```
from google.colab import files
uploaded = files.upload()
```

```
Choose Files Mycharger B…pdated.csv
```

- **Mycharger Bulletien_updated.csv**(text/csv) - 299002 bytes, last modified: 12/4/2022 - 100% done

```
Saving Mycharger Bulletien_updated.csv to Mycharger Bulletien_updated.csv
```

```
!pip install pytorch-ignite
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting pytorch-ignite
  Downloading pytorch_ignite-0.4.10-py3-none-any.whl (264 kB)
     |████████████████████████████████| 264 kB 4.0 MB/s
Requirement already satisfied: packaging in /usr/local/lib/python3.8/dist-packages (fro
Requirement already satisfied: torch<2,>=1.3 in /usr/local/lib/python3.8/dist-packages 
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.8/dis
Installing collected packages: pytorch-ignite
Successfully installed pytorch-ignite-0.4.10
```

```
!pip install rouge
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting rouge
  Downloading rouge-1.0.1-py3-none-any.whl (13 kB)
Requirement already satisfied: six in /usr/local/lib/python3.8/dist-packages (from roug
Installing collected packages: rouge
Successfully installed rouge-1.0.1
```

Libraries Import

```
import pandas as pd
import numpy as np
import os
import re
import warnings
warnings.filterwarnings("ignore")
from sklearn import datasets, linear_model
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
```

```
import nltk
```

```
nltk.download('stopwords')
```

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

Data Reading

```
# read data from the csv file (from the location it is stored)
Data = pd.read_csv('Mycharger Bulletien_updated.csv',encoding='cp1252')
Data = Data.astype(str)
rows, columns = Data.shape
```

```
Data
```

	headline	text
0	Board of Governors announces University of New...	The University of New Haven Board of Governors...
1	Justine Bernard, 19, University of New Haven p...	University of New Haven psychology major Justi...
2	After a long year, graduation is here and in p...	On May 17, the University of New Haven will ho...
3	Labor rights demonstration forms outside of Un...	On Thursday, a group of roughly 25 former clea...
4	The university’s Justice, Equity, Diversity & ...	The student body asked for promotion of divers...
...	...	...
94	“Squid Game” season 2 confirmed	“Squid Game” has been Netflix’s hit original s...
95	Agatha’s big break is coming with “WandaVision...	On Oct. 7, Marvel revealed their plans to give...
96	“Scream” series returns	In December 1996, Ghostface was introduced as ...
97	3 true-crime documentaries to watch this spook	As the leaves change and the weather becomes c

Cleaning the dataset

```
df=Data[Data['text'].isnull()==False]
df=Data[Data['headline'].isnull()==False]
df=df[df['text']!='nan']

df.drop_duplicates(subset=['text'],inplace=True)  #dropping duplicates
```

```
df
```

	headline	text
0	Board of Governors announces University of New...	The University of New Haven Board of Governors...
1	Justine Bernard, 19, University of New Haven p...	University of New Haven psychology major Justi...
2	After a long year, graduation is here and in p...	On May 17, the University of New Haven will ho...
3	Labor rights demonstration forms outside of Un...	On Thursday, a group of roughly 25 former clea...
4	The university's Justice, Equity, Diversity & ...	The student body asked for promotion of divers...
...	...	...
94	"Squid Game" season 2 confirmed	"Squid Game" has been Netflix's hit original s...
95	Agatha's big break is coming with "WandaVision...	On Oct. 7, Marvel revealed their plans to give...
96	"Scream" series returns	In December 1996, Ghostface was introduced as ...
97	3 true-crime documentaries to watch this spook	As the leaves change and the weather becomes a

```
stop_words = set(stopwords.words('english'))

def text_cleaner(text,num):
  str = text.lower()
  str = BeautifulSoup(str, "lxml").text
  str = re.sub(r'\([^)]*\)', '', str)
  str = re.sub('"','', str)
  str = ' '.join([contraction_mapping[t] if t in contraction_mapping else t for t in str.spli
  str = re.sub(r"'s\b","",str)
  str = re.sub("[^a-zA-Z]", " ", str)
  str = re.sub('[m]{2,}', 'mm', str)
  if(num==0):
    str = re.sub(r'\.',' . ',str)
  if(num==0):
      tokens = [w for w in str.split() if not w in stop_words]

  else:
      tokens=str.split()
  long_words=[]
  for i in tokens:
      if len(i)>1:                                                 #removing short words
          long_words.append(i)
  return (" ".join(long_words)).strip()


contraction_mapping = {"ain't": "is not", "aren't": "are not","can't": "cannot", "'cause": "b


#call the function
```

```
clean_text = []
for t in df['text']:
    clean_text.append(text_cleaner(t,0))


#call the function
clean_summary = []
for t in df['headline']:
    clean_summary.append(text_cleaner(t,0))


df['text']=clean_text
df['headline']=clean_summary

df.replace('', np.nan, inplace=True)
df.dropna(axis=0,inplace=True)
```

Updating the df

```
df
```

	headline	text
0	board governors announces university new trans...	university new board governors unanimously app...
1	justine bernard university new psychology majo...	university new psychology major justine bernar...
2	long year graduation person	may university new host spring commencement ce...
3	labor rights demonstration forms outside unive...	thursday group roughly former cleaning service...
4	university justice equity diversity inclusion ...	student body asked promotion diversity inclusi...
...	...	...
94	squid game season confirmed	squid game netflix hit original show since cam...
95	agatha big break coming wandavision spinoff	oct marvel revealed plans give kathryn hahn wa...
96	scream series returns	december ghostface introduced serial killer at...
97	true crime documentaries watch spooky season	leaves change weather becomes colder everyone ...

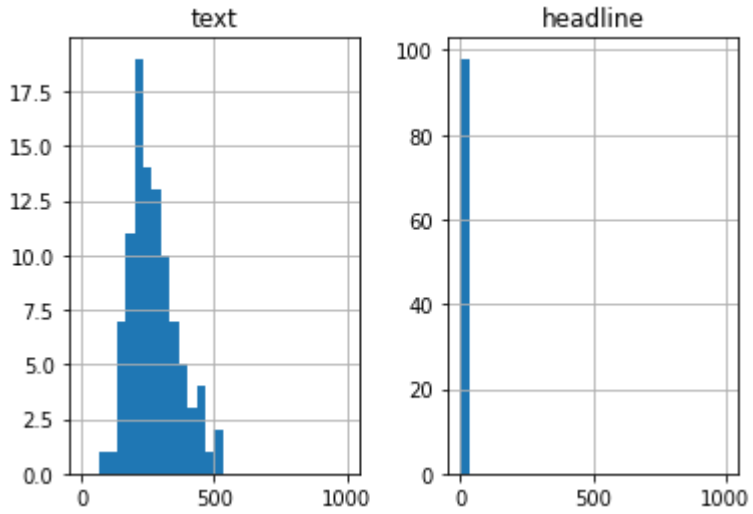
Analyzing the sequence distribution

```
import matplotlib.pyplot as plt
text_word_count = []
headline_word_count = []

# populate the lists with sentence lengths
for i in df['text']:
      temp=i.split()
      text_word_count.append(len(temp))

for j in df['headline']:
  #print(j)
  temp1=j.split()
  headline_word_count.append(len(temp1))

length_df = pd.DataFrame({'text':text_word_count, 'headline':headline_word_count})
length_df.hist(bins = 30,range=[0,1000])
plt.show()
```




```
# From the graph
# We can fix maximum length of text = 150 since most of the reviews have a length of 150 and

max_len_text= 500
max_len_headline=50


cnt=0
for i in df['text']:
    if(len(i.split())<=150):
        cnt=cnt+1
print(cnt/len(df['text']))
```

```
0.030612244897959183
```

Selecting text and headlines below the maximum lengths

```
text1 =np.array(df['text'])
headline1=np.array(df['headline'])

short_text=[]
short_summary=[]

for i in range(len(text1)):
    if(len(headline1[i].split())<=50 and len(text1[i].split())<=500):
        short_text.append(text1[i])
        short_summary.append(headline1[i])

df=pd.DataFrame({'text':short_text,'summary':short_summary})



# Validating the lengths

text1 =np.array(df['text'])
headline1=np.array(df['summary'])


for i in range(len(text1)):
    if(len(headline1[i].split())>=150):
      print(i)



df
```

	text	summary
0	university new board governors unanimously app...	board governors announces university new trans...
1	university new psychology major justine bernar...	justine bernard university new psychology majo...
2	may university new host spring commencement ce...	long year graduation person
3	thursday group roughly former cleaning service...	labor rights demonstration forms outside unive...
4	student body asked promotion diversity inclusi...	university justice equity diversity inclusion ...
...	...	...
91	squid game netflix hit original show since cam...	squid game season confirmed
92	oct marvel revealed plans give kathryn hahn wa...	agatha big break coming wandavision spinoff
93	december ghostface introduced serial killer at...	scream series returns
94	leaves change weather becomes colder everyone ...	true crime documentaries watch spooky season

```
print(df['text'][1],df['summary'][1],sep='\n')
```

```
university new psychology major justine bernard died morning june shooting downtown atla
justine bernard university new psychology major dies shot atlanta
```

Splitting data into train, test -- 70 - 30

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(df['text'],df['summary'],test_size=0.3,random_
```

```
print(len(x_train))
print(len(x_test))
```

```
67
29
```

Language Translation

```
#from torchtext.data import Field, BucketIterator
```

```
#pip install spacy
#!python -m spacy download en
```

```
from __future__ import unicode_literals, print_function, division
from io import open
import unicodedata
import string
import re
import random

import torch
import torch.nn as nn
from torch import optim
import torch.nn.functional as F

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Creating Vocabluray

```
SOS_token = 0
EOS_token = 1

class Lang:
```

```
        def __init__(self, name):
            self.name = name
            self.word2index = {}
            self.word2count = {}
            self.index2word = {0: "SOS", 1: "EOS"}
            self.n_words = 2  # Count SOS and EOS

        def addSentence(self, sentence):
            for word in sentence.split(' '):
                self.addWord(word)

        def addWord(self, word):
            if word not in self.word2index:
                self.word2index[word] = self.n_words
                self.word2count[word] = 1
                self.index2word[self.n_words] = word
                self.n_words += 1
            else:
                self.word2count[word] += 1


    def readLangs(text, summary, reverse=False):
        print("Reading lines...")

        # Split every line into pairs and normalize
        text=np.array(text)
        summary=np.array(summary)
        pairs = [[text[i],summary[i]] for i in range(len(text))]

        # Reverse pairs, make Lang instances
        if reverse:
            pairs = [list(reversed(p)) for p in pairs]
            input_lang = Lang(summary)
            output_lang = Lang(text)
        else:
            input_lang = Lang(text)
            output_lang = Lang(summary)

        return input_lang, output_lang, pairs
```

Sentence Normalizing

```
    def normalize_sentence(df, lang):
        sentence = df[lang].str.lower()
        sentence = sentence.str.replace('[^A-Za-z\s]+', '')
        sentence = sentence.str.normalize('NFD')
        sentence = sentence.str.encode('ascii', errors='ignore').str.decode('utf-8')
        return sentence
```

```
def read_sentence(df, lang1, lang2):
   sentence1 = normalize_sentence(df, lang1)
   sentence2 = normalize_sentence(df, lang2)
   return sentence1, sentence2

def read_file(loc, lang1, lang2):
   df = pd.read_csv(loc, delimiter='\t', header=None, names=[lang1, lang2])
   return df

def process_data(lang1,lang2):
   df = read_file('text/%s-%s.txt' % (lang1, lang2), lang1, lang2)
   print("Read %s sentence pairs" % len(df))
   sentence1, sentence2 = read_sentence(df, lang1, lang2)

   source = Lang()
   target = Lang()
   pairs = []
   for i in range(len(df)):
       if len(sentence1[i].split(' ')) < MAX_LENGTH and len(sentence2[i].split(' ')) < MAX_LE
           full = [sentence1[i], sentence2[i]]
           source.addSentence(sentence1[i])
           target.addSentence(sentence2[i])
           pairs.append(full)

   return source, target, pairs
```

Data Preparatiom

```
def prepareData(lang1, lang2, reverse=False):
    input_lang, output_lang, pairs = readLangs(lang1, lang2, reverse)
    print("Read %s sentence pairs" % len(pairs))
    print("Counting words...")
    for pair in pairs:
        input_lang.addSentence(pair[0])
        output_lang.addSentence(pair[1])
    print("Counted words:")
    print(input_lang.name,"--------------------", input_lang.n_words)
    #print(output_lang.name, output_lang.n_words)
    return input_lang, output_lang, pairs


input_lang, output_lang, pairs = prepareData(x_train, y_train , False)
```

```
      'university new psychology major justine bernard died morning june shooting downtow
      'evolution music industry picture swifties earning college credit new york universi
      'always enjoyed reading books long adapted binge worthy series films loved thrill c
      'batman dark gritty detective story different batman movie seen shows bruce wayne p
      'student body asked promotion diversity inclusion university responded creating jus
      'april twitter entered new era tesla ceo billionaire elon musk bought social media
      'friday morning paul pelosi husband house speaker nancy pelosi violently assaulted
      'halloween behind us may still looking something give little scare luckily theaters
```

```
 'american cancer society hosted breast cancer awareness walk lightpoint park past s
 'april student committee programming events hosted spring week reveal party announc
 'covid broadway theaters tentatively shut january amid new york phase reopening pla
 'ed sheeran taylor swift new collaboration joker queen track debut official singles
 'university new board governors unanimously approved official transition leadership
 'angela michael pullo three things mind opening business cats coffee community coup
 'following protests outside connecticut gov ned lamont residence decision release p
 'end year may creeping new movies still hitting theaters streaming platforms whethe
 'beckerman recreation center chargerrec staff leaving stone unturned comes implemen
 'oct west city council heard proposal founder branford based manufacturer wants bui
 'staying area thanksgiving break looking fun things got covered charger connection
 'years sanderson sisters finally coming back directed kenny ortega starring bette m
 'undergraduate student government association election results released via email m
 'simon property group group llc applied last evening site plan approval west planni
 'three years waited three years premiere season two euphoria one premiere anticipat
 'following elaborate multi faceted buildup harry styles officially announced releas
 'university fall production hatmaker wife underway theater department held audition
 'frank blackwell guilford conn went women march last year new york heard shooting p
 'although broadway closed theater doors pandemic halloween right around corner excu
 'dear members university community due increase number positive covid cases univers
 'recent years united states department agriculture introduced array techniques addr
 'transgender visibility week aims bring awareness transgender community experiences
 'former presidential candidate beto rourke visited newton oct talked gun violence l
 'looking new movies watch know start list new movie releases november want miss ete
 'december ghostface introduced serial killer attacked high school student sydney pr
 'leaves change weather becomes colder everyone enjoys cuddling watch movies time co
 'week iranian court issued first death sentence connection recent protests accordin
 'five teams student entrepreneurs gathered atrium university orange campus past thu
 'fx hulu comedy drama show reservation dogs year already making huge impact indigen
 'april twitter entered new era tesla ceo billionaire elon musk bought social media
 'college street music hall performance venue middle downtown new venue brings varie
 'past friday new solo powerhouse artist dakotaxela released two electric singles ev
 'university new fox connecticut fox affiliate announced today television station op
 'residential student traveling home holidays commuter student looking good movie wa
 'headlined marketing campaign placing actors stands right behind home plate major l
 'taylor swift fans weekend release swift album red also minute short film well fift
 'country music definitely faced growing pains controversy long history many fans sp
 'last semester university new brought variety food trucks campus become popular spo
 'tuesday connecticut voters filled number political positions friday voters chose d
 'updated include result statistics democrat nancy rossi declared winner west mayora

 'part international education week year panel discussion held modern languages offi
 'hollywood long way go comes accurate portrayals latinxs people color movies televi
 'march dozen recognized student organizations gathered celebrate beginning spring i
 'oct marvel revealed plans give kathryn hahn wandavision character spin series prem
 'summer walker second studio album still released nov project comprised songs featu
 'live nation recently announced massive pop punk music festival young originally sc
 'eternals recent movie release marvel cinematic universe franchise fans buzzing ant
 'although approaching end year still much look forward world entertainment followin
```

```
pairs[1]
```

```
['countless famous artists performed super bowl halftime show years year event featured
legendary performers dr dre mary blige snoop dogg eminem kendrick lamar five performers
```

```
      nearly grammy awards among golden gramophones collection snoop dogg grammy nominations
      california born rapper become household name last three decades thanks songs like gin
      juice drop like hot special distinction comes grammy awards snoop received grammy award
      nominations work years never grammy received first nomination best rap performance
      nuthin thang recent nomination featured artist kendrick lamar pimp butterfly released
      eminem grammy awards eminem game quite long fellow super bowl performers detroit born
      rapper holds grammy awards name date lose rapper earned grammy awards since turn
      century eminem first two grammy awards best rap solo performance best rap album nd
      grammy awards kendrick lamar grammy awards kendrick lamar american rapper received
      awards nominations including grammys six billboard music awards ascap vanguard award
      work composer lamar received first grammy nominations including best new artist album
      year good kid city received first grammy nominations best rap performance best rap song
      mary blinge grammy awards mary blinge shot fame debut album yielded top hit real love
      signature hits include going mary jane love grammy awards mary received nine awards
      thirty one nominations first grammy best rap performance duo group collaboration method
      man need get received first solo grammy best female vocal performance think know album
      drama dr dre grammy wins dr dre well known rapper producer since days member seminal
      hip hop group nwa first grammy best rap solo let ride rd grammy awards dr dre made
      grammy stage debut performing medley love way lie need doctor eminem skylar grey adam
      levine rihanna slew grammys name total seven wins whopping nominations dozens grammy
      award winners performed super bowl halftime show years winning grammys others honors
      means measure musical success mary blige dr dre nine seven grammy awards respectively',
       'super bowl halftime show performer grammy awards']


#---- Deep Model
```

Converting word to Index then index to word

```
SOS_token = 0
EOS_token = 1


def indexesFromSentence(lang, sentence):
    return [lang.word2index[word] for word in sentence.split(' ')]

def tensorFromSentence(lang, sentence):
    indexes = indexesFromSentence(lang, sentence)
    indexes.append(EOS_token)
    return torch.tensor(indexes, dtype=torch.long, device=device).view(-1, 1)

def tensorsFromPair(input_lang,output_lang,pair):
    input_tensor = tensorFromSentence(input_lang, pair[0])
    target_tensor = tensorFromSentence(output_lang, pair[1])
    return (input_tensor, target_tensor)
```

```
from __future__ import unicode_literals, print_function, division
from io import open
import unicodedata
```

```
import string
import re
import random

import torch
import torch.nn as nn
from torch import optim
import torch.nn.functional as F

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Encoder Model and Decoder Model

```
class Encoder(nn.Module):
   def __init__(self, input_dim, hidden_dim, embbed_dim, num_layers):
       super(Encoder, self).__init__()

       #set the encoder input dimesion , embbed dimesion, hidden dimesion, and number of laye
       self.input_dim = input_dim
       self.embbed_dim = embbed_dim
       self.hidden_dim = hidden_dim
       self.num_layers = num_layers

       #initialize the embedding layer with input and embbed dimention
       self.embedding = nn.Embedding(input_dim, self.embbed_dim)
      #change

       #intialize the GRU to take the input dimetion of embbed, and output dimention of hidde
       #set the number of gru layers
       self.gru = nn.GRU(self.embbed_dim, self.hidden_dim, num_layers=self.num_layers)
   def forward(self, src):

       embedded = self.embedding(src).view(1,1,-1)
       outputs, hidden = self.gru(embedded)
       return outputs, hidden

class Decoder(nn.Module):
   def __init__(self, output_dim, hidden_dim, embbed_dim, num_layers):
       super(Decoder, self).__init__()

#set the encoder output dimension, embed dimension, hidden dimension, and number of layers
       self.embbed_dim = embbed_dim
       self.hidden_dim = hidden_dim
       self.output_dim = output_dim
       self.num_layers = num_layers

# initialize every layer with the appropriate dimension. For the decoder layer, it will consi
       self.embedding = nn.Embedding(output_dim, self.embbed_dim)
       self.gru = nn.GRU(self.embbed_dim, self.hidden_dim, num_layers=self.num_layers)
```

```
        self.out = nn.Linear(self.hidden_dim, output_dim)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):

# reshape the input to (1, batch_size)
        input = input.view(1, -1)
        embedded = F.relu(self.embedding(input))
        output, hidden = self.gru(embedded, hidden)
        prediction = self.softmax(self.out(output[0]))

        return prediction, hidden



MAX_LENGTH = 300
```

Define Model

```
class Seq2Seq(nn.Module):
    def __init__(self, encoder, decoder, device, MAX_LENGTH=MAX_LENGTH):
        super().__init__()

#initialize the encoder and decoder
        self.encoder = encoder
        self.decoder = decoder
        self.device = device

    def forward(self, source, target, teacher_forcing_ratio=0.5):

        input_length = source.size(0) #get the input length (number of words in sentence)
        batch_size = target.shape[1]
        target_length = target.shape[0]
        vocab_size = self.decoder.output_dim

#initialize a variable to hold the predicted outputs
        outputs = torch.zeros(target_length, batch_size, vocab_size).to(self.device)

#encode every word in a sentence
        for i in range(input_length):
            encoder_output, encoder_hidden = self.encoder(source[i])

#use the encoder’s hidden layer as the decoder hidden
        decoder_hidden = encoder_hidden.to(device)

#add a token before the first predicted word
        decoder_input = torch.tensor([SOS_token], device=device)  # SOS

#topk is used to get the top K value over a list
```

```
#predict the output word from the current target word. If we enable the teaching force,  then

        for t in range(target_length):
            decoder_output, decoder_hidden = self.decoder(decoder_input, decoder_hidden)
            outputs[t] = decoder_output
            teacher_force = random.random() < teacher_forcing_ratio
            topv, topi = decoder_output.topk(1)
            input = (target[t] if teacher_force else topi)
            if(teacher_force == False and input.item() == EOS_token):
                break

        return outputs
```

Function to train Model

```
teacher_forcing_ratio = 0.5

def clacModel(model, input_tensor, target_tensor, model_optimizer, criterion):
    model_optimizer.zero_grad()

    input_length = input_tensor.size(0)
    loss = 0
    epoch_loss = 0
    # print(input_tensor.shape)

    output = model(input_tensor, target_tensor)

    num_iter = output.size(0)
    print(num_iter)

#calculate the loss from a predicted sentence with the expected result
    for ot in range(num_iter):
        loss += criterion(output[ot], target_tensor[ot])

    loss.backward()
    model_optimizer.step()
    epoch_loss = loss.item() / num_iter

    return epoch_loss

def trainModel(model, source, target, pairs, num_iteration=20000):
    model.train()

    optimizer = optim.SGD(model.parameters(), lr=0.01)
    criterion = nn.NLLLoss()
    total_loss_iterations = 0

    training_pairs = [tensorsFromPair(source, target, random.choice(pairs))
                      for i in range(num_iteration)]
```

```
    for iter in range(1, num_iteration+1):
        training_pair = training_pairs[iter - 1]
        input_tensor = training_pair[0]
        target_tensor = training_pair[1]

        loss = clacModel(model, input_tensor, target_tensor, optimizer, criterion)

        total_loss_iterations += loss

        if iter % 5000 == 0:
            avarage_loss= total_loss_iterations / 5000
            total_loss_iterations = 0
            print('%d %.4f' % (iter, avarage_loss))

    torch.save(model.state_dict(), 'mytraining.pt')
    return model
```

Evaluation matrix

```
MAX_LENGTH=200


def evaluate(model, input_lang, output_lang, sentences, max_length=MAX_LENGTH):
    with torch.no_grad():
        input_tensor = tensorFromSentence(input_lang, sentences[0])
        output_tensor = tensorFromSentence(output_lang, sentences[1])

        decoded_words = []

        output = model(input_tensor, output_tensor)
        # print(output_tensor)

        for ot in range(output.size(0)):
            topv, topi = output[ot].topk(1)
            # print(topi)

            if topi[0].item() == EOS_token:
                decoded_words.append('<EOS>')
                break
            else:
                decoded_words.append(output_lang.index2word[topi[0].item()])
    return decoded_words
def evaluateRandomly(model, source, target, pairs, n=10):
    for i in range(n):
        output_sentence=""
        pair = random.choice(pairs)
        text = pair[0]
        summary = pair[1]
```

```
        output_words = evaluate(model, source, target, pair)
        output_sentence = ' '.join(output_words)
        print("Summary is:  ", pair[1])
        print("Predicted Summary is:",output_sentence)
        score = calculate_rogue(pair[1], output_sentence)
        print(score)


for i in range(2):
        pair = random.choice(pairs)
        text = pair[1]
        print(text)
```

```
    russia begins illegal referendums ukraine annex disputed regions
    euphoria take anymore
```

Training model with epochs

```
embed_size = 256
hidden_size = 512
num_layers = 7
num_iteration = 50
output_size = output_lang.n_words
#create encoder-decoder model
encoder = Encoder(input_lang.n_words, hidden_size, embed_size, num_layers)
decoder = Decoder(output_size, hidden_size, embed_size, num_layers)

model = Seq2Seq(encoder, decoder, device).to(device)
#print model
print(encoder)
print(decoder)

model = trainModel(model, input_lang, output_lang, pairs, num_iteration)
evaluateRandomly(model, input_lang, output_lang, pairs,  n=10)
```

```
    5
    5
    5
    5
    7
    9
    9
    8
    6
    9
    6
    9
    7
    7
    6
    9
    7
```

```
8
10
6
4
5
9
2
2
5
Summary is:   batman detective story waiting
Predicted Summary is: <EOS>
0.0
Summary is:   sheeran swift joker queen set royal debut
Predicted Summary is: <EOS>
0.0
Summary is:   batman detective story waiting
Predicted Summary is: <EOS>
0.0
Summary is:   reviewing newest horror movie smile
Predicted Summary is: <EOS>
0.0
Summary is:   highlighting native american media reservation dogs fx
Predicted Summary is: <EOS>
0.0
Summary is:   disability entertainment
Predicted Summary is: <EOS>
0.0
Summary is:   wheeler walker jr gaudy awful inappropriate best singer songwriter nas
Predicted Summary is: <EOS>
0.0
Summary is:   swift releases well short film
Predicted Summary is: <EOS>
0.0
Summary is:   meet usga election winners
Predicted Summary is: <EOS>
0.0
Summary is:   marvel eternals fan favorite mcu
Predicted Summary is: <EOS>
0.0
```

```
attn_plot_threshold = 0.45

def evaluateRandomlyprint_1(model, input_lang, output_lang,pairs, n=5):
    text=list()
    headline=list()
    pred_headline=list()

    for i in range(n):
        pair = random.choice(pairs)
        tokenized_input = nltk.word_tokenize(pair[0])
        output_words = evaluate(model, input_lang, output_lang, pair)
        output_sentence = ' '.join(output_words)
```

```
        score = calculate_rogue(pair[1], output_sentence)
        print(score)
```

Evaluation Matrix

```
import nltk
nltk.download('punkt')
from rouge import Rouge
```

```
    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Package punkt is already up-to-date!
```

```
def calculate_rogue(src_trg, pred_trg):

  #cut off <eos> token
  pred_trg = pred_trg[:-6]


  if (len(pred_trg) == 0):
    rogue_score = 0.0
  else:
    s = rouge.get_scores(pred_trg, src_trg, avg= True)
    rogue_score = s['rouge-1']['f']

  return rogue_score


rouge = Rouge()
#model, input_lang, output_lang, pairs,  n=10
#evaluateRandomly(model, input_lang, output_lang, pairs,  n=10)
evaluateRandomlyprint_1(model, input_lang, output_lang,pairs,n=1)
```

```
    0.0
```

Defining Model

```
model
```

```
    Seq2Seq(
      (encoder): Encoder(
        (embedding): Embedding(5487, 256)
        (gru): GRU(256, 512, num_layers=7)
      )
      (decoder): Decoder(
```

```
          (embedding): Embedding(340, 256)
          (gru): GRU(256, 512, num_layers=7)
          (out): Linear(in_features=512, out_features=340, bias=True)
          (softmax): LogSoftmax(dim=1)
        )
      )
```

Freezing model for Transfer learning

```
for param in model.parameters():
    param.requires_grad = False
```

Mini Network

```
class Encodermini(nn.Module):
   def __init__(self, input_dim, hidden_dim, embbed_dim, num_layers):
       super(Encodermini, self).__init__()

       #set the encoder input dimesion , embbed dimesion, hidden dimesion, and number of laye
       self.input_dim = input_dim
       self.embbed_dim = embbed_dim
       self.hidden_dim = hidden_dim
       self.num_layers = num_layers

       #initialize the embedding layer with input and embbed dimention
       self.embedding = nn.Embedding(input_dim, self.embbed_dim)
      #change

       #intialize the GRU to take the input dimetion of embbed, and output dimention of hidde
       #set the number of gru layers
       self.gru = nn.GRU(self.embbed_dim, self.hidden_dim, num_layers=self.num_layers)
   def forward(self, src):

       embedded = self.embedding(src).view(1,1,-1)
       outputs, hidden = self.gru(embedded)
       return outputs, hidden

class Decodermini(nn.Module):
   def __init__(self, output_dim, hidden_dim, embbed_dim, num_layers):
       super(Decodermini, self).__init__()

#set the encoder output dimension, embed dimension, hidden dimension, and number of layers
       self.embbed_dim = embbed_dim
       self.hidden_dim = hidden_dim
       self.output_dim = output_dim
       self.num_layers = num_layers

# initialize every layer with the appropriate dimension. For the decoder layer, it will consi
```

```
        self.embedding = nn.Embedding(output_dim, self.embbed_dim)
        self.gru = nn.GRU(self.embbed_dim, self.hidden_dim, num_layers=self.num_layers)
        self.out = nn.Linear(self.hidden_dim, output_dim)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):

# reshape the input to (1, batch_size)
        input = input.view(1, -1)
        embedded = F.relu(self.embedding(input))
        output, hidden = self.gru(embedded, hidden)
        prediction = self.softmax(self.out(output[0]))

        return prediction, hidden

class Seq2Seqmin(nn.Module):
    def __init__(self, encoder, decoder, device, MAX_LENGTH=MAX_LENGTH):
        super().__init__()

#initialize the encoder and decoder
        self.encoder = encoder
        self.decoder = decoder
        self.device = device

    def forward(self, source, target, teacher_forcing_ratio=0.5):

        input_length = source.size(0) #get the input length (number of words in sentence)
        batch_size = target.shape[1]
        target_length = target.shape[0]
        vocab_size = self.decoder.output_dim

#initialize a variable to hold the predicted outputs
        outputs = torch.zeros(target_length, batch_size, vocab_size).to(self.device)

#encode every word in a sentence
        for i in range(input_length):
            encoder_output, encoder_hidden = self.encoder(source[i])

#use the encoder’s hidden layer as the decoder hidden
        decoder_hidden = encoder_hidden.to(device)

#add a token before the first predicted word
        decoder_input = torch.tensor([SOS_token], device=device)  # SOS

#topk is used to get the top K value over a list
#predict the output word from the current target word. If we enable the teaching force,  then

        for t in range(target_length):
            decoder_output, decoder_hidden = self.decoder(decoder_input, decoder_hidden)
            outputs[t] = decoder_output
            teacher_force = random.random() < teacher_forcing_ratio
```

```
            topv, topi = decoder_output.topk(1)
            input = (target[t] if teacher_force else topi)
            if(teacher_force == False and input.item() == EOS_token):
                break

        return outputs


teacher_forcing_ratio = 0.5

def clacModel(model, input_tensor, target_tensor, model_optimizer, criterion):
    model_optimizer.zero_grad()

    input_length = input_tensor.size(0)
    loss = 0
    epoch_loss = 0
    # print(input_tensor.shape)

    output = model(input_tensor, target_tensor)

    num_iter = output.size(0)
    print(num_iter)

#calculate the loss from a predicted sentence with the expected result
    for ot in range(num_iter):
        loss += criterion(output[ot], target_tensor[ot])

    loss.backward()
    model_optimizer.step()
    epoch_loss = loss.item() / num_iter

    return epoch_loss

def trainModel(model, source, target, pairs, num_iteration=20000):
    model.train()

    optimizer = optim.SGD(model.parameters(), lr=0.01)
    criterion = nn.NLLLoss()
    total_loss_iterations = 0

    training_pairs = [tensorsFromPair(source, target, random.choice(pairs))
                      for i in range(num_iteration)]

    for iter in range(1, num_iteration+1):
        training_pair = training_pairs[iter - 1]
        input_tensor = training_pair[0]
        target_tensor = training_pair[1]

        loss = clacModel(model, input_tensor, target_tensor, optimizer, criterion)

        total_loss_iterations += loss
```

```
        if iter % 5000 == 0:
            avarage_loss= total_loss_iterations / 5000
            total_loss_iterations = 0
           #  print('%d %.4f' % (iter, avarage_loss))

    torch.save(model.state_dict(), 'mytraining.pt')
    return model


MAX_LENGTH=200


embed_size = 256
hidden_size = 512
num_layers = 4
num_iteration = 100
output_size = output_lang.n_words
#create encoder-decoder model
encoder = Encodermini(input_lang.n_words, hidden_size, embed_size, num_layers)
decoder = Decodermini(output_size, hidden_size, embed_size, num_layers)

model = Seq2Seqmin(encoder, decoder, device).to(device)
#print model
print(encoder)
print(decoder)

model = trainModel(model, input_lang, output_lang, pairs, num_iteration)
evaluateRandomly(model, input_lang, output_lang, pairs,  n=1)
```

```
10
6
9
5
6
10
9
7
6
10
7
8
9
9
8
6
7
6
4
9
8
7
11
9
```

```
9
3
7
6
6
9
8
7
9
8
8
7
5
5
9
8
7
8
9
10
7
10
6
11
5
8
9
8
6
2
12
Summary is:   justine bernard university new psychology major dies shot atlanta
Predicted Summary is: <EOS>
0.0
```

Transfer Learning

```
!pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/pub
Collecting transformers
  Downloading transformers-4.25.1-py3-none-any.whl (5.8 MB)
     |████████████████████████████████| 5.8 MB 4.3 MB/s
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.8/dist-packages (fr
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.8/dist-packages
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.8/dist-packag
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-packages (from
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
```

```
  Downloading tokenizers-0.13.2-cp38-cp38-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
     |████████████████████████████████| 7.6 MB 34.4 MB/s
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.8/dist-packages (fro
Requirement already satisfied: filelock in /usr/local/lib/python3.8/dist-packages (from
Collecting huggingface-hub<1.0,>=0.10.0
  Downloading huggingface_hub-0.11.1-py3-none-any.whl (182 kB)
     |████████████████████████████████| 182 kB 61.3 MB/s
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.8/dist-packages (fr
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.8/di
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/python3.8/dist
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.8/dist-packag
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dist-packages (f
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local/lib
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.11.1 tokenizers-0.13.2 transformers-4.25.1
```

```
from transformers import pipeline
import os


os.environ["CUDA_VISIBLE_DEVICES"] = "0"



summarizer = pipeline("summarization")
```

```
No model was supplied, defaulted to sshleifer/distilbart-cnn-12-6 and revision a4f8f3e (
Using a pipeline without specifying a model name and revision in production is not recom
```

```
Downloading: 100%  1.80k/1.80k [00:00<00:00, 9.90kB/s]
Downloading: 100%  1.22G/1.22G [00:53<00:00, 15.7MB/s]
Downloading: 100%  26.0/26.0 [00:00<00:00, 722B/s]
Downloading: 100%  899k/899k [00:01<00:00, 1.25MB/s]
Downloading: 100%  456k/456k [00:00<00:00, 628kB/s]
```

```
summarizer = pipeline("summarization", model="t5-base", tokenizer="t5-base", framework="tf")
```

```
Downloading: 100%                                    1.20k/1.20k [00:00<00:00, 24.1kB/s]
```

```
summary_text = summarizer(pair[0], max_length=100, min_length=5, do_sample=False)[0]['summary
print(summary_text)
```

```
 season two of euphoria one premiere premiered on tuesday . fans waited three years to wa
```

```
Downloading: 100%                                    1.39M/1.39M [00:01<00:00, 1.20MB/s]
```

Colab paid products - Cancel contracts here

✓ 1m 3s completed at 11:22 PM