies.
Xuemeng Song, Fuli Feng, Jinhuan Liu, Zekun Li, Liqiang Nie, and Jun Ma	Bayesian personalized ranking	Comment generation is not possible
Hi, magic closet, tell me what to wear! Si Liu, Jiashi Feng, Zheng Song, Tianzhu Zhang, Hanqing Lu, Changsheng Xu, and Shuicheng Yan,	Latent Support Vector Machine	Based on small manually annotated dataset which prevents the development of complex models
Deep residual learning for image recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun,	Residual network	Training the architectures is not easy since they have many parameters and need a lot of data and time to train
Neurostylist: Neural compatibility modeling for clothing matching	Dual auto-encoder network	The matching of the tops and bottoms are of low efficiencies.
Xuemeng Song, Fuli Feng, Jinhuan Liu, Zekun Li, Liqiang Nie, and Jun Ma	Bayesian personalized ranking	Comment generation is not possible
Hi, magic closet, tell me what to wear! Si Liu, Jiashi Feng, Zheng Song, Tianzhu Zhang, Hanqing Lu, Changsheng Xu, and Shuicheng Yan,	Latent Support Vector Machine	Based on small manually annotated dataset which prevents the development of complex models
Deep residual learning for image recognition Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun,	Residual network	Training the architectures is not easy since they have many parameters and need a lot of data and time to train

2.2 Merits of Proposed Methodology:

- No previous work has studied the task of explainable outfit recommendation and generating natural language comments as explanations. Therefore, is this research on a very new direction of outfit recommendation
- Since this uses a large, real-world dataset, NOR achieves the best performance then any other models in terms of AUC. MAP and MRR. Moreover, comments generated from NOR achieve impressive ROGUE and BLEU scores.
- Simulating users to generate comments gets the model closer to user's perspective that fully expresses user's experience and feelings, making it easier for the users to understand and accept.
- The outfit Matching here is done on a public perspective since the factors that influence people's selections are current fashion, occupation and age the NOR takes everything into account for better performance of the model.
- The generated comments learned from multiple user comments reflect a general and common opinion on behalf of multiple users instead of a single specific user.
- The Outfit matching task involves different modalities for better performance
- Instead of using RNN, a more complex cross modality attention mechanism is introduced here to handle the mapping from the visual to textual space.
- NOR consistently outperforms all baseline methods in terms of MAP, MRR and AUC metrics on the Exp Fashion Dataset and achieves best result on all metrics

2.3 Demerits of Proposed Methodology

- The Proposed NOR is not personalized for a specific individual.
- Sometimes NOR cannot accurately rank the positive item at the first place.
- The generated comments are grammatical and syntactic and appropriate most of the times but sometime they are out of context for example “thank you so much for your lovely comments!” which is basically feedback on other users posted as a comment.
- In the dataset, a few comments are communication between the users so it’s better to use comment filtering methods.
- In Polyvore, comments are for outfits which include not only tops and bottoms, but also shoes, necklaces and so on. So generated comments may include items other than tops and bottoms.
- There are some other problems in the comments, like duplicate comments or duplicate words, short comments and meaningless comments
- NOR rarely generates negative comments to explain why a particular outfit combination doesn’t match which is because most of the comments in the dataset are positive
- Since there are a large percentage of short comments in the dataset, NOR generally tends to generate short comments

CHAPTER 3

SOURCE CODE : OUTFIT MATCHING

IMPORTING MODULES

```
pip uninstall numpy

Found existing installation: numpy 1.19.5
Uninstalling numpy-1.19.5:
  Would remove:
    /usr/local/bin/f2py
    /usr/local/bin/f2py3
    /usr/local/bin/f2py3.7
    /usr/local/lib/python3.7/dist-packages/numpy-1.19.5.dist-info/*
    /usr/local/lib/python3.7/dist-packages/numpy.libs/libgfortran-2e0d59d6.so.5.0.0
    /usr/local/lib/python3.7/dist-packages/numpy.libs/libopenblasp-r0-09e95953.3.13.so
    /usr/local/lib/python3.7/dist-packages/numpy.libs/libquadmath-2d0c479f.so.0.0.0
    /usr/local/lib/python3.7/dist-packages/numpy.libs/libz-eb09ad1d.so.1.2.3
    /usr/local/lib/python3.7/dist-packages/numpy/*
Proceed (y/n)? y
  Successfully uninstalled numpy-1.19.5
```

```
[ ] pip install numpy==1.19.5

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting numpy==1.19.5
  Using cached numpy-1.19.5-cp37-cp37m-manylinux2010_x86_64.whl (14.8 MB)
Installing collected packages: numpy
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the followir
xarray-einstats 0.2.2 requires numpy>=1.21, but you have numpy 1.19.5 which is incompatible.
tensorflow 2.8.2+zzzcolab20220527125636 requires numpy>=1.20, but you have numpy 1.19.5 which is incompatible.
datascience 0.10.6 requires folium==0.2.1, but you have folium 0.8.3 which is incompatible.
albumentations 0.1.12 requires imgaug<0.2.7,>=0.2.5, but you have imgaug 0.2.9 which is incompatible.
Successfully installed numpy-1.19.5
WARNING: The following packages were previously imported in this runtime:
  [numpy]
You must restart the runtime in order to use newly installed versions.

RESTART RUNTIME
```

```
[1]  import numpy
     numpy.version.version

     '1.19.5'

[2]  %tensorflow_version 1.x

     TensorFlow 1.x selected.

[3]  import tensorflow
     print(tensorflow.__version__)

     1.15.2
```

```
[4]  import tensorflow as tf
     from tensorflow.python.layers.core import Dense
     import tensorflow.contrib.keras as keras
     from keras.preprocessing.sequence import pad_sequences
     import os
     import numpy as np
     import pandas as pd
     from pandas import DataFrame
     import time
     import random
     from PIL import Image
     #from trec_eval import trec_eval
     import nltk
     import copy
```

```
[5]  os.environ['CUDA_VISIBLE_DEVICES'] = '0'
     config = tf.ConfigProto()
     config.gpu_options.allow_growth = True
```

```
[6]  random.seed(1)
     np.random.seed(1)
```

USEFUL FUNCTIONS:

```
[7]  def read_comments(file_name):
         with open(file_name,'r') as f:
             file_content = f.readlines()
         comments = []
         for line in file_content:
             comments.append(line[:-1].split())
         return comments
```

```
def build_vocab(file_name,min_num):
    with open(file_name,'r') as f:
        file_content = f.readlines()
    word_to_int = {}
    int_to_word = {}
    words_num = 0
    for line in file_content:
        line = line[:-1].split('\t')
        if int(line[2]) >= min_num:
            word_to_int[line[1]] = int(line[0])
            int_to_word[int(line[0])] = line[1]
            words_num += 1
        else:
            break
    word_to_int['<PAD>'] = 0
    word_to_int['<UNK>'] = words_num+1
    word_to_int['<GO>'] = words_num+2
    word_to_int['<EOS>'] = words_num+3
    int_to_word[0] = '<PAD>'
    int_to_word[words_num+1] = '<UNK>'
    int_to_word[words_num+2] = '<GO>'
    int_to_word[words_num+3] = '<EOS>'
    return word_to_int,int_to_word
```

```
[9]  def convert_comments(comments,word_to_int,int_to_word):
         comments_to_int = []
         for comment in comments:
             comment_to_int = [word_to_int[word] if word_to_int.get(word) != None else word_to_int['<UNK>'] for word in comment]
             comment_to_int.insert(0,word_to_int['<GO>'])
             comment_to_int.append(word_to_int['<EOS>'])
             comments_to_int.append(comment_to_int)
         return comments_to_int
```

```
[10] def negative_samples(num_samples,toplist,downlist,combinationlist):
         sampledata = []
         num = 0
         while num < num_samples:
             top = random.sample(toplist,1)[0]
             down = random.sample(downlist,1)[0]
             if top+down not in combinationlist:
                 sampledata.append((top,down,-1))
                 num += 1
         return sampledata
```

```
[11] def pad_batch(batch,pad_int):
         max_length = max([len(comment) for comment in batch])
         pad_batch = pad_sequences(batch,maxlen=max_length,value=pad_int,padding='post')
         return pad_batch
```

```
[12] def batch_to_input(batch,comments,imglist,topidlist,downidlist,pad_int):
         img1 = []#for top
         img2 = []#for down
         img1id = []
         img2id = []
         label = []
         sequence = []
         sequence_length = []
         weight = []
         for instance in batch:
             img1.append(imglist[instance[0]])
             img2.append(imglist[instance[1]])
             img1id.append(topidlist[instance[0]])
             img2id.append(downidlist[instance[1]])
             commentid = instance[2]
             if commentid == -1:
                 label.append([1,0])
                 weight.append(0)
             else:
                 label.append([0,1])
                 weight.append(1)
             sequence.append(comments[commentid])
             sequence_length.append(len(comments[commentid])-1)
         sequence = pad_batch(sequence,pad_int)
         sequence_input = sequence[:,:-1]
         sequence_output = sequence[:,1:]
         max_sequence_length = np.max(sequence_length)
         return np.array(img1),np.array(img2),np.array(img1id),np.array(img2id),np.array(label),sequence_input,sequence_output,sequence_length,max_sequence_length,np.array(weight)
```

```
[13] def get_batches(data,batch_size,comments,toplist,downlist,combinationlist,imglist,topidlist,downidlist,pad_int):
         datacopy = copy.copy(data)
         datacopy.extend(negative_samples(len(datacopy),toplist,downlist,combinationlist))
         random.shuffle(datacopy)
         for batch_i in range(0,len(datacopy)//batch_size+1):
             start_i = batch_i*batch_size
             batch = datacopy[start_i:start_i+batch_size]
             yield batch_to_input(batch,comments,imglist,topidlist,downidlist,pad_int)
```

```
[14] def build_evaluation_batch(fixitem,itemlist,state,imglist,topidlist,downidlist):
         img1 = []
         img2 = []
         img1id = []
         img2id = []
         if state == 0:#top,downs
             for item in itemlist:
                 img1.append(imglist[fixitem])
                 img2.append(imglist[item])
                 img1id.append(topidlist[fixitem])
                 img2id.append(downidlist[item])
         if state == 1:#down,tops
             for item in itemlist:
                 img1.append(imglist[item])
                 img2.append(imglist[fixitem])
                 img1id.append(topidlist[item])
                 img2id.append(downidlist[fixitem])
         return np.array(img1),np.array(img2),np.array(img1id),np.array(img2id)
```

```
[15] def id_seq_to_word_seq(id_seq,id_vocab,eos):
         index = 0
         while index < len(id_seq):
             if id_seq[index] == eos:
                 break
             index += 1
         valid_id_seq = id_seq[:index+1]
         return ' '.join([id_vocab[id] for id in valid_id_seq])
```

```
[16] def accuracy(label,prediction):
         return (label.argmax(axis=1) == prediction.argmax(axis=1)).sum()/float(len(label))
```

```
def prepare_evaluation(data_path,comments,int_to_word,word_to_int):
    with open(data_path,'r') as f:
        content = f.readlines()
    data = {}
    orderlist = []
    model_comments = {}
    labellist = {}
    query_number = 0
    for line in content:
        line = line[:-1].split('\t')
        if data.get(line[0]) != None:
            data[line[0]].append(line[1])
        else:
            data[line[0]] = [line[1]]
            labellist[query_number] = {}
            query_number += 1
            orderlist.append(line[0])
        if int(line[2]) == 1:
            model_comments[(line[0],line[1])] = [id_seq_to_word_seq(comments[int(comment)],int_to_word,word_to_int['<EOS>']).split()[1:-1] for comment in line[3].split('|')]
            labellist[query_number-1][line[1]] = 1
        else:
            labellist[query_number-1][line[1]] = 0
    return data,orderlist,model_comments,labellist
```

```
[18] def trec_evaluation(qrel_file_path,trec_file_path,trec):
         with open(trec_file_path,'w') as f:
             i = 0
             while i < len(trec):
                 j = 0
                 while j < len(trec[i]):
                     f.write(str(i)+' '+'Q0 '+trec[i][j][0]+' '+str(j+1)+' '+str(trec[i][j][1])+' '+'Exp'+'\n')
                     j += 1
                 i += 1
         result = trec_eval(qrel_file_path,trec_file_path)
         print(result)
         return result
```

```
[19] def bleu_evalaution(model_comments,system_comments,beamsearch):
         select = {}
         bleus = []
         if beamsearch:
             for combination,comments in system_comments.items():
                 scores = []
                 for comment in comments:
                     scores.append(nltk.translate.bleu_score.sentence_bleu(model_comments[combination],comment,weights=[1.0]))
                 scores = np.array(scores)
                 bleus.append(scores.max())
                 select[combination] = scores.argmax()#we only select the best for evaluation
         else:
             for combination,comment in system_comments.items():
                 bleus.append(nltk.translate.bleu_score.sentence_bleu(model_comments[combination],comment,weights=[1.0]))
         bleus = np.array(bleus)
         print(bleus.mean())
         return bleus.mean(),select
```

```
[20] def auc_evaluation(labellist,trec):
         query_number = 0
         record = []
         while query_number < len(trec):
             negative = 0
             temp = []
             for combination in trec[query_number]:
                 if labellist[query_number][combination[0]] == 1:
                     temp.append(negative)
                 else:
                     negative += 1
             record.extend([(negative-val)/float(negative) for val in temp])
             query_number += 1
         auc = np.array(record).mean()
         print(auc)
         return auc
```

MOUNTING GDRIVE:

```
[21] from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive

[22] !unzip /content/drive/MyDrive/mini_datasets.zip
       inflating: datasets/matching/img/87272222.jpg
       inflating: datasets/matching/img/87272227.jpg
       inflating: datasets/matching/img/87274269.jpg
       inflating: datasets/matching/img/87296037.jpg
       inflating: datasets/matching/img/87301745.jpg
       inflating: datasets/matching/img/87301787.jpg
       inflating: datasets/matching/img/87301999.jpg
       inflating: datasets/matching/img/87302068.jpg
       inflating: datasets/matching/img/87306041.jpg
       inflating: datasets/matching/img/87306042.jpg
       inflating: datasets/matching/img/87319379.jpg
       inflating: datasets/matching/img/87319468.jpg
       inflating: datasets/matching/img/87320156.jpg
       inflating: datasets/matching/img/87320917.jpg
       inflating: datasets/matching/img/87323243.jpg
       inflating: datasets/matching/img/87327953.jpg
       inflating: datasets/matching/img/87337446.jpg
       inflating: datasets/matching/img/87337582.jpg
       inflating: datasets/matching/img/87337784.jpg
       inflating: datasets/matching/img/87339295.jpg
       inflating: datasets/matching/img/87342545.jpg
       inflating: datasets/matching/img/87342651.jpg
       inflating: datasets/matching/img/87343030.jpg
       inflating: datasets/matching/img/87353324.jpg
       inflating: datasets/matching/img/87358007.jpg
       inflating: datasets/matching/img/87360729.jpg
```

PREPARING DATASETS:

```
[23] comments_path = '/content/drive/MyDrive/mini/text.dat'
     vocab_path = '/content/drive/MyDrive/mini/vocab.dat'
     min_num = 5

[24] comments = read_comments(comments_path)
     comments.append([])
     comments
      'top',
      'fashion',
      'set',
      '!'],
     ['yay', '!', 'congrats', 'dear', '!'],
     ['terrific', '!', 'ts', 'congrats', '!'],
     ['i', 'knew', 'this', 'was', 'a', 'winner', '!', 'congratulations', '.'],
     ['congratulations',
      'on',
      'your',
      'well',
      'deserved',
      'top',
      'fashion',
      'set',
      '.'],
     ['congrsts', 'dear', '.'],
     ['congratulations', 'on', 'top', 'sets', 'darling', '.'],
     ['gorgeousness', ',', 'congrats', '!'],
     ['gorgeous', 'top', 'fashion', 'set', '!', 'congrats', '!'],
     ['so', 'much', 'everyone', '!', 'have', 'a', 'lovely', 'day', '!'],
     ['congratulations', 'on', 'your', 'tfs', 'x', '.'],
```

```
[25] word_to_int,int_to_word = build_vocab(vocab_path,min_num)
     vocab_size = len(word_to_int)
     print(vocab_size)

     16519

     comments = convert_comments(comments,word_to_int,int_to_word)
     comments
     [16517, 53, 32, 20, 24, 73, 2, 16518],
     [16517, 20, 24, 27, 25, 73, 16, 1, 16518],
     [16517, 62, 24, 27, 2760, 94, 2, 16518],
     [16517, 30, 59, 84, 6, 1, 20, 1, 16518],
     [16517, 39, 4, 15, 65, 50, 28, 760, 1, 16518],
     [16517, 1015, 182, 18, 2, 16518],
     [16517, 52, 98, 4, 37, 30, 18, 1, 16518],
     [16517, 46, 4, 23, 12, 1, 47, 65, 14, 50, 877, 34, 40, 2, 16518],
     [16517, 121, 31, 4, 105, 49, 342, 97, 433, 28, 64, 1, 16518],
     [16517, 39, 37, 36, 295, 1, 34, 40, 2, 16518],
     [16517, 19, 18, 19, 6, 19, 285, 420, 699, 232, 167, 50, 2202, 2, 16518],
     [16517, 153, 18, 4, 19, 91, 1, 16518],
     [16517, 132, 6, 275, 437, 2, 16518],
     [16517, 121, 111, 35, 2, 34, 40, 2, 16518],
     [16517,
      117,
      35,
      33,
      8,
      21,
      3,
      35,
      55,
      126,
      76,
```

```
[27] toplist = []
     topidlist = {}
     with open('/content/datasets/CG/train/newtoplist.dat','r') as f:#in toplist, the first col is img_name of top, the second col is comments_index
         content = f.readlines()
     for line in content:
         line = line.split('\t')
         toplist.append(line[0])
         topidlist[line[0]] = len(topidlist)
     toplist
```

```
['100007206',
 '100015289',
 '100018253',
 '100018720',
 '100027085',
 '100049216',
 '100051783',
 '100058266',
 '100058372',
 '100060461',
 '100060516',
 '100060655',
 '100061551',
 '100061862',
 '100062091',
 '100062481',
 '100062898',
 '100063158',
 '100063339',
 '100063869',
 '100064184',
 '100064275',
 '100064286',
```

```
[28] topidlist
```

```
{'100007206': 0,
 '100015289': 1,
 '100018253': 2,
 '100018720': 3,
 '100027085': 4,
 '100049216': 5,
 '100051783': 6,
 '100058266': 7,
 '100058372': 8,
 '100060461': 9,
 '100060516': 10,
 '100060655': 11,
 '100061551': 12,
 '100061862': 13,
 '100062091': 14,
 '100062481': 15,
 '100062898': 16,
 '100063158': 17,
 '100063339': 18,
 '100063869': 19,
 '100064184': 20,
 '100064275': 21,
 '100064286': 22,
 '100064910': 23,
 '100080519': 24,
 '100080542': 25,
 '10008186': 26,
 '100082634': 27,
 '100083743': 28,
 '100085137': 29,
```

```
[29] downlist = []
     downidlist = {}
     with open('/content/datasets/CG/train/newdownlist.dat','r') as f:#in downlist, the first col is img_name of down(i.e. bottom), the second col is comments_index
         content = f.readlines()
     for line in content:
         line = line.split('\t')
         downlist.append(line[0])
         downidlist[line[0]] = len(downidlist)
     downlist
```

```
'94156995',
'94254297',
'94866118',
'94948605',
'95311461',
'95322887',
'95615141',
'95768810',
'95778686',
'9578746',
'95926654',
'9594361',
'95997696',
'96086948',
'96242129',
'96312246',
'96361303',
'96419272',
'96425641',
'9674181',
'96954859',
'97111963',
```

```
[31] combinationlist = set()
     with open('/content/datasets/CG/train/newcombinationlist.dat','r') as f:#
         content = f.readlines()
     for line in content:
         line = line[:-1].split('\t')
         combinationlist.add(line[0]+line[1])
     combinationlist
```

```
'100210288102413755',
'100210288103500571',
'100210288147198772',
'10021028868451009',
'100223900100906803',
'100223900112578303',
'10022390070561513',
'100224625100560071',
'100224625103638887',
'100224625159925773',
'10022462553327834',
'10022462571363838',
'10022462589907225',
'10022462599608057',
'10022462599752503',
'100239810103027161',
'100239810103135174',
'100239810104052163',
'100239810105459290',
'100239810106119429',
'100239810109634894',
'100239810111411982',
'100239810114749718',
```

```
[32] imglist = {}
     for img_idx in toplist:
         img = Image.open('/content/datasets/matching/img/'+img_idx+'.jpg')
         img = np.array(img)
         img = img/255.0
         imglist[img_idx] = img
     for img_idx in downlist:
         img = Image.open('/content/datasets/matching/img/'+img_idx+'.jpg')
         img = np.array(img)
         img = img/255.0
         imglist[img_idx] = img
     imglist
```

```
        [1., 1., 1.],
        [1., 1., 1.],
        ...,
        [1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]],

       [[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.],
        ...,
        [1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]],

       [[1., 1., 1.],
        [1., 1., 1.],
```

BUILDING THE MODEL:

```
[33] def get_input():
         img1 = tf.placeholder(tf.float32,[None,224,224,3],'img1')
         img2 = tf.placeholder(tf.float32,[None,224,224,3],'img2')
         img1id = tf.placeholder(tf.int32,[None,],'img1id')
         img2id = tf.placeholder(tf.int32,[None,],'img2id')
         label = tf.placeholder(tf.float32,[None,2],'label')
         sequence_input = tf.placeholder(tf.int32,[None,None],name='sequence_input')
         sequence_output = tf.placeholder(tf.int32,[None,None],name='sequence_output')
         sequence_length = tf.placeholder(tf.int32,[None,],name='sequence_length')
         max_sequence_length = tf.placeholder(tf.int32,[],name='max_sequence_length')
         batch_size = tf.placeholder(tf.int32,[],name='batch_size')
         learning_rate = tf.placeholder(tf.float32,[],name='learning_rate')
         keep_prob = tf.placeholder(tf.float32,[],name='keep_prob')
         ratio_c = tf.placeholder(tf.float32,[],name='ratio_c')
         ratio_g = tf.placeholder(tf.float32,[],name='ratio_g')
         weight = tf.placeholder(tf.float32,[None,],name='weight')
         flag = tf.placeholder(tf.bool,name='flag')
         return img1,img2,img1id,img2id,label,sequence_input,sequence_output,sequence_length,max_sequence_length,batch_size,learning_rate,keep_prob,ratio_c,ratio_g,weight,flag
```

```
[34] def extractor(img):
         conv1 = keras.layers.Conv2D(filters=32,kernel_size=(3,3),strides=(1,1),padding='same',activation='relu',data_format='channels_last',kernel_initializer='glorot_normal')(img)
         conv2 = keras.layers.Conv2D(filters=32,kernel_size=(3,3),strides=(1,1),padding='same',activation='relu',data_format='channels_last',kernel_initializer='glorot_normal')(conv1)
         pool1 = keras.layers.MaxPool2D(pool_size=(16,16),padding='same')(conv1)
         #print(pool1)
         pool2 = keras.layers.MaxPool2D(pool_size=(16,16),padding='same')(conv2)
         #print(pool2)
         concat = keras.layers.Concatenate(axis=-1)([pool1,pool2])
         #print(concat)
         globalpool = keras.layers.GlobalAveragePooling2D()(concat)
         #print(globalpool)
         return concat,globalpool
```

```
def image_to_image_attention(conv,globalpool):#conv=[batch_size,14,14,64]ï¼Œglobalpool=[batch_size,64]
    weights1 = tf.get_variable('weights1',shape=[64,64],initializer=tf.contrib.layers.xavier_initializer(uniform=False))
    weights2 = tf.get_variable('weights2',shape=[64,64],initializer=tf.contrib.layers.xavier_initializer(uniform=False))
    weights3 = tf.get_variable('weights3',shape=[64,1],initializer=tf.contrib.layers.xavier_initializer(uniform=False))
    attn_from = tf.matmul(globalpool,weights1)#attn_form=[batch_size,64]
    features = keras.layers.Reshape([-1,64])(conv)#features=[batch_size,196,64]
    attn_to = tf.matmul(tf.reshape(features,[-1,64]),weights2)#tf.reshape(features,[-1,64])=[batch_size*196,64]ï¼attn_to=[batch_size*196,64]
    attn_from  = tf.expand_dims(attn_from,1)#attn_from=[batch_size,1,64]
    attn_to = tf.reshape(attn_to,tf.shape(features))#attn_to=[batch_size,196,64]
    attn_logit = tf.add(attn_from,attn_to)#attn_logit=[batch_size,196,64]
    attn_logit = tf.reshape(attn_logit,[-1,64])#attn_logit=[batch_size*196,64]
    attn_logit = tf.tanh(attn_logit)
    attn_weight = tf.matmul(attn_logit,weights3)#attn_weight=[batch_size*196,1]
    attn_weight = tf.reshape(attn_weight,shape=[tf.shape(conv)[0],tf.shape(conv)[1]*tf.shape(conv)[2]])#attn_weight=[batch_size,196]
    attn_weight = tf.nn.softmax(attn_weight,name='attention_img2img')
    attn_weight = tf.expand_dims(attn_weight,-1)#attn_weight=[batch_size,196,1]
    attn_conv = tf.multiply(features,attn_weight)#attn_conv=[batch_size,196,64]
    attn_conv = tf.reduce_sum(attn_conv,axis=1)#attn_conv=[batch_size,64]
    return features,attn_conv#e=v^Ttanh(W1s+W2h)ï¼a=softmax(e)
```

```
[36] def img2vec(conv):
        extractor_output = keras.layers.Dense(300,activation='relu',kernel_initializer='glorot_normal')(conv)
        return extractor_output
```

```
[37] def img_embedding(img1id,img2id):
        top_embedding_matrix = tf.get_variable('top_embedding_matrix',shape=[len(toplist),embedding_size],initializer=tf.contrib.layers.xavier_initializer(uniform=False))
        down_embedding_matrix = tf.get_variable('down_embedding_matrix',shape=[len(downlist),embedding_size],initializer=tf.contrib.layers.xavier_initializer(uniform=False))
        img1_embedding = tf.nn.embedding_lookup(top_embedding_matrix,img1id)
        img2_embedding = tf.nn.embedding_lookup(down_embedding_matrix,img2id)
        return img1_embedding,img2_embedding
```

```
[40] def generator(sequence_input,initial_state,encoder_output,batch_size,sequence_length,max_sequence_length,vocab_size,embedding_size,keep_prob):
        embedding_matrix = tf.get_variable('embedding_matrix',shape=[vocab_size,embedding_size],initializer=tf.contrib.layers.xavier_initializer(uniform=False))
        generator_embed_sequence = tf.nn.embedding_lookup(embedding_matrix,sequence_input)
        generator_cell = tf.contrib.rnn.MultiRNNCell([get_gru_cell(keep_prob) for _ in range(1)])
        output_layer = Dense(vocab_size,kernel_initializer=tf.contrib.layers.xavier_initializer(uniform=False))
        with tf.variable_scope('generator'):
            training_helper = tf.contrib.seq2seq.TrainingHelper(generator_embed_sequence,sequence_length=sequence_length,time_major=False)
            #attention
            training_LuongAttention = tf.contrib.seq2seq.LuongAttention(num_units=512,memory=encoder_output,memory_sequence_length=None)
            training_attn_cell = tf.contrib.seq2seq.AttentionWrapper(cell=generator_cell,attention_mechanism=training_LuongAttention,attention_layer_size=512,alignment_history=False,output_atten
            training_attn_state = training_attn_cell.zero_state(batch_size,tf.float32).clone(cell_state=initial_state)
            #attention
            training_decoder = tf.contrib.seq2seq.BasicDecoder(training_attn_cell,helper=training_helper,initial_state=training_attn_state,output_layer=output_layer)
            training_generator_output,training_generator_state,_ = tf.contrib.seq2seq.dynamic_decode(training_decoder,output_time_major=False,impute_finished=True,maximum_iterations=max_sequence_
        with tf.variable_scope('generator',reuse=True):
            start_tokens = tf.tile(tf.constant([word_to_int['<GO>']],dtype=tf.int32),[batch_size])
            #attention
            predicting_LuongAttention = tf.contrib.seq2seq.LuongAttention(num_units=512,memory=encoder_output,memory_sequence_length=None)
            predicting_attn_cell = tf.contrib.seq2seq.AttentionWrapper(cell=generator_cell,attention_mechanism=predicting_LuongAttention,attention_layer_size=512,alignment_history=True,output_att
            predicting_attn_state = predicting_attn_cell.zero_state(batch_size,tf.float32).clone(cell_state=initial_state)
            #attention
            predicting_helper = tf.contrib.seq2seq.GreedyEmbeddingHelper(embedding_matrix,start_tokens,word_to_int['<EOS>'])
            predicting_decoder = tf.contrib.seq2seq.BasicDecoder(predicting_attn_cell,predicting_helper,predicting_attn_state,output_layer)
            predicting_generator_output,predicting_generator_state,_ = tf.contrib.seq2seq.dynamic_decode(predicting_decoder,output_time_major=False,impute_finished=True,maximum_iterations=max_seq
            attention_matrix = tf.identity(predicting_generator_state.alignment_history.stack(),name='attention_matrix')
            #print(attention_matrix)
        with tf.variable_scope('generator',reuse=True):
            start_tokens = tf.tile(tf.constant([word_to_int['<GO>']],dtype=tf.int32),[batch_size])
            beamsearch_initial_state = tf.contrib.seq2seq.tile_batch(initial_state,multiplier=3)
            #attention
            beamsearch_encoder_output = tf.contrib.seq2seq.tile_batch(encoder_output,multiplier=3)
            beamsearch_LuongAttention = tf.contrib.seq2seq.LuongAttention(num_units=512,memory=beamsearch_encoder_output,memory_sequence_length=None)
            beamsearch_attn_cell = tf.contrib.seq2seq.AttentionWrapper(cell=generator_cell ,attention_mechanism=beamsearch_LuongAttention,attention_layer_size=512,alignment_history=False,output_
            beamsearch_attn_state = beamsearch_attn_cell.zero_state(batch_size*3,tf.float32).clone(cell_state=beamsearch_initial_state)
```

```
[40]        #attention
            predicting_helper = tf.contrib.seq2seq.GreedyEmbeddingHelper(embedding_matrix,start_tokens,word_to_int['<EOS>'])
            predicting_decoder = tf.contrib.seq2seq.BasicDecoder(predicting_attn_cell,predicting_helper,predicting_attn_state,output_layer)
            predicting_generator_output,predicting_generator_state,_ = tf.contrib.seq2seq.dynamic_decode(predicting_decoder,output_time_major=False,impute_finished=True,maximum_iterations=max_seq
            attention_matrix = tf.identity(predicting_generator_state.alignment_history.stack(),name='attention_matrix')
            #print(attention_matrix)
        with tf.variable_scope('generator',reuse=True):
            start_tokens = tf.tile(tf.constant([word_to_int['<GO>']],dtype=tf.int32),[batch_size])
            beamsearch_initial_state = tf.contrib.seq2seq.tile_batch(initial_state,multiplier=3)
            #attention
            beamsearch_encoder_output = tf.contrib.seq2seq.tile_batch(encoder_output,multiplier=3)
            beamsearch_LuongAttention = tf.contrib.seq2seq.LuongAttention(num_units=512,memory=beamsearch_encoder_output,memory_sequence_length=None)
            beamsearch_attn_cell = tf.contrib.seq2seq.AttentionWrapper(cell=generator_cell ,attention_mechanism=beamsearch_LuongAttention,attention_layer_size=512,alignment_history=False,output_a
            beamsearch_attn_state = beamsearch_attn_cell.zero_state(batch_size*3,tf.float32).clone(cell_state=beamsearch_initial_state)
            #attention
            beamsearch_predicting_decoder = tf.contrib.seq2seq.BeamSearchDecoder(beamsearch_attn_cell,embedding=embedding_matrix,start_tokens=start_tokens,end_token=word_to_int['<EOS>'],initial_s
            beamsearch_generator_output,beamsearch_generator_state,_ = tf.contrib.seq2seq.dynamic_decode(beamsearch_predicting_decoder,output_time_major=False,impute_finished=False,maximum_iterat
        return training_generator_output,predicting_generator_output,beamsearch_generator_output
```

```
[41] def loss(classifier_output,label,training_generator_output,sequence_output,sequence_length,max_sequence_length,ratio_c,ratio_g,weight,flag):
        classifier_loss = tf.reduce_mean(tf.contrib.keras.losses.categorical_crossentropy(label,classifier_output),name='classifier_loss')
        classifier_loss_freeze = tf.stop_gradient(classifier_loss)
        classifier_loss = tf.where(flag,classifier_loss,classifier_loss_freeze)
        training_logits = tf.identity(training_generator_output.rnn_output,name='training_logits')
        masks = tf.sequence_mask(sequence_length,max_sequence_length,dtype=tf.float32,name='mask')
        generator_loss = tf.contrib.seq2seq.sequence_loss(training_logits,sequence_output,masks,average_across_timesteps=False,average_across_batch=False)
        generator_loss = tf.reduce_sum(generator_loss,axis=1)
        generator_loss = tf.multiply(weight,generator_loss)
        generator_loss = tf.reduce_mean(generator_loss,name='generator_loss')
        classifier_loss = tf.multiply(ratio_c,classifier_loss)
        generator_loss = tf.multiply(ratio_g,generator_loss)
        tv = tf.trainable_variables()
        reg_loss = tf.reduce_sum([tf.nn.l2_loss(v) for v in tv])
        reg_loss_gen = tf.reduce_sum([tf.nn.l2_loss(v) for v in tv if ('generator' in v.name)])
        reg_loss = tf.where(flag,reg_loss,reg_loss_gen)
        loss = tf.add_n([classifier_loss,generator_loss,0.0001*reg_loss],name='loss')
        return loss
```

```
[42] def optimizer(loss,learning_rate):
        optimizer = tf.train.AdamOptimizer(learning_rate)
        gradients = optimizer.compute_gradients(loss)
        capped_gradients = [(tf.clip_by_value(grad,-5.,5.),var) for grad,var in gradients if grad is not None]
        train_op = optimizer.apply_gradients(capped_gradients)
        return train_op
```

```
[43] def prediction(classifier_output):
        prediction = tf.identity(classifier_output,name='prediction')
        return prediction
```

```
[44] def generation(predicting_generator_output,beamsearch_generator_output):
        greedysearch_sequence = tf.identity(predicting_generator_output.sample_id,name='greedysearch_sequence')
        beamsearch_sequence = tf.identity(beamsearch_generator_output.predicted_ids,name='beamsearch_sequence')
        return greedysearch_sequence,beamsearch_sequence
```

```
[45] embedding_size = 300
    train_graph = tf.Graph()
    with train_graph.as_default():
        tf.set_random_seed(1)
        with tf.name_scope('inputs'):
            img1,img2,img11d,img21d,label,sequence_input,sequence_output,sequence_length,max_sequence_length,batch_size,learning_rate,keep_prob,ratio_c,ratio_g,weight,flag = get_input()
        with tf.name_scope('extractor'):
            with tf.variable_scope('extractor'):
                conv_img1,globalpool_img1 = extractor(img1)
            with tf.variable_scope('extractor',reuse=True):
                conv_img2,globalpool_img2 = extractor(img2)
            with tf.variable_scope('image_to_image_attention'):
                features_img1,attn_conv_img1 = image_to_image_attention(conv_img1,globalpool_img2)
            with tf.variable_scope('image_to_image_attention',reuse=True):
                features_img2,attn_conv_img2 = image_to_image_attention(conv_img2,globalpool_img1)
            with tf.variable_scope('img2vec'):
                extractor_output_img1 = img2vec(attn_conv_img1)
            with tf.variable_scope('img2vec',reuse=True):
                extractor_output_img2 = img2vec(attn_conv_img2)
            with tf.variable_scope('img_embedding'):
                img1_embedding,img2_embedding = img_embedding(img11d,img21d)
            extractor_output = tf.concat([extractor_output_img1,extractor_output_img2,img1_embedding,img2_embedding],axis=1)
            encoder_output = tf.concat([features_img1,features_img2],axis=1)
            encoder_output_freeze = tf.stop_gradient(encoder_output)
            extractor_output_freeze = tf.stop_gradient(extractor_output)
            encoder_output = tf.where(flag,encoder_output,encoder_output_freeze)
            extractor_output = tf.where(flag,extractor_output,extractor_output_freeze)
        with tf.name_scope('classifier'):
            classifier_output = classifier(extractor_output,keep_prob)
        with tf.name_scope('prediction'):
            prediction = prediction(classifier_output)
        with tf.name_scope('generator'):
            dense_output = keras.layers.Dense(512,activation='tanh',kernel_initializer='glorot_normal')(extractor_output)
```

```
[45]            with tf.variable_scope('image_to_image_attention'):
                    features_img1,attn_conv_img1 = image_to_image_attention(conv_img1,globalpool_img2)
                with tf.variable_scope('image_to_image_attention',reuse=True):
                    features_img2,attn_conv_img2 = image_to_image_attention(conv_img2,globalpool_img1)
                with tf.variable_scope('img2vec'):
                    extractor_output_img1 = img2vec(attn_conv_img1)
                with tf.variable_scope('img2vec',reuse=True):
                    extractor_output_img2 = img2vec(attn_conv_img2)
                with tf.variable_scope('img_embedding'):
                    img1_embedding,img2_embedding = img_embedding(img11d,img21d)
                extractor_output = tf.concat([extractor_output_img1,extractor_output_img2,img1_embedding,img2_embedding],axis=1)
                encoder_output = tf.concat([features_img1,features_img2],axis=1)
                encoder_output_freeze = tf.stop_gradient(encoder_output)
                extractor_output_freeze = tf.stop_gradient(extractor_output)
                encoder_output = tf.where(flag,encoder_output,encoder_output_freeze)
                extractor_output = tf.where(flag,extractor_output,extractor_output_freeze)
            with tf.name_scope('classifier'):
                classifier_output = classifier(extractor_output,keep_prob)
            with tf.name_scope('prediction'):
                prediction = prediction(classifier_output)
            with tf.name_scope('generator'):
                dense_output = keras.layers.Dense(512,activation='tanh',kernel_initializer='glorot_normal')(extractor_output)
                initial_state = (dense_output,)
                training_generator_output,predicting_generator_output,beamsearch_generator_output = generator(sequence_input,initial_state,encoder_output,batch_size,sequence_length,max_sequence_lengt
            with tf.name_scope('generation'):
                greedysearch_sequence,beamsearch_sequence = generation(predicting_generator_output,beamsearch_generator_output)
            with tf.name_scope('loss'):
                loss = loss(classifier_output,label,training_generator_output,sequence_output,sequence_length,max_sequence_length,ratio_c,ratio_g,weight,flag)
            with tf.name_scope('optimizer'):
                train_op = optimizer(loss,learning_rate)
```

TRAINING THE MODEL:

Train Model

```
[46] with open('/content/datasets/CG/train/newtraindata.dat','r') as f:
         content = f.readlines()
     traindata = []
     for line in content:
         line = line[:-1].split('\t')
         traindata.append((line[0],line[1],int(line[2])))
     traindata
```

```
('100018720', '101034623', 299638),
('100018720', '101034623', 299639),
('100018720', '101034623', 299640),
('100018720', '101034623', 299641),
('100018720', '101034623', 299642),
('100018720', '101034623', 299643),
('100018720', '101034623', 299644),
('100018720', '101034623', 299645),
('100018720', '101034623', 299646),
('100018720', '101034623', 299647),
('100018720', '101034623', 299648),
('100018720', '101034623', 299649),
('100018720', '101034623', 299650),
('100018720', '101034623', 299651),
('100018720', '101034623', 299652),
('100018720', '101034623', 299653),
('100018720', '101034623', 299654),
('100018720', '101034623', 299655),
('100018720', '101034623', 299656),
('100018720', '101034623', 299657),
('100018720', '101034623', 299658),
('100018720', '101034623', 299659),
('100018720', '101034623', 299660),
```

```
[47] tops_qrel_file_path = '/content/datasets/CG/eval/newdevdata_tops_qrel.dat'
     tops_trec_file_path = '/content/datasets/CG/eval/newdevdata_tops_trec.dat'
     #downs_qrel_file_path = 'evaluation/devdata_downs_qrel.dat'
     #downs_trec_file_path = 'evaluation/devdata_downs_trec.dat'

[48] data_path = '/content/datasets/CG/train/newdevdata_tops.dat'
     dev_tops_data,tops_orderlist,model_tops_comments,tops_labellist = prepare_evaluation(data_path,comments,int_to_word,word_to_int)

[49] #data_path = 'dataset/devdata_downs.dat'
     #dev_downs_data,downs_orderlist,model_downs_comments,downs_labellist = prepare_evaluation(data_path,comments,int_to_word,word_to_int)
```

```
[50] model_tops_comments

{('100080542', '139427517'): [['excellent', 'casual', 'look', '!'],
  ['i',
   'love',
   'cozy',
   'fashion',
   'for',
   'winter',
   'and',
   'nice',
   'style',
   'too',
   'sweetie',
   'great',
   'set',
   'here',
   'just',
   'so',
   'perfect',
   '~',
   '.'],
```

```
[51] #model_downs_comments

[52] lr = 0.001
     rat_c = 1.0
     rat_g = 1.0
     epochs = 5
     rate = 1.0

[53] cla_cost_list = []
     gen_cost_list = []
     bleus_tops = []
     auc_tops = []
     trec_evals_tops = []
     #bleus_downs = []
     #trec_evals_downs = []
     #auc_downs = []
```

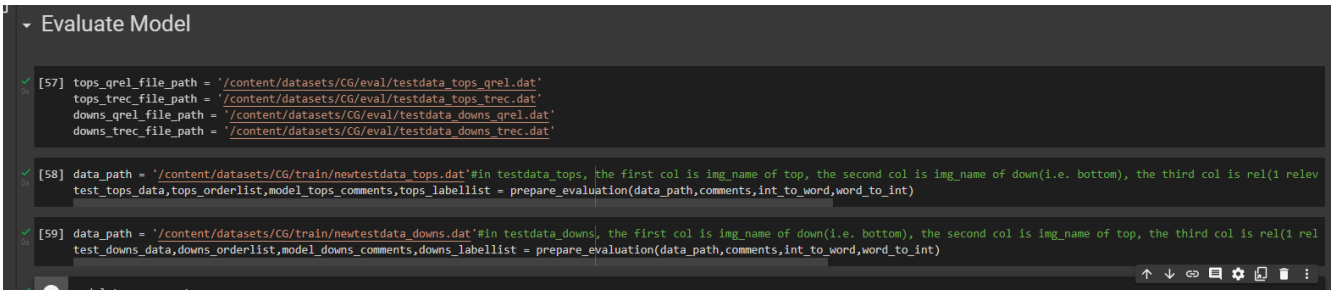
```
[54] beamsearch = True
    checkpoint = 'checkpoint/trained_model.ckpt'
    with tf.Session(graph=train_graph,config=config) as sess:
        writer = tf.summary.FileWriter('checkpoint/',sess.graph)
        saver = tf.train.Saver()
        sess.run(tf.global_variables_initializer())
        print(time.localtime())
        classifier_loss = train_graph.get_tensor_by_name('loss/classifier_loss:0')
        generator_loss = train_graph.get_tensor_by_name('loss/generator_loss:0')
        for epoch in range(epochs):
            b_s = 64#batch_size
            train_cla_cost = 0
            train_gen_cost = 0
            temp_cla_cost_list = []
            temp_gen_cost_list = []
            step = 0
            for _,(x_i1,x_i2,x_id1,x_id2,y_1,x_s_i,x_s_o,seq_len,max_seq_len,wei) in enumerate(get_batches(traindata,b_s,comments,toplist,downlist,combinationlist,imglist,topidlist,downidlist,wor
                _,cost1,cost2 = sess.run([train_op,classifier_loss,generator_loss],{img1:x_i1,img2:x_i2,img1id:x_id1,img2id:x_id2,label:y_1,sequence_input:x_s_i,sequence_output:x_s_o,sequence_len
                train_cla_cost += cost1
                train_gen_cost += cost2
                step += 1
                if step%1000 == 0:
                    temp_cla_cost_list.append(train_cla_cost/step)
                    temp_gen_cost_list.append(train_gen_cost/step)
                    print(str(train_cla_cost/step)+'&'+str(train_gen_cost/step)+' '+'pass!')
            temp_cla_cost_list.append(train_cla_cost/step)
            temp_gen_cost_list.append(train_gen_cost/step)
            cla_cost_list.append(temp_cla_cost_list)
            gen_cost_list.append(temp_gen_cost_list)
            print('Epoch {}/{} - Training Loss: {:.3f}&{:.3f}'.format(epoch+1,epochs,train_cla_cost/step,train_gen_cost/step))
            saver.save(sess,checkpoint,global_step=epoch+1)
            print('Model Trained and Saved')
            print(time.localtime())
```

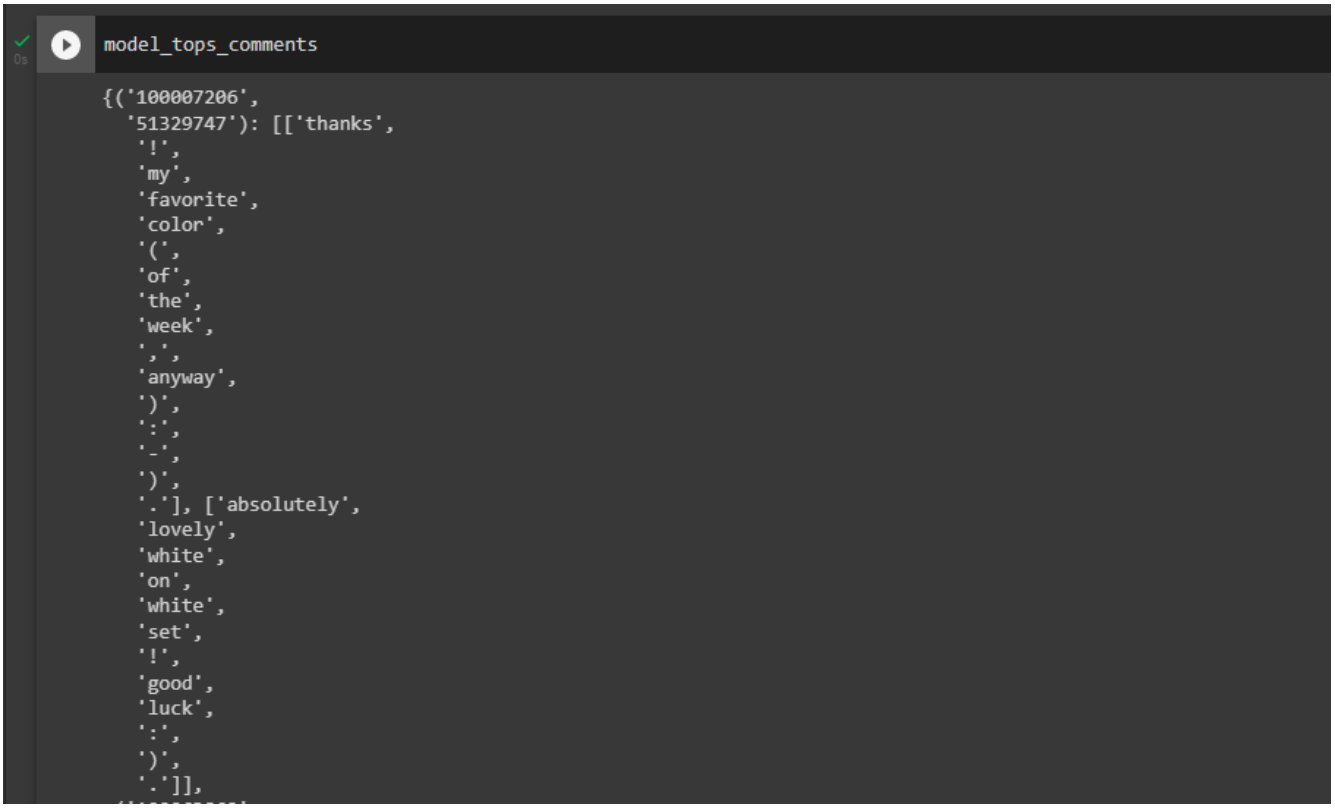
```
[54]  cla_cost_list.append(temp_cla_cost_list)
      gen_cost_list.append(temp_gen_cost_list)
      print('Epoch {}/{} - Training Loss: {:.3f}&{:.3f}'.format(epoch+1,epochs,train_cla_cost/step,train_gen_cost/step))
      saver.save(sess,checkpoint,global_step=epoch+1)
      print('Model Trained and Saved')
      print(time.localtime())
      #validation
      b_s = 64
      max_seq_len = 30
      system_tops_comments = {}
      tops_trec = {}
      query_number = 0
      step = 0
      for top in tops_orderlist:
          downsoftop = dev_tops_data[top]
          probabilitylist = {}
          for batch_i in range(len(downsoftop)//b_s+1):
              start_i = batch_i*b_s
              downs = downsoftop[start_i:start_i+b_s]
              x_i1,x_i2,x_id1,x_id2 = build_evaluation_batch(top,downs,0,imglist,topidlist,downidlist)
              seq_len = [30]*len(x_i1)
              prob,gred_seq,beam_seq = sess.run([prediction,greedysearch_sequence,beamsearch_sequence],{img1:x_i1,img2:x_i2,img1id:x_id1,img2id:x_id2,sequence_length:seq_len,max_sequence_le
              j = 0
              for down in downs:
                  probabilitylist[down] = prob[j][1]
                  if model_tops_comments.get((top,down)) != None:
                      if beamsearch:
                          system_tops_comments[(top,down)] = [(id_seq_to_word_seq(beam_seq[j][:,index],int_to_word,word_to_int['<EOS>'])).split()[:-1] for index in range(3)]#3 is beam_width
                      else:
                          system_tops_comments[(top,down)] = (id_seq_to_word_seq(gred_seq[j],int_to_word,word_to_int['<EOS>'])).split()[:-1]
                  j += 1
              step += 1
              if step%1000 == 0:
                  print('pass!')
```

```
[53]                  if model_tops_comments.get((top,down)) != None:
                          if beamsearch:
                              system_tops_comments[(top,down)] = [(id_seq_to_word_seq(beam_seq[j][:,index],int_to_word,word_to_int['<EOS>'])).split()[:-1] for index in range(3)]#3 is beam_width
                          else:
                              system_tops_comments[(top,down)] = (id_seq_to_word_seq(gred_seq[j],int_to_word,word_to_int['<EOS>'])).split()[:-1]
                      j += 1
                  step += 1
                  if step%1000 == 0:
                      print('pass!')
              tops_trec[query_number] = sorted(probabilitylist.items(),key=lambda item:item[1],reverse=True)
              del probabilitylist,downsoftop
              query_number += 1
          bleu,_ = bleu_evalaution(model_tops_comments,system_tops_comments,beamsearch)
          bleus_tops.append(bleu)
          del system_tops_comments
          auc_tops.append(auc_evaluation(tops_labellist,tops_trec))
          #trec_evals_tops.append(trec_evaluation(tops_qrel_file_path,tops_trec_file_path,tops_trec))
          del tops_trec
```

```
[54]  #validation
      print(time.localtime())

time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=14, tm_sec=20, tm_wday=2, tm_yday=180, tm_isdst=0)
Epoch 1/5 - Training Loss: 0.564&30.283
Model Trained and Saved
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=14, tm_sec=59, tm_wday=2, tm_yday=180, tm_isdst=0)
0.14938787205237164
0.0
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=15, tm_sec=0, tm_wday=2, tm_yday=180, tm_isdst=0)
Epoch 2/5 - Training Loss: 0.479&25.355
Model Trained and Saved
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=15, tm_sec=29, tm_wday=2, tm_yday=180, tm_isdst=0)
0.6755731804675386
0.0
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=15, tm_sec=29, tm_wday=2, tm_yday=180, tm_isdst=0)
Epoch 3/5 - Training Loss: 0.461&23.101
Model Trained and Saved
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=15, tm_sec=58, tm_wday=2, tm_yday=180, tm_isdst=0)
0.6347623272188053
0.0
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=15, tm_sec=58, tm_wday=2, tm_yday=180, tm_isdst=0)
Epoch 4/5 - Training Loss: 0.415&21.365
Model Trained and Saved
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=16, tm_sec=28, tm_wday=2, tm_yday=180, tm_isdst=0)
0.8044625229866565
0.0
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=16, tm_sec=28, tm_wday=2, tm_yday=180, tm_isdst=0)
Epoch 5/5 - Training Loss: 0.370&19.943
Model Trained and Saved
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=16, tm_sec=58, tm_wday=2, tm_yday=180, tm_isdst=0)
0.8097623272188053
0.0
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=16, tm_sec=58, tm_wday=2, tm_yday=180, tm_isdst=0)
```

EVALUVATING THE MODEL:





```
beamsearch = True
print(time.localtime())
checkpoint = 'checkpoint/trained_model.ckpt-1'
test_graph = tf.Graph()
with tf.Session(graph=test_graph,config=config) as sess:
    loader = tf.train.import_meta_graph(checkpoint+'.meta')
    loader.restore(sess,checkpoint)
    img1 = test_graph.get_tensor_by_name('inputs/img1:0')
    img2 = test_graph.get_tensor_by_name('inputs/img2:0')
    img1id = test_graph.get_tensor_by_name('inputs/img1id:0')
    img2id = test_graph.get_tensor_by_name('inputs/img2id:0')
    sequence_length = test_graph.get_tensor_by_name('inputs/sequence_length:0')
    max_sequence_length = test_graph.get_tensor_by_name('inputs/max_sequence_length:0')
    batch_size = test_graph.get_tensor_by_name('inputs/batch_size:0')
    keep_prob = test_graph.get_tensor_by_name('inputs/keep_prob:0')
    flag = test_graph.get_tensor_by_name('inputs/flag:0')
    prediction = test_graph.get_tensor_by_name('prediction/prediction:0')
    greedysearch_sequence = test_graph.get_tensor_by_name('generation/greedysearch_sequence:0')
    beamsearch_sequence = test_graph.get_tensor_by_name('generation/beamsearch_sequence:0')
    b_s = 64
    max_seq_len = 30
    system_tops_comments = {}
    tops_trec = {}
    query_number = 0
```

```
    query_number = 0
    step = 0
    for top in tops_orderlist:
        downsoftop = test_tops_data[top]
        probabilitylist = {}
        for batch_i in range(len(downsoftop)//b_s+1):
            start_i = batch_i*b_s
            downs = downsoftop[start_i:start_i+b_s]
            x_i1,x_i2,x_id1,x_id2 = build_evaluation_batch(top,downs,0,imglist,topidlist,downidlist)
            seq_len = [30]*len(x_i1)
            prob,gred_seq,beam_seq = sess.run([prediction,greedysearch_sequence,beamsearch_sequence],{img1:x_i1,img2:x_i2,img1id:x_id1,img2id:x_id2,sequence_length:seq_len,max_sequence_length
            j = 0
            for down in downs:
                probabilitylist[down] = prob[j][1]
                if model_tops_comments.get((top,down)) != None:
                    if beamsearch:
                        system_tops_comments[(top,down)] = [(id_seq_to_word_seq(beam_seq[j][:,index],int_to_word,word_to_int['<EOS>'])).split()[:-1] for index in range(3)]
                    else:
                        system_tops_comments[(top,down)] = (id_seq_to_word_seq(gred_seq[j],int_to_word,word_to_int['<EOS>'])).split()[:-1]
                j += 1
            step += 1
            if step%1000 == 0:
                print('pass!')
        tops_trec[query_number] = sorted(probabilitylist.items(),key=lambda item:item[1],reverse=True)
        del probabilitylist,downsoftop
        query_number += 1
    _,select_tops = bleu_evalaution(model_tops_comments,system_tops_comments,beamsearch)
    #auc_evaluation(tops_labellist,tops_trec)
    #trec_evaluation(tops_qrel_file_path,tops_trec_file_path,tops_trec)
    del tops_trec
    system_downs_comments = {}
    downs_trec = {}
    query_number = 0
    step = 0
```

```
        topsofdown = test_downs_data[down]
        probabilitylist = {}
        for batch_i in range(len(topsofdown)//b_s+1):
            start_i = batch_i*b_s
            tops = topsofdown[start_i:start_i+b_s]
            x_i1,x_i2,x_id1,x_id2 = build_evaluation_batch(down,tops,1,imglist,topidlist,downidlist)
            seq_len = [30]*len(x_i1)
            prob,gred_seq,beam_seq = sess.run([prediction,greedysearch_sequence,beamsearch_sequence],{img1:x_i1,img2:x_i2,img1id:x_id1,img2id:x_id2,sequence_length:seq_len,max_sequence_length:max_seq_len,batch_size
            j = 0
            for top in tops:
                probabilitylist[top] = prob[j][1]
                if model_downs_comments.get((down,top)) != None:
                    if beamsearch:
                        system_downs_comments[(down,top)] = [(id_seq_to_word_seq(beam_seq[j][:,index],int_to_word,word_to_int['<EOS>'])).split()[:-1] for index in range(3)]
                    else:
                        system_downs_comments[(down,top)] = (id_seq_to_word_seq(gred_seq[j],int_to_word,word_to_int['<EOS>'])).split()[:-1]
                j += 1
            step += 1
            if step%1000 == 0:
                print('pass!')
        downs_trec[query_number] = sorted(probabilitylist.items(),key=lambda item:item[1],reverse=True)
        del probabilitylist,topsofdown
        query_number += 1
    _,select_downs = bleu_evalaution(model_downs_comments,system_downs_comments,beamsearch)
    #auc_evaluation(downs_labellist,downs_trec)
    #trec_evaluation(downs_qrel_file_path,downs_trec_file_path,downs_trec)
    del downs_trec
print(time.localtime())
```

```
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=36, tm_sec=46, tm_wday=2, tm_yday=180, tm_isdst=0)
INFO:tensorflow:Restoring parameters from checkpoint/trained_model.ckpt-1
0.10266908037989876
0.08981171857766156
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=29, tm_hour=2, tm_min=36, tm_sec=51, tm_wday=2, tm_yday=180, tm_isdst=0)
```

```
[62] with open('/content/drive/MyDrive/mini/system_comments/system_tops_comments.dat','w') as f:
        if beamsearch:
            for combination,commentlist in system_tops_comments.items():
                comment = ' '.join(commentlist[select_tops[combination]])
                f.write(combination[0]+'\t'+combination[1]+'\t'+comment+'\n')
        else:
            for combination,comment in system_downs_comments.items():
                comment = ' '.join(comment)
                f.write(combination[0]+'\t'+combination[1]+'\t'+comment+'\n')
```

```
with open('/content/drive/MyDrive/mini/system_comments/system_downs_comments.dat','w') as f:
    if beamsearch:
        for combination,commentlist in system_downs_comments.items():
            comment = ' '.join(commentlist[select_downs[combination]])
            f.write(combination[0]+'\t'+combination[1]+'\t'+comment+'\n')
    else:
        for combination,comment in system_downs_comments.items():
            comment = ' '.join(comment)
            f.write(combination[0]+'\t'+combination[1]+'\t'+comment+'\n')
```

CHAPTER 4

SOURCE CODE: COMMENT GENERATION

IMPORTING MODULES:

```
pip uninstall numpy
```

```
Found existing installation: numpy 1.21.6
Uninstalling numpy-1.21.6:
  Would remove:
    /usr/bin/f2py
    /usr/local/bin/f2py
    /usr/local/bin/f2py3
    /usr/local/bin/f2py3.7
    /usr/local/lib/python3.7/dist-packages/numpy-1.21.6.dist-info/*
    /usr/local/lib/python3.7/dist-packages/numpy.libs/libgfortran-2e0d59d6.so.5.0.0
    /usr/local/lib/python3.7/dist-packages/numpy.libs/libopenblasp-r0-2d23e62b.3.17.so
    /usr/local/lib/python3.7/dist-packages/numpy.libs/libquadmath-2d0c479f.so.0.0.0
    /usr/local/lib/python3.7/dist-packages/numpy/*
Proceed (y/n)? y
  Successfully uninstalled numpy-1.21.6
```

```
[ ] pip install numpy==1.19.5
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: numpy==1.19.5 in /usr/local/lib/python3.7/dist-packages (1.19.5)
```

```
[ ] import numpy
    numpy.version.version
```

```
'1.19.5'
```

```
[ ] %tensorflow_version 1.x
```

```
TensorFlow 1.x selected.
```

```
[ ] import tensorflow
    print(tensorflow.__version__)
```

```
1.15.2
```

```
import tensorflow as tf
from tensorflow.python.layers.core import Dense
import tensorflow.contrib.keras as keras
from keras.preprocessing.sequence import pad_sequences
import os
import numpy as np
import pandas as pd
from pandas import DataFrame
import time
import random
from PIL import Image
from trec_eval import trec_eval
import nltk
import copy
```

```
WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
  * https://github.com/tensorflow/addons
  * https://github.com/tensorflow/io (for I/O related ops)
If you depend on functionality not listed there, please file an issue.

Using TensorFlow backend.
```

```
[5]  os.environ['CUDA_VISIBLE_DEVICES'] = '0'
     config = tf.ConfigProto()
     config.gpu_options.allow_growth = True
```

```
random.seed(1)
np.random.seed(1)
```

USEFUL FUNCTIONS:

```
[7]  def read_comments(file_name):
         with open(file_name,'r') as f:
             file_content = f.readlines()
         comments = []
         for line in file_content:
             comments.append(line[:-1].split())
         return comments
```

```
def build_vocab(file_name,min_num):
    with open(file_name,'r') as f:
        file_content = f.readlines()
    word_to_int = {}
    int_to_word = {}
    words_num = 0
    for line in file_content:
        line = line[:-1].split('\t')
        if int(line[2]) >= min_num:
            word_to_int[line[1]] = int(line[0])
            int_to_word[int(line[0])] = line[1]
            words_num += 1
        else:
            break
    word_to_int['<PAD>'] = 0
    word_to_int['<UNK>'] = words_num+1
    word_to_int['<GO>'] = words_num+2
    word_to_int['<EOS>'] = words_num+3
    int_to_word[0] = '<PAD>'
    int_to_word[words_num+1] = '<UNK>'
    int_to_word[words_num+2] = '<GO>'
    int_to_word[words_num+3] = '<EOS>'
    return word_to_int,int_to_word
```

```
def convert_comments(comments,word_to_int,int_to_word):
    comments_to_int = []
    for comment in comments:
        comment_to_int = [word_to_int[word] if word_to_int.get(word) != None else word_to_int['<UNK>'] for word in comment]
        comment_to_int.insert(0,word_to_int['<GO>'])
        comment_to_int.append(word_to_int['<EOS>'])
        comments_to_int.append(comment_to_int)
    return comments_to_int
```

```
[10] def negative_samples(num_samples,toplist,downlist,combinationlist):
        sampledata = []
        num = 0
        while num < num_samples:
            top = random.sample(toplist,1)[0]
            down = random.sample(downlist,1)[0]
            if top+down not in combinationlist:
                sampledata.append((top,down,-1))
                num += 1
        return sampledata
```

```
[11] def pad_batch(batch,pad_int):
        max_length = max([len(comment) for comment in batch])
        pad_batch = pad_sequences(batch,maxlen=max_length,value=pad_int,padding='post')
        return pad_batch
```

```
def batch_to_input(batch,comments,imglist,topidlist,downidlist,pad_int):
    img1 = []#for top
    img2 = []#for down
    img1id = []
    img2id = []
    label = []
    sequence = []
    sequence_length = []
    weight = []
    for instance in batch:
        img1.append(imglist[instance[0]])
        img2.append(imglist[instance[1]])
        img1id.append(topidlist[instance[0]])
        img2id.append(downidlist[instance[1]])
        commentid = instance[2]
        if commentid == -1:
            label.append([1,0])
            weight.append(0)
        else:
            label.append([0,1])
            weight.append(1)
        sequence.append(comments[commentid])
        sequence_length.append(len(comments[commentid])-1)
    sequence = pad_batch(sequence,pad_int)
    sequence_input = sequence[:,:-1]
    sequence_output = sequence[:,1:]
    max_sequence_length = np.max(sequence_length)
    return np.array(img1),np.array(img2),np.array(img1id),np.array(img2id),np.array(label),sequence_input,sequence_output,sequence_length,max_sequence_length
```

```
[ ] def get_batches(data,batch_size,comments,toplist,downlist,combinationlist,imglist,topidlist,downidlist,pad_int):
        datacopy = copy.copy(data)
        datacopy.extend(negative_samples(len(datacopy),toplist,downlist,combinationlist))
        random.shuffle(datacopy)
        for batch_i in range(0,len(datacopy)//batch_size+1):
            start_i = batch_i*batch_size
            batch = datacopy[start_i:start_i+batch_size]
            yield batch_to_input(batch,comments,imglist,topidlist,downidlist,pad_int)
```

```
[ ] def build_evaluation_batch(fixitem,itemlist,state,imglist,topidlist,downidlist):
        img1 = []
        img2 = []
        img1id = []
        img2id = []
        if state == 0:#top,downs
            for item in itemlist:
                img1.append(imglist[fixitem])
                img2.append(imglist[item])
                img1id.append(topidlist[fixitem])
                img2id.append(downidlist[item])
        if state == 1:#down,tops
            for item in itemlist:
                img1.append(imglist[item])
                img2.append(imglist[fixitem])
                img1id.append(topidlist[item])
                img2id.append(downidlist[fixitem])
        return np.array(img1),np.array(img2),np.array(img1id),np.array(img2id)
```

```
[15] def accuracy(label,prediction):
        return (label.argmax(axis=1) == prediction.argmax(axis=1)).sum()/float(len(label))
```

```
def prepare_evaluation(data_path,comments,int_to_word,word_to_int):
    with open(data_path,'r') as f:
        content = f.readlines()
    data = {}
    orderlist = []
    model_comments = {}
    labellist = {}
    query_number = 0
    for line in content:
        line = line[:-1].split('\t')
        if data.get(line[0]) != None:
            data[line[0]].append(line[1])
        else:
            data[line[0]] = [line[1]]
            labellist[query_number] = {}
            query_number += 1
            orderlist.append(line[0])
        if int(line[2]) == 1:
            model_comments[(line[0],line[1])] = [id_seq_to_word_seq(comments[int(comment)],int_to_word,word_to_int['<EOS>']).split()[1:-1] for comment in lin
            labellist[query_number-1][line[1]] = 1
        else:
            labellist[query_number-1][line[1]] = 0
    return data,orderlist,model_comments,labellist
```

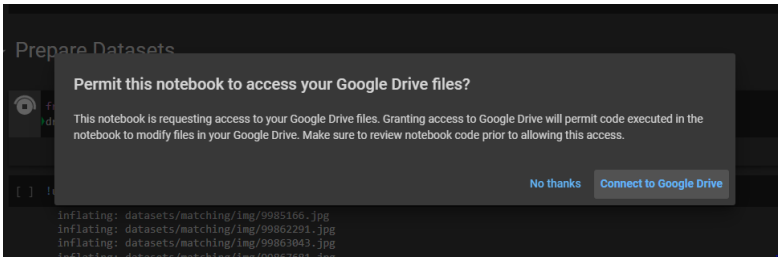
```
[ ] def trec_evaluation(qrel_file_path,trec_file_path,trec):
        with open(trec_file_path,'w') as f:
            i = 0
            while i < len(trec):
                j = 0
                while j < len(trec[i]):
                    f.write(str(i)+' '+'Q0 '+trec[i][j][0]+' '+str(j+1)+' '+str(trec[i][j][1])+' '+'Exp'+'\n')
                    j += 1
                i += 1
        result = trec_eval(qrel_file_path,trec_file_path)
        print(result)
        return result
```

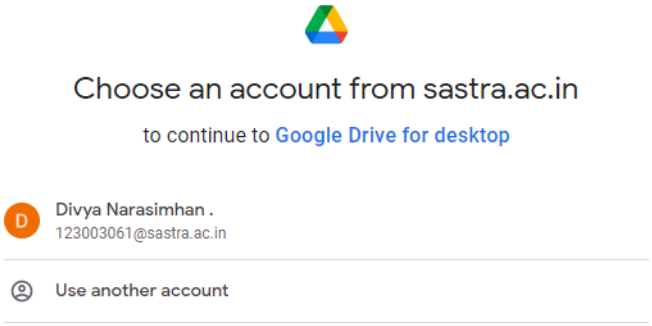
```
[ ] def bleu_evalaution(model_comments,system_comments,beamsearch):
        select = {}
        bleus = []
        if beamsearch:
            for combination,comments in system_comments.items():
                scores = []
                for comment in comments:
                    scores.append(nltk.translate.bleu_score.sentence_bleu(model_comments[combination],comment,weights=[1.0]))
                scores = np.array(scores)
                bleus.append(scores.max())
                select[combination] = scores.argmax()#we only select the best for evaluation
        else:
            for combination,comment in system_comments.items():
                bleus.append(nltk.translate.bleu_score.sentence_bleu(model_comments[combination],comment,weights=[1.0]))
        bleus = np.array(bleus)
        print(bleus.mean())
        return bleus.mean(),select
```

```
def auc_evaluation(labellist,trec):
    query_number = 0
    record = []
    while query_number < len(trec):
        negative = 0
        temp = []
        for combination in trec[query_number]:
            if labellist[query_number][combination[0]] == 1:
                temp.append(negative)
            else:
                negative += 1
        record.extend([(negative-val)/float(negative) for val in temp])
        query_number += 1
    auc = np.array(record).mean()
    print(auc)
    return auc
```

MOUNTING GDRIVE:

```
[ ]  from google.colab import drive
     drive.mount('/content/drive')
```





PREPARING DATASETS:

```
!unzip /content/drive/MyDrive/datasets.zip
  inflating: datasets/matching/img/9985166.jpg
  inflating: datasets/matching/img/99862291.jpg
  inflating: datasets/matching/img/99863043.jpg
  inflating: datasets/matching/img/99867681.jpg
  inflating: datasets/matching/img/99884984.jpg
  inflating: datasets/matching/img/99885091.jpg
  inflating: datasets/matching/img/99894456.jpg
  inflating: datasets/matching/img/99913288.jpg
  inflating: datasets/matching/img/99913380.jpg
  inflating: datasets/matching/img/99913468.jpg
  inflating: datasets/matching/img/99919728.jpg
  inflating: datasets/matching/img/99920063.jpg
  inflating: datasets/matching/img/99928029.jpg
  inflating: datasets/matching/img/99931031.jpg
  inflating: datasets/matching/img/99931198.jpg
  inflating: datasets/matching/img/99932834.jpg
  inflating: datasets/matching/img/99933024.jpg
  inflating: datasets/matching/img/99933207.jpg
  inflating: datasets/matching/img/99935163.jpg
```

```
toplist = []
topidlist = {}
with open('/content/drive/MyDrive/Mini_project/CG/train/newtoplist.dat','r') as f:
    content = f.readlines()
for line in content:
    line = line[:-1]
    toplist.append(line)
    topidlist[line] = len(topidlist)
toplist
```

```
['136011468',
 '165010182',
 '139223042',
 '190436286',
 '162991690',
 '125552516',
 '154674432',
 '190977941',
 '171692610',
 '195788221',
 '193543471',
 '187593522',
 '174141173',
 '151460702',
```

```
[18] toplist[214]= '194796874'

[19] topidlist['194796874'] = topidlist['19479687']
     del topidlist['19479687']

print(topidlist['194796874'])
```

```
214
```

```
downlist = []
downidlist = {}
with open('/content/drive/MyDrive/Mini_project/CG/train/newdownlist.dat','r') as f:
    content = f.readlines()
for line in content:
    line = line[:-1]
    downlist.append(line)
    downidlist[line] = len(downidlist)
downlist
```

```
['161425425',
 '183639308',
 '149494468',
 '185256499',
 '187619920',
 '192304171',
 '154908928',
 '195127827',
 '201020544',
 '184706162',
 '150894162',
 '181131950',
 '179103801',
 '189834308',
```

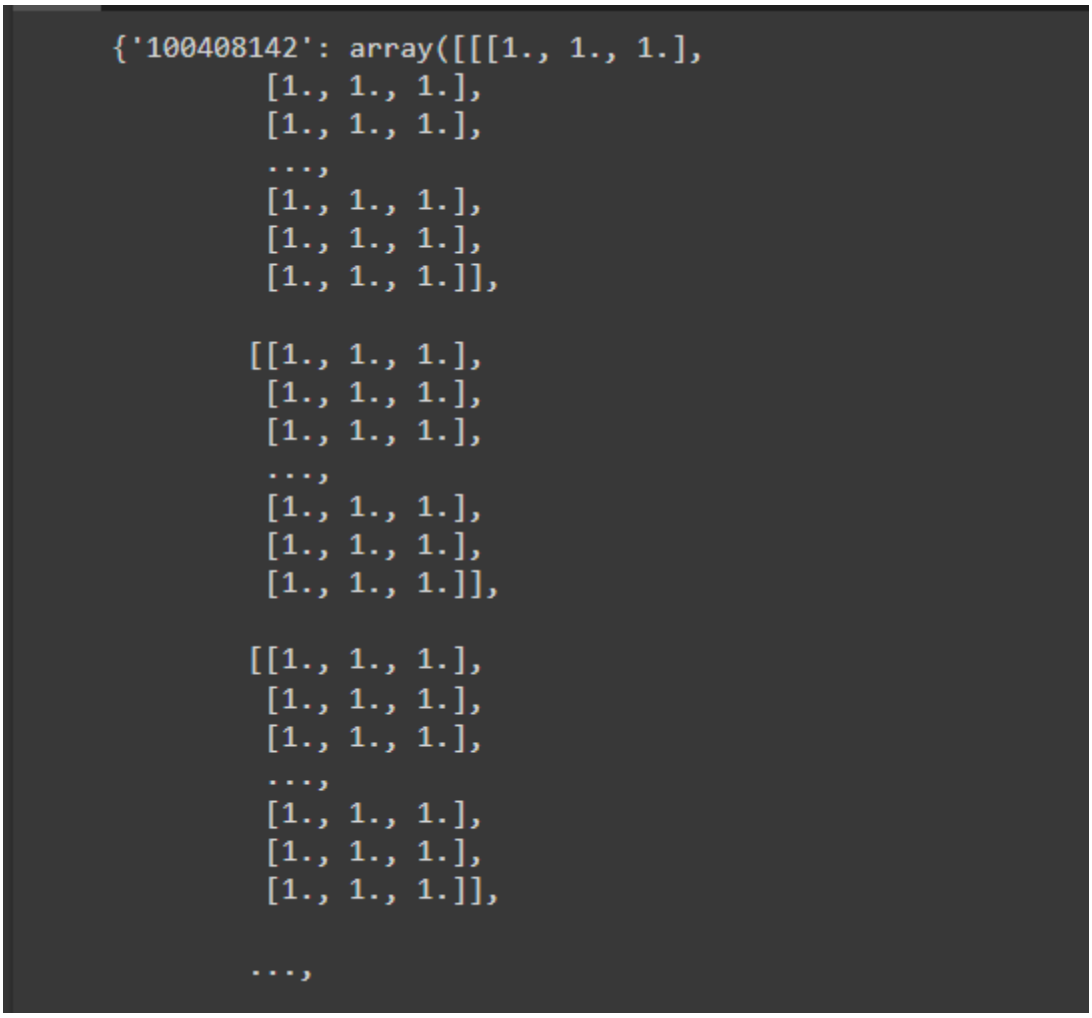
```
downidlist
```

```
{'101347443': 134,
 '103449172': 84,
 '106239071': 104,
 '109344972': 96,
 '113657786': 118,
 '114499573': 44,
 '114600316': 143,
 '116289573': 193,
 '117016854': 130,
 '119101477': 76,
 '119380232': 14,
 '120839784': 214,
 '120960130': 202,
```

```
combinationlist = set()
with open('/content/drive/MyDrive/Mini_project/CG/train/newcombinationlist.dat','r') as f:
    content = f.readlines()
for line in content:
    line = line[:-2].split('\t')
    combinationlist.add(line[0]+line[1])
combinationlist
```

```
{'11385229919635234',
 '12555251618713250',
 '13601146813562891',
 '13601146814481685',
 '13601146816676619',
 '13601146818202819',
 '13601146819358698',
 '13922304214447507',
 '14868494814096459',
 '15099990615212628',
 '15146070210934497',
 '15467443219604954',
 '15752815416142542',
```

```
imglist = {}
for img_idx in toplist:
    img = Image.open('/content/datasets/CG/img/'+img_idx+'.jpg')
    if img.mode != 'RGB':
        img = img.convert('RGB')
    img = np.array(img)
    img = img/255.0
    imglist[img_idx] = img
for img_idx in downlist:
    img = Image.open('/content/datasets/CG/img/'+img_idx+'.jpg')
    if img.mode != 'RGB':
        img = img.convert('RGB')
    img = np.array(img)
    img = img/255.0
    imglist[img_idx] = img
imglist
```



BUILDING THE MODEL:

```
def get_input():
    img1 = tf.placeholder(tf.float32,[None,150,150,3],'img1')
    img2 = tf.placeholder(tf.float32,[None,150,150,3],'img2')
    img1id = tf.placeholder(tf.int32,[None,],'img1id')
    img2id = tf.placeholder(tf.int32,[None,],'img2id')
    label = tf.placeholder(tf.float32,[None,2],'label')
    learning_rate = tf.placeholder(tf.float32,[],name='learning_rate')
    keep_prob = tf.placeholder(tf.float32,[],name='keep_prob')
    return img1,img2,img1id,img2id,label,learning_rate,keep_prob
```

```
def extractor(img):
    conv1 = keras.layers.Conv2D(filters=32,kernel_size=(3,3),strides=(1,1),padding='same',activation='relu',data_format='channels_last',
    conv2 = keras.layers.Conv2D(filters=32,kernel_size=(3,3),strides=(1,1),padding='same',activation='relu',data_format='channels_last',
    pool1 = keras.layers.MaxPool2D(pool_size=(16,16),padding='same')(conv1)
    pool2 = keras.layers.MaxPool2D(pool_size=(16,16),padding='same')(conv2)
    concat = keras.layers.Concatenate(axis=-1)([pool1,pool2])
    globalpool = keras.layers.GlobalAveragePooling2D()(concat)
    return concat,globalpool
```

```
def image_to_image_attention(conv,globalpool):#conv=[batch_size,14,14,64], globalpool=[batch_size,64]
    weights1 = tf.get_variable('weights1',shape=[64,64],initializer=tf.contrib.layers.xavier_initializer(uniform=False))
    weights2 = tf.get_variable('weights2',shape=[64,64],initializer=tf.contrib.layers.xavier_initializer(uniform=False))
    weights3 = tf.get_variable('weights3',shape=[64,1],initializer=tf.contrib.layers.xavier_initializer(uniform=False))
    attn_from = tf.matmul(globalpool,weights1)#attn_form=[batch_size,64]
    features = keras.layers.Reshape([-1,64])(conv)#features=[batch_size,196,64]
    attn_to = tf.matmul(tf.reshape(features,[-1,64]),weights2)#tf.reshape(features,[-1,64])=[batch_size*196,64], attn_to=[batch_size*196,64]
    attn_from  = tf.expand_dims(attn_from,1)#attn_from=[batch_size,1,64]
    attn_to = tf.reshape(attn_to,tf.shape(features))#attn_to=[batch_size,196,64]
    attn_logit = tf.add(attn_from,attn_to)#attn_logit=[batch_size,196,64]
    attn_logit = tf.reshape(attn_logit,[-1,64])#attn_logit=[batch_size*196,64]
    attn_logit = tf.tanh(attn_logit)
    attn_weight = tf.matmul(attn_logit,weights3)#attn_weight=[batch_size*196,1]
    attn_weight = tf.reshape(attn_weight,shape=[tf.shape(conv)[0],tf.shape(conv)[1]*tf.shape(conv)[2]])#attn_weight=[batch_size,196]
    attn_weight = tf.nn.softmax(attn_weight,name='attention_img2img')
    attn_weight = tf.expand_dims(attn_weight,-1)#attn_weight=[batch_size,196,1]
    attn_conv = tf.multiply(features,attn_weight)#attn_conv=[batch_size,196,64]
    attn_conv = tf.reduce_sum(attn_conv,axis=1)#attn_conv=[batch_size,64]
    return features,attn_conv#e=v^Ttanh(W1s+W2h), a=softmax(e)
```

```
[28] def img2vec(conv):
         extractor_output = keras.layers.Dense(300,activation='relu',kernel_initializer='glorot_normal')(conv)
         return extractor_output
```

```
def img_embedding(img1id,img2id):
    top_embedding_matrix = tf.get_variable('top_embedding_matrix',shape=[len(toplist),embedding_size],initializer=tf.contrib.layers.xavier_init
    down_embedding_matrix = tf.get_variable('down_embedding_matrix',shape=[len(downlist),embedding_size],initializer=tf.contrib.layers.xavier_i
    img1_embedding = tf.nn.embedding_lookup(top_embedding_matrix,img1id)
    img2_embedding = tf.nn.embedding_lookup(down_embedding_matrix,img2id)
    return img1_embedding,img2_embedding
```

```
[30] def classifier(extractor_output,keep_prob):
         dense = keras.layers.Dense(256,activation='relu',kernel_initializer='glorot_normal')(extractor_output)
         dropout = tf.nn.dropout(dense,keep_prob)
         classifier_output = keras.layers.Dense(2,activation='softmax',kernel_initializer='glorot_normal')(dropout)
         return classifier_output
```

```
def loss(classifier_output,label):
    classifier_loss = tf.reduce_mean(tf.contrib.keras.losses.categorical_crossentropy(label,classifier_output),name='classifier_loss')
    tv = tf.trainable_variables()
    reg_loss = tf.reduce_sum([tf.nn.l2_loss(v) for v in tv])
    loss = tf.add(classifier_loss,0.00001*reg_loss,name='loss')
    return loss
```

```
[32] def optimizer(loss,learning_rate):
         optimizer = tf.train.AdamOptimizer(learning_rate)
         gradients = optimizer.compute_gradients(loss)
         capped_gradients = [(tf.clip_by_value(grad,-5.,5.),var) for grad,var in gradients if grad is not None]
         train_op = optimizer.apply_gradients(capped_gradients)
         return train_op
```

```
def prediction(classifier_output):
    prediction = tf.identity(classifier_output,name='prediction')
    return prediction
```

```
embedding_size = 300
train_graph = tf.Graph()
with train_graph.as_default():
    tf.set_random_seed(1)
    with tf.name_scope('inputs'):
        img1,img2,img1id,img2id,label,learning_rate,keep_prob = get_input()
    with tf.name_scope('extractor'):
        with tf.variable_scope('extractor'):
            conv_img1,globalpool_img1 = extractor(img1)
        with tf.variable_scope('extractor',reuse=True):
            conv_img2,globalpool_img2 = extractor(img2)
        with tf.variable_scope('image_to_image_attention'):
            features_img1,attn_conv_img1 = image_to_image_attention(conv_img1,globalpool_img2)
        with tf.variable_scope('image_to_image_attention',reuse=True):
            features_img2,attn_conv_img2 = image_to_image_attention(conv_img2,globalpool_img1)
        with tf.variable_scope('img2vec'):
            extractor_output_img1 = img2vec(attn_conv_img1)
        with tf.variable_scope('img2vec',reuse=True):
            extractor_output_img2 = img2vec(attn_conv_img2)
        with tf.variable_scope('img_embedding'):
            img1_embedding,img2_embedding = img_embedding(img1id,img2id)
        encoder_output = tf.concat([features_img1,features_img2],axis=1,name='encoder_output')
        extractor_output = tf.concat([extractor_output_img1,extractor_output_img2,img1_embedding,img2_embedding],axis=1,
```

```
    with tf.name_scope('prediction'):
        prediction = prediction(classifier_output)
    with tf.name_scope('loss'):
        loss = loss(classifier_output,label)
    with tf.name_scope('optimizer'):
        train_op = optimizer(loss,learning_rate)

WARNING:tensorflow:From /tensorflow-1.15.2/python3.7/tensorflow_core/python/ops/resource_variable_ops.py:1630: calling BaseResourceVariable.__in
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From <ipython-input-30-9ef19db2e43e>:3: calling dropout (from tensorflow.python.ops.nn_ops) with keep_prob is deprecated and
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
```

TRAINING THE MODEL:

```
[35] with open('/content/drive/MyDrive/Mini_project/CG/train/newtraindata.dat','r') as f:#
         content = f.readlines()
     traindata = []
     for line in content:
         line = line[:-1].split('\t')
         traindata.append((line[0],line[1],1))
     len(traindata)

     199
```

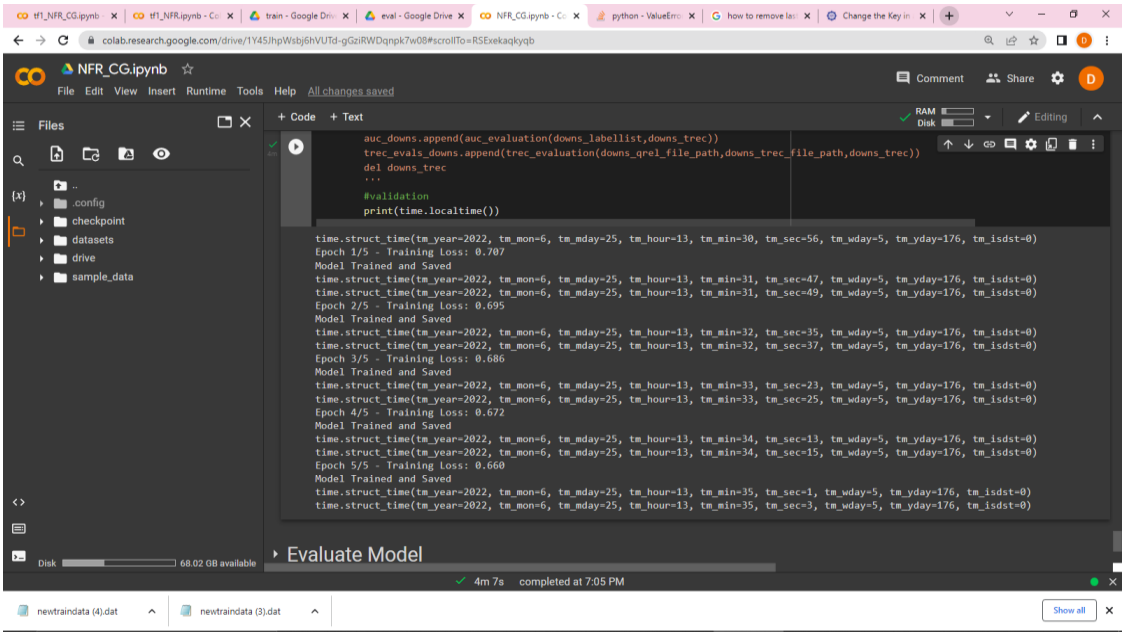
```
type(traindata[1])
#temp = traindata[198]
temp = traindata.pop()
temp
tup = ('179389915', '190945208', 1)
traindata.append(tup)
#traindata
```

```
tops_qrel_file_path = '/content/drive/MyDrive/Mini_project/CG/eval/devdata_tops_qrel.dat'
tops_trec_file_path = '/content/drive/MyDrive/Mini_project/CG/eval/devdata_tops_trec.dat'
#downs_qrel_file_path = 'evaluation/devdata_downs_qrel.dat'
#downs_trec_file_path = 'evaluation/devdata_downs_trec.dat'
```

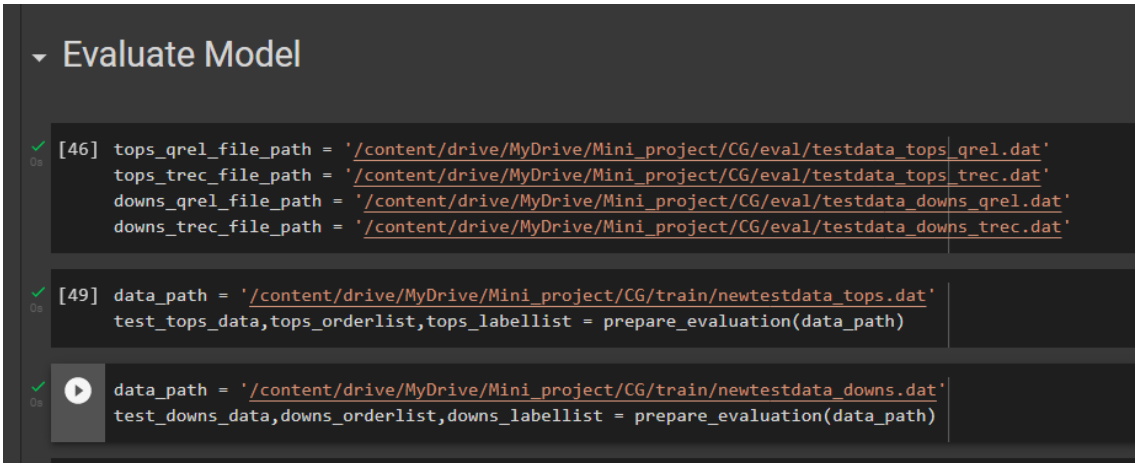
```
[40] data_path = '/content/drive/MyDrive/Mini_project/CG/train/newdevdata_tops.dat'
     dev_tops_data,tops_orderlist,tops_labellist = prepare_evaluation(data_path)
     #tops_orderlist

[41] #data_path = 'dataset/devdata_downs.dat'
     #dev_downs_data,downs_orderlist,downs_labellist = prepare_evaluation(data_path)

[42] lr = 0.001
     epochs = 5
     rate = 1.0
     batch_size = 64

     cost_list = []
     auc_tops = []
     trec_evals_tops = []
     #trec_evals_downs = []
     #auc_downs = []
```

```
checkpoint = 'checkpoint/trained_model.ckpt'
with tf.Session(graph=train_graph,config=config) as sess:
    writer = tf.summary.FileWriter('checkpoint/',sess.graph)
    saver = tf.train.Saver(max_to_keep=30)
    sess.run(tf.global_variables_initializer())
    print(time.localtime())
    classifier_loss = train_graph.get_tensor_by_name('loss/classifier_loss:0')
    for epoch in range(epochs):
        train_cost = 0
        temp_cost_list = []
        step = 0
        for _,(x_i1,x_i2,x_id1,x_id2,y_l) in enumerate(get_batches(traindata,batch_size,toplist,downlist,combinationlist,imglist,topidlist,down
            _,cost = sess.run([train_op,classifier_loss],{img1:x_i1,img2:x_i2,img1id:x_id1,img2id:x_id2,label:y_l,learning_rate:lr,keep_prob:ra
            train_cost += cost
            step += 1
            if step%1000 == 0:
                temp_cost_list.append(train_cost/step)
                print(str(train_cost/step)+' '+'pass!')
        temp_cost_list.append(train_cost/step)
        cost_list.append(temp_cost_list)
        print('Epoch {}/{} - Training Loss: {:.3f}'.format(epoch+1,epochs,train_cost/step))
        saver.save(sess,checkpoint,global_step=epoch+1)
        print('Model Trained and Saved')
        print(time.localtime())
        #validation
        tops_trec = {}
        query_number = 0
```

```
        query_number = 0
        step = 0
        for top in tops_orderlist:
            downsoftop = dev_tops_data[top]
            probabilitylist = {}
            for batch_i in range(len(downsoftop)//batch_size+1):
                start_i = batch_i*batch_size
                downs = downsoftop[start_i:start_i+batch_size]
                x_i1,x_i2,x_id1,x_id2 = build_evaluation_batch(top,downs,0,imglist,topidlist,downidlist)
                seq_len = [30]*len(x_i1)
                prob = sess.run(prediction,{img1:x_i1,img2:x_i2,img1id:x_id1,img2id:x_id2,keep_prob:1.0})
                j = 0
                for down in downs:
                    probabilitylist[down] = prob[j][1]
                    j += 1
                step += 1
                if step%1000 == 0:
                    print('pass!')
            tops_trec[query_number] = sorted(probabilitylist.items(),key=lambda item:item[1],reverse=True)
            del probabilitylist,downsoftop
            query_number += 1
        auc_tops.append(auc_evaluation(tops_labellist,tops_trec))
        trec_evals_tops.append(trec_evaluation(tops_qrel_file_path,tops_trec_file_path,tops_trec))
        del tops_trec
        '''
```



EVALUVATING THE MODEL:



```
print(time.localtime())
checkpoint = 'checkpoint/trained_model.ckpt-2'
test_graph = tf.Graph()
with tf.Session(graph=test_graph,config=config) as sess:
    loader = tf.train.import_meta_graph(checkpoint+'.meta')
    loader.restore(sess,checkpoint)
    img1 = test_graph.get_tensor_by_name('inputs/img1:0')
    img2 = test_graph.get_tensor_by_name('inputs/img2:0')
    img1id = test_graph.get_tensor_by_name('inputs/img1id:0')
    img2id = test_graph.get_tensor_by_name('inputs/img2id:0')
    keep_prob = test_graph.get_tensor_by_name('inputs/keep_prob:0')
    prediction = test_graph.get_tensor_by_name('prediction/prediction:0')
    batch_size = 64
    tops_trec = {}
    query_number = 0
    step = 0
    for top in tops_orderlist:
        downsoftop = test_tops_data[top]
        probabilitylist = {}
        for batch_i in range(len(downsoftop)//batch_size+1):
            start_i = batch_i*batch_size
            downs = downsoftop[start_i:start_i+batch_size]
            x_i1,x_i2,x_id1,x_id2 = build_evaluation_batch(top,downs,0,imglist,topidlist,downidlist)
            prob = sess.run(prediction,{img1:x_i1,img2:x_i2,img1id:x_id1,img2id:x_id2,keep_prob:1.0})
```

```
        for batch_i in range(len(downsoftop)//batch_size+1):
            start_i = batch_i*batch_size
            downs = downsoftop[start_i:start_i+batch_size]
            x_i1,x_i2,x_id1,x_id2 = build_evaluation_batch(top,downs,0,imglist,topidlist,downidlist)
            prob = sess.run(prediction,{img1:x_i1,img2:x_i2,img1id:x_id1,img2id:x_id2,keep_prob:1.0})
            j = 0
            for down in downs:
                probabilitylist[down] = prob[j][1]
                j += 1
            step += 1
            if step%1000 == 0:
                print('pass!')
        tops_trec[query_number] = sorted(probabilitylist.items(),key=lambda item:item[1],reverse=True)
        del probabilitylist,downsoftop
        query_number += 1
    auc_evaluation(tops_labellist,tops_trec)
    trec_evaluation(tops_qrel_file_path,tops_trec_file_path,tops_trec)
    del tops_trec
    downs_trec = {}
    query_number = 0
    step = 0
    for down in downs_orderlist:
        topsofdown = test_downs_data[down]
        probabilitylist = {}
        for batch_i in range(len(topsofdown)//batch_size+1):
            start_i = batch_i*batch_size
            tops = topsofdown[start_i:start_i+batch_size]
            x_i1,x_i2,x_id1,x_id2 = build_evaluation_batch(down,tops,1,imglist,topidlist,downidlist)
            prob = sess.run(prediction,{img1:x_i1,img2:x_i2,img1id:x_id1,img2id:x_id2,keep_prob:1.0})
```

```
    for down in downs_orderlist:
        topsofdown = test_downs_data[down]
        probabilitylist = {}
        for batch_i in range(len(topsofdown)//batch_size+1):
            start_i = batch_i*batch_size
            tops = topsofdown[start_i:start_i+batch_size]
            x_i1,x_i2,x_id1,x_id2 = build_evaluation_batch(down,tops,1,imglist,topidlist,downidlist)
            prob = sess.run(prediction,{img1:x_i1,img2:x_i2,img1id:x_id1,img2id:x_id2,keep_prob:1.0})
            j = 0
            for top in tops:
                probabilitylist[top] = prob[j][1]
                j += 1
            step += 1
            if step%1000 == 0:
                print('pass!')
        downs_trec[query_number] = sorted(probabilitylist.items(),key=lambda item:item[1],reverse=True)
        del probabilitylist,topsofdown
        query_number += 1
    auc_evaluation(downs_labellist,downs_trec)
    trec_evaluation(downs_qrel_file_path,downs_trec_file_path,downs_trec)
    del downs_trec
print(time.localtime())
```

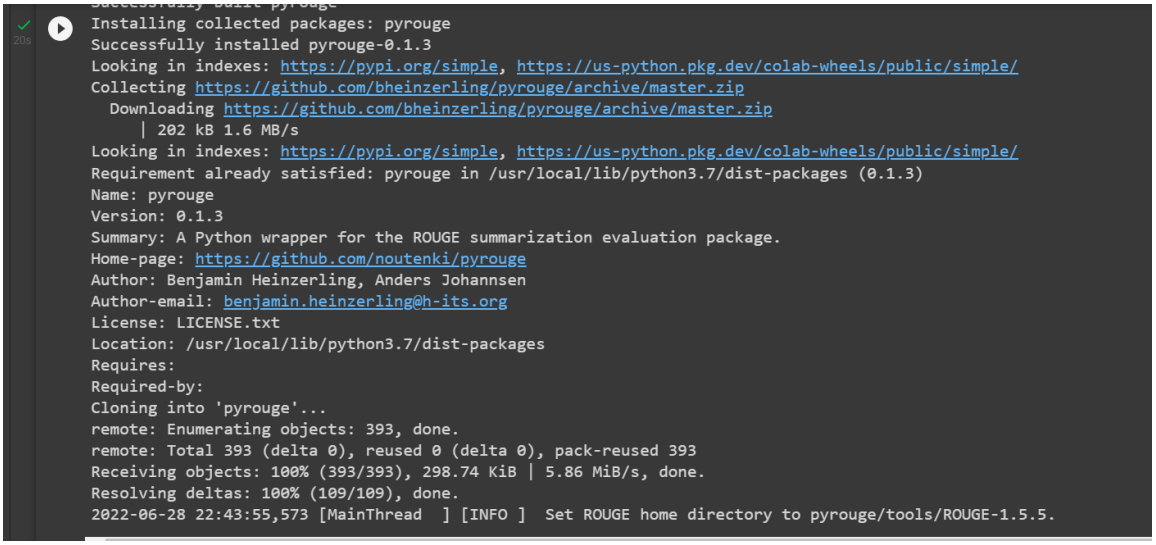
```
time.struct_time(tm_year=2022, tm_mon=6, tm_mday=25, tm_hour=17, tm_min=44, tm_sec=10, tm_wday=5, tm_yday=176, tm_isdst=0)
INFO:tensorflow:Restoring parameters from checkpoint/trained_model.ckpt-2
```

CHAPTER 5

ROGUE EVALUVATION

IMPORTING AND INSTALLATION OF PYROUGE AND ROUGE155:

```
!pip install pyrouge --upgrade
!pip install https://github.com/bheinzerling/pyrouge/archive/master.zip
!pip install pyrouge
!pip show pyrouge
!git clone https://github.com/andersjo/pyrouge.git
from pyrouge import Rouge155
!pyrouge_set_rouge_path 'pyrouge/tools/ROUGE-1.5.5'

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyrouge
  Downloading pyrouge-0.1.3.tar.gz (60 kB)
     |████████████████████████████████| 60 kB 3.5 MB/s
Building wheels for collected packages: pyrouge
  Building wheel for pyrouge (setup.py) ... done
  Created wheel for pyrouge: filename=pyrouge-0.1.3-py3-none-any.whl size=191621 sha256=569395b599e261f2ad0f0f377cbe5dca2eaf22fdcf7f57aa3bc20dbbe85dbd
  Stored in directory: /root/.cache/pip/wheels/68/35/6a/ffb9a1f51b2b00fee42e7f67f5a5d8e10c67d048cda09ccd57
Successfully built pyrouge
Installing collected packages: pyrouge
Successfully installed pyrouge-0.1.3
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting https://github.com/bheinzerling/pyrouge/archive/master.zip
  Downloading https://github.com/bheinzerling/pyrouge/archive/master.zip
     | 202 kB 1.6 MB/s
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pyrouge in /usr/local/lib/python3.7/dist-packages (0.1.3)
Name: pyrouge
```



```
Installing collected packages: pyrouge
Successfully installed pyrouge-0.1.3
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting https://github.com/bheinzerling/pyrouge/archive/master.zip
  Downloading https://github.com/bheinzerling/pyrouge/archive/master.zip
     | 202 kB 1.6 MB/s
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pyrouge in /usr/local/lib/python3.7/dist-packages (0.1.3)
Name: pyrouge
Version: 0.1.3
Summary: A Python wrapper for the ROUGE summarization evaluation package.
Home-page: https://github.com/noutenki/pyrouge
Author: Benjamin Heinzerling, Anders Johannsen
Author-email: benjamin.heinzerling@h-its.org
License: LICENSE.txt
Location: /usr/local/lib/python3.7/dist-packages
Requires:
Required-by:
Cloning into 'pyrouge'...
remote: Enumerating objects: 393, done.
remote: Total 393 (delta 0), reused 0 (delta 0), pack-reused 393
Receiving objects: 100% (393/393), 298.74 KiB | 5.86 MiB/s, done.
Resolving deltas: 100% (109/109), done.
2022-06-28 22:43:55,573 [MainThread  ] [INFO ]  Set ROUGE home directory to pyrouge/tools/ROUGE-1.5.5.
```

```
!sudo apt-get install libxml-parser-perl

Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  libnvidia-common-460
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  libauthen-sasl-perl libdata-dump-perl libencode-locale-perl
  libfile-listing-perl libfont-afm-perl libhtml-form-perl libhtml-format-perl
  libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl
  libhttp-cookies-perl libhttp-daemon-perl libhttp-date-perl
  libhttp-message-perl libhttp-negotiate-perl libio-html-perl
  libio-socket-ssl-perl liblwp-mediatypes-perl liblwp-protocol-https-perl
  libmailtools-perl libnet-http-perl libnet-smtp-ssl-perl libnet-ssleay-perl
  libtimedate-perl libtry-tiny-perl liburi-perl libwww-perl
  libwww-robotrules-perl netbase perl-openssl-defaults
Suggested packages:
  libdigest-hmac-perl libgssapi-perl libcrypt-ssleay-perl libauthen-ntlm-perl
The following NEW packages will be installed:
  libauthen-sasl-perl libdata-dump-perl libencode-locale-perl
  libfile-listing-perl libfont-afm-perl libhtml-form-perl libhtml-format-perl
  libhtml-parser-perl libhtml-tagset-perl libhtml-tree-perl
  libhttp-cookies-perl libhttp-daemon-perl libhttp-date-perl
  libhttp-message-perl libhttp-negotiate-perl libio-html-perl
```

```
%%shell
cd pyrouge/tools/ROUGE-1.5.5/data
rm WordNet-2.0.exc.db # only if exist
cd WordNet-2.0-Exceptions
rm WordNet-2.0.exc.db # only if exist

./buildExeptionDB.pl . exc WordNet-2.0.exc.db
cd ../
ln -s WordNet-2.0-Exceptions/WordNet-2.0.exc.db WordNet-2.0.exc.db
whackiest
```

USEFUL FUNCTIONS:

```
[4]  import numpy as np
     import pandas as pd
     from pandas import DataFrame
     from pyrouge import Rouge155
     import logging
```

```
[5]  def read_comments(file_name):
         with open(file_name,'r') as f:
             file_content = f.readlines()
         comments = []
         for line in file_content:
             comments.append(line[:-1].split())
         return comments
```

```python
def build_vocab(file_name,min_num):
    with open(file_name,'r') as f:
        file_content = f.readlines()
    word_to_int = {}
    int_to_word = {}
    words_num = 0
    for line in file_content:
        line = line[:-1].split('\t')
        if int(line[2]) >= min_num:
            word_to_int[line[1]] = int(line[0])
            int_to_word[int(line[0])] = line[1]
            words_num += 1
        else:
            break
    word_to_int['<UNK>'] = words_num+1
    int_to_word[words_num+1] = '<UNK>'
    return word_to_int,int_to_word
```

```python
[7] def convert_comments(comments,word_to_int,int_to_word):
        comments_to_int = []
        for comment in comments:
            comment_to_int = [word_to_int[word] if word_to_int.get(word) != None else word_to_int['<UNK>'] for word in comment]
            comments_to_int.append(comment_to_int)
        return comments_to_int

[8] def id_seq_to_word_seq(id_seq,id_vocab):
        return ' '.join([id_vocab[id] for id in id_seq])

def prepare_evaluation(data_path,comments,int_to_word,word_to_int):
    with open(data_path,'r') as f:
        content = f.readlines()
    model_comments = {}
    for line in content:
        line = line[:-1].split('\t')
        if int(line[2]) == 1:
            model_comments[(line[0],line[1])] = [id_seq_to_word_seq(comments[int(comment)],int_to_word) for comment in line[3].split('|')]
    return model_comments
```

```python
[10] def read_system_comments(data_path):
        system_comments = {}
        with open(data_path,'r') as f:
            content = f.readlines()
        for line in content:
            line = line.split('\t')
            system_comments[(line[0],line[1])] = line[2][:-1]
        return system_comments
```

```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[12] !unzip /content/drive/MyDrive/datasets.zip
```

```
  inflating: datasets/matching/img/91103749.jpg
  inflating: datasets/matching/img/91117110.jpg
  inflating: datasets/matching/img/91135989.jpg
  inflating: datasets/matching/img/91136011.jpg
  inflating: datasets/matching/img/91138945.jpg
  inflating: datasets/matching/img/91139990.jpg
  inflating: datasets/matching/img/91140417.jpg
  inflating: datasets/matching/img/91145288.jpg
  inflating: datasets/matching/img/91146812.jpg
  inflating: datasets/matching/img/91147829.jpg
  inflating: datasets/matching/img/91147881.jpg
  inflating: datasets/matching/img/91148317.jpg
  inflating: datasets/matching/img/91148794.jpg
  inflating: datasets/matching/img/91148813.jpg
  inflating: datasets/matching/img/91152285.jpg
  inflating: datasets/matching/img/91154701.jpg
  inflating: datasets/matching/img/91155131.jpg
  inflating: datasets/matching/img/91161650.jpg
  inflating: datasets/matching/img/91162795.jpg
  inflating: datasets/matching/img/91170171.jpg
  inflating: datasets/matching/img/91170176.jpg
  inflating: datasets/matching/img/91178542.jpg
  inflating: datasets/matching/img/91178544.jpg
  inflating: datasets/matching/img/91186722.jpg
```

PREPARING THE DATASETS:

```
[15] comments_path = '/content/drive/MyDrive/train/text.dat'
     vocab_path = '/content/drive/MyDrive/train/vocab.dat'
     min_num = 5
```

```
[16] comments = read_comments(comments_path)
     comments
```

```
 '!'],
['cute', 'outfit', 'dear', '!', 'have', 'a', 'great', 'week', 'ahead', '!'],
['outstanding',
 'in',
 'set',
 'layout',
 'and',
 'beautifully',
 'styled',
 'dressed',
 'up',
 'jeans',
 'look',
 '.'],
['simply', 'beautiful', '!'],
['wow',
 ',',
 'all',
 'the',
 'pieces',
 'work',
 'so',
 'well',
 'together',
```

```
[17] word_to_int,int_to_word = build_vocab(vocab_path,min_num)
     vocab_size = len(word_to_int)
```

```
[18] comments = convert_comments(comments,word_to_int,int_to_word)
     comments
```

```
[2802, 1900, 9115, 2727, 533, 16516, 1, 612, 1356, 2],
[220, 6, 3, 18, 2],
[853, 94, 2],
[20, 24, 19, 377, 2],
[870, 16, 2],
[4829, 58, 1],
[62, 1, 53, 6, 135, 70, 2],
[20, 24, 27, 31, 73, 16, 1],
[46, 1, 20, 24, 73, 1],
[60, 1, 62, 16, 24, 73, 1],
[20, 24, 59, 6, 1, 7, 125, 1],
[151, 4, 2820, 15, 1],
[19, 18, 19, 6, 19, 285, 420, 201, 232, 167, 50, 581, 2],
[20, 24, 10, 25, 6, 275, 437, 2],
[105, 7, 30, 3, 38, 1],
[62, 24, 73, 1],
[41, 1, 20, 296, 2],
[118, 99, 115, 8, 1, 20, 1],
[20, 4, 2515, 2],
[73, 20, 1],
[20, 24, 73, 2],
[50, 114, 1, 20, 1],
[139, 64, 2],
[20, 24, 25, 73, 16, 1],
[426, 64, 2],
[46, 121, 25, 20, 24, 377, 2],
```

```
[ ] model_tops_comments
```

```
{('100007206',
  '51329747'): ['thanks ! my favorite color ( of the week , anyway ) : - ) .', 'absolutely lovely white on white set ! good luck : ) .'],
 ('100063869',
  '115809124'): ["gorgeous set ! i 'm so sorry , i accidentally liked the set without seeing the text ; please remove me from your taglist ! thank you and i 'm sorry about the <UNK> .", 'pickles ! btw love your sets they really inspire me !', 'ahh this is amazing ! i live in canada , so thanksgiving was a few weeks ago , but my <UNK> came over and we celebrated hehe ; ) .', "pickles ( only if you want to , of course ) lololol i did nothing because we do n't celebrate thanksgiving where i 'm from haha .", 'pickles : ) set is amazing ahhh blue xx .', 'pickles c : .', 'pickles please c : .'],
 ('100063869', '137623366'): ['haha , girl you rock ; d .',
  'thank you ! :3 .',
  'thanks ! do too girl ! ; p .'],
 ('100063869', '176712253'): ["i 'm glad you like it ."],
 ('100063869', '64030343'): ['good luck ! an this set is soo prettyyyy u.u .',
  'its so cute tbh i wanted drew and claire to be together but i liked the ending it was very adorable : ) aww thanks : d ikr lilly collins is a babe .',
  'i love this layout a lot everything looks super cool n i love lily collins ahh < 333 good look on yr permit test ! xx .',
  'oh my god i read <UNK> too .',
  "i am the sun <UNK> you 're so cute xx .",
  'good luck hunny bun .',
  'thanks : ) ) & ily : ) .',
  'loving these colours ! so cute xx .',
  'this is so pretty ah love .',
  "tbh i did n't finish it bc it was kinda slow for me but they were v cute : ( all the food sounded so good .",
  'oh my goodness i loved <UNK> so much !'],
```

```
[21] data_path = '/content/drive/MyDrive/train/newtestdata_downs.dat'
     model_downs_comments = prepare_evaluation(data_path,comments,int_to_word,word_to_int)

[22] nfr_system_tops_comments_path = '/content/drive/MyDrive/Mini_project/matching/system comments/system_tops_comments.dat'
     nfr_system_downs_comments_path = '/content/drive/MyDrive/Mini_project/matching/system comments/system_downs_comments.dat'

[23] nfr_system_tops_comments = read_system_comments(nfr_system_tops_comments_path)
     nfr_system_downs_comments = read_system_comments(nfr_system_downs_comments_path)

[25] F = open('rouge/rouge_conf.xml','w')
     F.write('<ROUGE-EVAL version=\"1.55\">\n')
     index = 1
```

```
[26] for combination,comments in model_tops_comments.items():
         F.write('\t<EVAL ID=\"'+str(index)+'\">\n')
         F.write('\t\t<MODEL-ROOT>model</MODEL-ROOT>\n')
         F.write('\t\t<PEER-ROOT>system</PEER-ROOT>\n')
         F.write('\t\t<INPUT-FORMAT TYPE=\"SEE\">\n')
         F.write('\t\t</INPUT-FORMAT>\n')
         F.write('\t\t<PEERS>\n')
         with open('rouge/system/nfr.'+str(index)+'.txt','w') as f:
             if nfr_system_tops_comments.get(combination) != None:
                 f.write(nfr_system_tops_comments[combination])
             else:
                 f.write('')
         F.write('\t\t\t<P ID=\"nfr\">'+'nfr.'+str(index)+'.txt'+'</P>\n')
         F.write('\t\t</PEERS>\n')
         F.write('\t\t<MODELS>\n')
         i = 1
         for comment in comments:
             with open('rouge/model/common.'+str(i)+'.'+str(index)+'.txt','w') as f:
                 f.write(comment)
             F.write('\t\t\t<M ID=\"'+str(i)+'\">'+'common.'+str(i)+'.'+str(index)+'.txt'+'</M>\n')
             i += 1
         F.write('\t\t</MODELS>\n')
         F.write('\t</EVAL>\n')
         index += 1
```

```
[27] for combination,comments in model_downs_comments.items():
        F.write('\t<EVAL ID=\"'+str(index)+'\">\n')
        F.write('\t\t<MODEL-ROOT>model</MODEL-ROOT>\n')
        F.write('\t\t<PEER-ROOT>system</PEER-ROOT>\n')
        F.write('\t\t<INPUT-FORMAT TYPE=\"SEE\">\n')
        F.write('\t\t</INPUT-FORMAT>\n')
        F.write('\t\t<PEERS>\n')
        with open('rouge/system/nfr.'+str(index)+'.txt','w') as f:
            if nfr_system_downs_comments.get(combination) != None:
                f.write(nfr_system_downs_comments[combination])
            else:
                f.write('')
        F.write('\t\t\t<P ID=\"nfr\">'+'nfr.'+str(index)+'.txt'+'</P>\n')
        F.write('\t\t</PEERS>\n')
        F.write('\t\t<MODELS>\n')
        i = 1
        for comment in comments:
            with open('rouge/model/common.'+str(i)+'.'+str(index)+'.txt','w') as f:
                f.write(comment)
            F.write('\t\t\t<M ID=\"'+str(i)+'\">'+'common.'+str(i)+'.'+str(index)+'.txt'+'</M>\n')
            i += 1
        F.write('\t\t</MODELS>\n')
        F.write('\t</EVAL>\n')
        index += 1
```

CREATION OF ROUGE155 MODEL OBJECT:

```
[29] system_dir = 'rouge/system'
     model_dir = 'rouge/model'
     x = logging.ERROR
     r = Rouge155()
     r.convert_summaries_to_rouge_format(system_dir,system_dir)
     r.convert_summaries_to_rouge_format(model_dir,model_dir)

2022-06-29 02:27:45,955 [MainThread  ] [INFO ]  Processing files in rouge/system.
2022-06-29 02:27:45,959 [MainThread  ] [INFO ]  Processing nfr.9.txt.
2022-06-29 02:27:45,962 [MainThread  ] [INFO ]  Processing nfr.12.txt.
2022-06-29 02:27:45,966 [MainThread  ] [INFO ]  Processing nfr.7.txt.
2022-06-29 02:27:45,969 [MainThread  ] [INFO ]  Processing nfr.5.txt.
2022-06-29 02:27:45,971 [MainThread  ] [INFO ]  Processing nfr.6.txt.
2022-06-29 02:27:45,975 [MainThread  ] [INFO ]  Processing nfr.4.txt.
2022-06-29 02:27:45,977 [MainThread  ] [INFO ]  Processing nfr.16.txt.
2022-06-29 02:27:45,980 [MainThread  ] [INFO ]  Processing nfr.17.txt.
2022-06-29 02:27:45,982 [MainThread  ] [INFO ]  Processing nfr.3.txt.
2022-06-29 02:27:45,985 [MainThread  ] [INFO ]  Processing nfr.13.txt.
2022-06-29 02:27:45,988 [MainThread  ] [INFO ]  Processing nfr.15.txt.
2022-06-29 02:27:45,991 [MainThread  ] [INFO ]  Processing nfr.8.txt.
2022-06-29 02:27:45,994 [MainThread  ] [INFO ]  Processing nfr.14.txt.
2022-06-29 02:27:45,997 [MainThread  ] [INFO ]  Processing nfr.10.txt.
2022-06-29 02:27:45,999 [MainThread  ] [INFO ]  Processing nfr.11.txt.
2022-06-29 02:27:46,001 [MainThread  ] [INFO ]  Processing nfr.1.txt.
2022-06-29 02:27:46,004 [MainThread  ] [INFO ]  Processing nfr.2.txt.
2022-06-29 02:27:46,006 [MainThread  ] [INFO ]  Saved processed files to rouge/system.
2022-06-29 02:27:46,008 [MainThread  ] [INFO ]  Processing files in rouge/model.
2022-06-29 02:27:46,010 [MainThread  ] [INFO ]  Processing common.8.7.txt.
2022-06-29 02:27:46,013 [MainThread  ] [INFO ]  Processing common.8.4.txt.
2022-06-29 02:27:46,015 [MainThread  ] [INFO ]  Processing common.11.7.txt.
2022-06-29 02:27:46,017 [MainThread  ] [INFO ]  Processing common.11.4.txt.
2022-06-29 02:27:46,019 [MainThread  ] [INFO ]  Processing common.3.7.txt.
2022-06-29 02:27:46,021 [MainThread  ] [INFO ]  Processing common.1.16.txt.
2022-06-29 02:27:46,023 [MainThread  ] [INFO ]  Processing common.2.4.txt.
2022-06-29 02:27:46,026 [MainThread  ] [INFO ]  Processing common.1.9.txt.
```

ROUGE_CONF.XML - Generated by Code

```
<ROUGE-EVAL version="1.55">
    <EVAL ID="1">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.1.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.1.txt</M>
            <M ID="2">common.2.1.txt</M>
            <M ID="3">common.3.1.txt</M>
            <M ID="4">common.4.1.txt</M>
            <M ID="5">common.5.1.txt</M>
            <M ID="6">common.6.1.txt</M>
            <M ID="7">common.7.1.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="2">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.2.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.2.txt</M>
            <M ID="2">common.2.2.txt</M>
            <M ID="3">common.3.2.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="3">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.3.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.3.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="4">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.4.txt</P>
```

```
        </PEERS>
        <MODELS>
            <M ID="1">common.1.4.txt</M>
            <M ID="2">common.2.4.txt</M>
            <M ID="3">common.3.4.txt</M>
            <M ID="4">common.4.4.txt</M>
            <M ID="5">common.5.4.txt</M>
            <M ID="6">common.6.4.txt</M>
            <M ID="7">common.7.4.txt</M>
            <M ID="8">common.8.4.txt</M>
            <M ID="9">common.9.4.txt</M>
            <M ID="10">common.10.4.txt</M>
            <M ID="11">common.11.4.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="5">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.5.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.5.txt</M>
            <M ID="2">common.2.5.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="6">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.6.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.6.txt</M>
            <M ID="2">common.2.6.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="7">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.7.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.7.txt</M>
            <M ID="2">common.2.7.txt</M>
            <M ID="3">common.3.7.txt</M>
            <M ID="4">common.4.7.txt</M>
            <M ID="5">common.5.7.txt</M>
            <M ID="6">common.6.7.txt</M>
            <M ID="7">common.7.7.txt</M>
            <M ID="8">common.8.7.txt</M>
            <M ID="9">common.9.7.txt</M>
```

```
            <M ID="10">common.10.7.txt</M>
            <M ID="11">common.11.7.txt</M>
            <M ID="12">common.12.7.txt</M>
            <M ID="13">common.13.7.txt</M>
            <M ID="14">common.14.7.txt</M>
            <M ID="15">common.15.7.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="8">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.8.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.8.txt</M>
            <M ID="2">common.2.8.txt</M>
            <M ID="3">common.3.8.txt</M>
            <M ID="4">common.4.8.txt</M>
            <M ID="5">common.5.8.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="9">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.9.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.9.txt</M>
            <M ID="2">common.2.9.txt</M>
            <M ID="3">common.3.9.txt</M>
            <M ID="4">common.4.9.txt</M>
            <M ID="5">common.5.9.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="10">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.10.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.10.txt</M>
            <M ID="2">common.2.10.txt</M>
            <M ID="3">common.3.10.txt</M>
            <M ID="4">common.4.10.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="11">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
```

```
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.11.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.11.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="12">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.12.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.12.txt</M>
            <M ID="2">common.2.12.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="13">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.13.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.13.txt</M>
            <M ID="2">common.2.13.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="14">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.14.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.14.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="15">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.15.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.15.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="16">
```

```
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.16.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.16.txt</M>
        </MODELS>
    </EVAL>
    <EVAL ID="17">
        <MODEL-ROOT>model</MODEL-ROOT>
        <PEER-ROOT>system</PEER-ROOT>
        <INPUT-FORMAT TYPE="SEE">
        </INPUT-FORMAT>
        <PEERS>
            <P ID="nfr">nfr.17.txt</P>
        </PEERS>
        <MODELS>
            <M ID="1">common.1.17.txt</M>
        </MODELS>
    </EVAL>
</ROUGE-EVAL>
```

SAMPLE SYSTEM_DIR FILE:

nfr.1.txt

```
<html>
<head>
<title>dummy title</title>
</head>
<body bgcolor="white">
<a name="1">[1]</a> <a href="#1" id=1>love ! ! ! ! ! ! . . . . . .</a>
</body>
</html>
```

SAMPLE MODEL_DIR FILE:

common.1.14.txt

```
<html>
<head>
<title>dummy title</title>
</head>
<body bgcolor="white">
<a name="1">[1]</a> <a href="#1" id=1>perfect combo ! -- breaking rocks clothing .</a>
</body>
</html>
```

CHAPTER 6

CONCLUSION AND FUTURE PLANS

The Model studied explainable outfit recommendation and abstractive comment generation. The 2 main challenges found from previous works namely: the compatibility of fashion factors and the transformation between visual and textual information are concentrated and handled in this deep learning NOR framework. . The framework uses large real-world dataset, ExpFashion, including images, contextual metadata of items, and user comments.

In the experiments the effectiveness of NOR is shown and is demonstrated how signicant improvements are shown over state-of-the-art baselines in terms of MAP, MRR and AUC. And from the experiments , it is found that NOR achieves impressive ROUGE and BLEU scores with respect to human-written comments. The Framework also uses the mutual attention and cross-modality attention mechanisms for outfit recommendation and comment generation which have shown to give very good performance.

Future Plans

- As to future work, we plan to explore more fashion items in our dataset, e.g., hats, glasses and shoes.
- We also plan to alleviate the limitations of generating meaningless comments and short comments
- We also want to incorporate other mechanisms, such as an auto-encoder, to further improve the performance
- We would like to build a personalized outfit recommendation system.

CHAPTER 7

REFERENCES

References:

- Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin, "Explainable recommendation via multi-task learning in opinionated text data," in International Conference on Research and Development in Information Retrieval, 2018.
- Piji Li, Wai Lam, Lidong Bing, and Zihao Wang, "Deep recurrent generative decoder for abstractive text summarization," in Conference on Empirical Methods in Natural Language Processing, 2017, pp. 2091–2100.
- Vignesh Jagadeesh, Robinson Piramuthu, Anurag Bhardwaj, Wei Di, and Neel Sundaresan, "Large scale visual recommendations from street fashion images," in ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2014, pp. 1925–1934.