Open in Colab

Overview

Here we'll try to predict the Short email responses.
As most of the emails have shorter reply.
So that it can be easy to reply, without the hassle of typing.
It can help in saving the time of the person.

```
#######################################################
```

Will update later.

```
#######################################################
```

[my kilnkn]

```
import tensorflow as tf
tf.__version__
```

```
'1.15.0'
```

```
import tensorflow as tf
tf.keras.__version__
```

```
The default version of TensorFlow in Colab will soon switch to TensorFlow 2.x.
We recommend you upgrade now or ensure your notebook will continue to use TensorFlow 1.x via the %tens
'2.2.4-tf'
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import pickle
import tensorflow as tf
import os
from tensorflow.python.keras.layers import Layer
from tensorflow.python.keras import backend as K
```

```
# !pip3 install keras-self-attention
```

Attention Layer

```
class BahdanauAttention(tf.keras.layers.Layer):
```

```
  def __init__(self, units):
    super(BahdanauAttention, self).__init__()
    self.W1 = tf.keras.layers.Dense(units)
    self.W2 = tf.keras.layers.Dense(units)
    self.V = tf.keras.layers.Dense(1)

  def call(self,input, query, values):
    # hidden shape == (batch_size, hidden size)
    # hidden_with_time_axis shape == (batch_size, 1, hidden size)
    # we are doing this to perform addition to calculate the score
    input = None
    hidden_with_time_axis = tf.expand_dims(query, 1)

    # score shape == (batch_size, max_length, 1)
    # we get 1 at the last axis because we are applying score to self.V
    # the shape of the tensor before applying self.V is (batch_size, max_length, units)
    score = self.V(tf.nn.tanh(
        self.W1(values) + self.W2(hidden_with_time_axis)))

    # attention_weights shape == (batch_size, max_length, 1)
    attention_weights = tf.nn.softmax(score, axis=1)

    # context_vector shape after sum == (batch_size, hidden_size)
    context_vector = attention_weights * values
    context_vector = tf.reduce_sum(context_vector, axis=1)

    return context_vector, attention_weightst
```

Reading the data

```
# Load the Drive helper and mount
from google.colab import drive
# This will prompt for authorization.
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318

Enter your authorization code:
··********
Mounted at /content/drive
```

```
import json
arr = []
question = []
answer = []
with open("drive/My Drive/JSON_data/train_data.txt","rb") as json_file:
    data = json.load(json_file)
    for data_ in data['data']:
```

```
    for j,para in enumerate(data_['paragraphs']):
      for k,qas in enumerate(para['qas']):
        for ans in qas['answers']:
          answer.append(ans['text'])
          question.append(qas['question'])
          break
```

```
df = pd.DataFrame(list(zip(question, answer)),
               columns =['question', 'reply'])
```

```
df.tail(3)
```

	question	reply
86818	With what Belorussian city does Kathmandu have...	Minsk
86819	In what year did Kathmandu create its initial ...	1975
86820	What is KMC an initialism of?	Kathmandu Metropolitan City

```
#Add more data to the above data
df2 = pd.read_csv("drive/My Drive/JSON_data/qa_dataset.csv", encoding = "ISO-8859-1", low_mem
```

```
df2.head(2)
```

	ArticleTitle	Question	Answer	DifficultyFromQuestioner	D
0	Alessandro_Volta	Was Volta an Italian physicist?	yes	easy	
1	Alessandro_Volta	Is Volta buried in the city of Pittsburgh?	no	easy	

```
#Drop the columns which we don' need
df2 = df2.drop(['ArticleTitle','DifficultyFromQuestioner','DifficultyFromAnswerer','ArticleFi
df2.columns = ['question', 'reply']
```

```
df3 = pd.read_csv("drive/My Drive/JSON_data/music_questions.csv")
df3.columns = ['q','question', 'reply']
df3.drop(['q'],axis = 1).head(2)
```

	question	reply
0	how long is this cord? the pictures looks like...	I took a photo: <http://imgur.com/G48f1C4>I bo...
1	Has anyone used this to split a stereo signal?...	I believe this adapter yields a mono split and...

```
df4 = pd.read_csv("drive/My Drive/JSON_data/grocery_questions.csv")
df4.columns = ['q','question', 'reply']
```

```
df4.drop(['q'],axis = 1).head(2)
```

	question	reply
0	what are the colors that come in the package?	All colors seen on box plus Teal, Burgundy, Bl...
1	difference between meat cure and pickling salt	Pickling salt is a very pure form of salt. A m...

```
df5 = pd.read_csv("drive/My Drive/JSON_data/video_game_qa.csv")
df5.columns = ['q','question', 'reply']
df5.drop(['q'],axis = 1).head(2)
```

	question	reply
0	Yes, you will need to go to their website to d...	Yes, you will need to go to their website to d...
1	As long as it has a USB port it should work fi...	As long as it has a USB port it should work fi...

```
df2.shape
```

```
(2917, 2)
```

```
df.shape
```

```
(86821, 2)
```

```
df3.shape
```

```
(2976, 3)
```

```
df4.shape
```

```
(2997, 3)
```

```
df5.shape
```

```
(1183, 3)
```

```
#Append both the dataset

frames = [df, df2,df3,df4,df5]
data = pd.concat(frames)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: FutureWarning: Sorting b
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  This is separate from the ipykernel package so we can avoid doing imports until
```

```
# data = pd.read_csv('qa_dataset.csv', encoding = "ISO-8859-1", low_memory=False)
```

```
# data.head(2)
```

	ArticleTitle	Question	Answer	DifficultyFromQuestioner	D
0	Alessandro_Volta	Was Volta an Italian physicist?	yes	easy	
1	Alessandro_Volta	Is Volta buried in the city of Pittsburgh?	no	easy	

```
# data = data.drop(['ArticleTitle','DifficultyFromQuestioner','DifficultyFromAnswerer','Artic
```

```
# data.columns = ['question','reply']
```

```
# Columns
# Description of the data frame
print("="*25 + " Data Overview " + "="*25)
print("\n", data.info(), "\n")
print("="*25 + " Sample Data " + "="*25)
print(data.head(5))
```

```
========================= Data Overview =========================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2917 entries, 0 to 2916
Data columns (total 2 columns):
question    2917 non-null object
reply       2917 non-null object
dtypes: object(2)
memory usage: 45.7+ KB

 None

========================= Sample Data =========================
                                          question           reply
0                   Was Volta an Italian physicist?             yes
1         Is Volta buried in the city of Pittsburgh?              no
2  Did Volta have a passion for the study of elec...             yes
3  What is the battery made by Volta credited to be?  the first cell
4  What important electrical unit was named in ho...        the volt
```

Preprocessing

```
# Convert data into lower case
data = data.apply(lambda x: x.astype(str).str.lower())
data.head(4)
```

	question	reply
0	was volta an italian physicist?	yes
1	is volta buried in the city of pittsburgh?	no
2	did volta have a passion for the study of elec...	yes
3	what is the battery made by volta credited to be?	the first cell

```
data.head()
```

	question	reply
0	was volta an italian physicist?	yes
1	is volta buried in the city of pittsburgh?	no
2	did volta have a passion for the study of elec...	yes
3	what is the battery made by volta credited to be?	the first cell
4	what important electrical unit was named in ho...	the volt

```
#counting length of each sentence in target
sentences = data['reply'].values
len_arr = []
for sent in sentences:
  count = 0
  for words in sent:
    count += 1
  len_arr.append(count)
```

```
data['reply_length'] = len_arr
```

```
#Calculating average length of th target sentences
avg_length = sum(len_arr)/len(len_arr)
print('average length',avg_length)
```

```
average length 25.082961947206034
```

```
data.tail(2)
```

	question	reply	r
2915	was wilson , a staunch opponent of antisemitis...	yes	
2916	what happened in 1917?	raised billions through liberty loans, imposed...	

```
# Remove the full stops from the dataframe reply
bad_chars = [';', ':', '!', "*",'.',')','(','?']
preprocessed_reply = []

for reply in data['reply'].values:
  for i in bad_chars :
    reply = reply.replace(i, '')
  preprocessed_reply.append(reply)

data['reply'] = preprocessed_reply
data['reply'] = 'start_ ' + data['reply'] + ' _end'


# trim the string the dataframe
data = data.applymap(lambda x: x.strip() if isinstance(x, str) else x)


data.head(2)
```

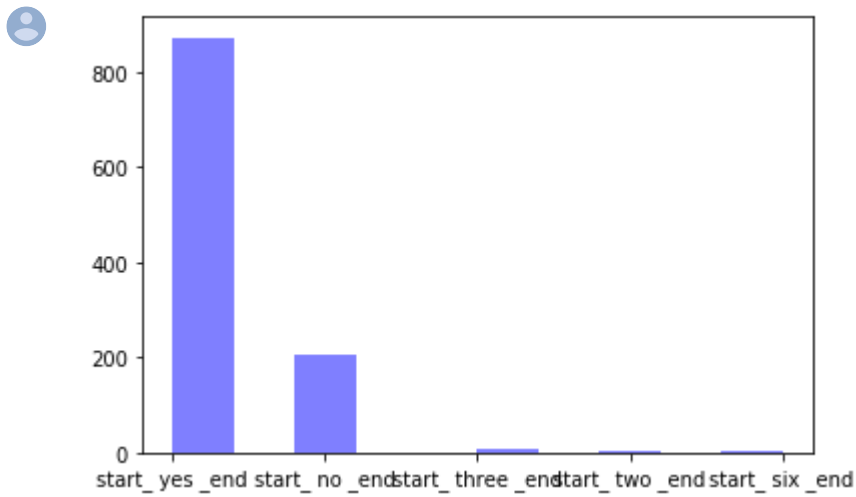
	question	reply	reply_length
0	was volta an italian physicist?	start_ yes _end	3
1	is volta buried in the city of pittsburgh?	start_ no _end	2

```
reply_list = list(data['reply'].values)
reply_dict = {i:reply_list.count(i) for i in reply_list}


from collections import OrderedDict
reply_dict_sorted = OrderedDict(sorted(reply_dict.items(), key=lambda x: x[1]))


reply = []
keys = []
for item in reply_dict_sorted.items():
  reply.append(item[0])
  keys.append(item[1])


reply.reverse()
keys.reverse()


#top 10 reply
top_5_reply = reply[:5]
top_5_keys = keys[:5]
```

```
temp_reply_list = []
i = 0
for key in top_5_keys:
  for _ in range(0,key):
    temp_reply_list.append(top_5_reply[i])
  i += 1
```

Article title can be used while giving the sentence for tokenizing later question do we need to preprocess the data

```
plt.hist(temp_reply_list, 10,
         histtype='bar',
         facecolor='b',
         alpha=0.5)
plt.show()
```



From the above graph we can observe that **yes** and **no** reply are the most common reply in the mails later will force model to predict mostly yes

To overcome above problem we need to remove some of the data which is having yes and no as resp

```
#drop the yes/no response from
temp_df_wo_yes = data[data['reply'] != 'start_ yes _end']
temp_df_wo_no_yes = temp_df_wo_yes[temp_df_wo_yes['reply'] != 'start_ no _end']
```

```
temp_df_w_yes = data[data['reply'] == 'start_ yes _end']
temp_df_w_no = data[data['reply'] == 'start_ no _end']
```

```
#calcuating data with yes and no replies
print('yes replies length',len(temp_df_w_yes))
print('no replies length',len(temp_df_w_no))
```

```
yes replies length 872
no replies length 206
```

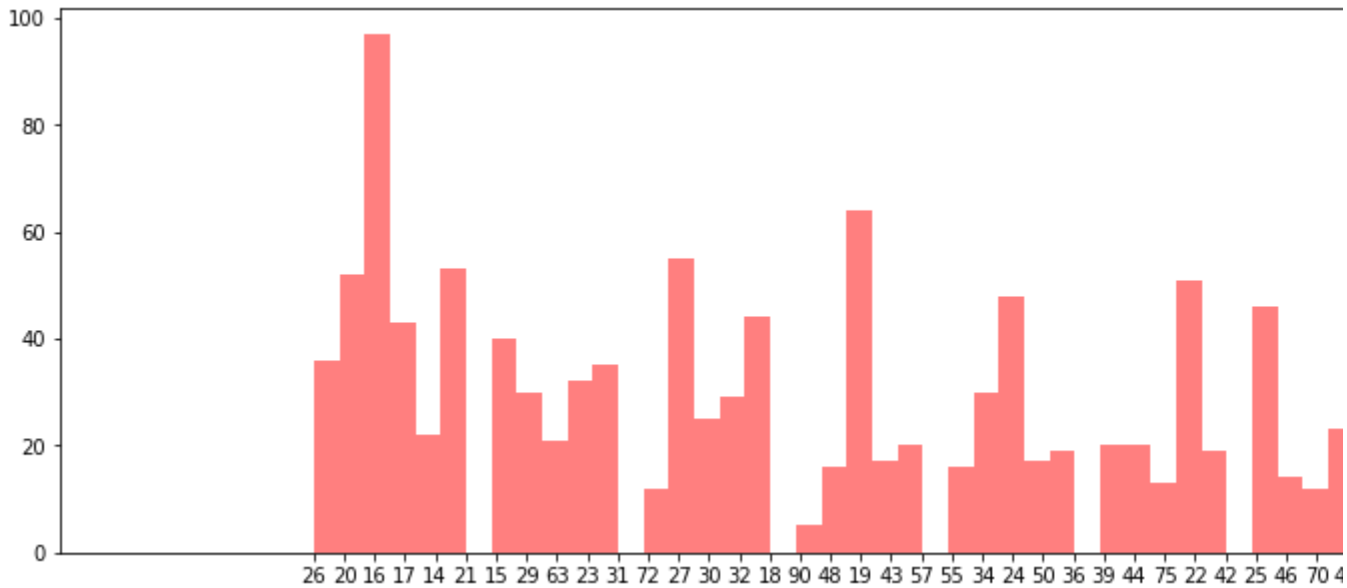
```
#We'll taek only 100 responses with yes and 100 with no
yes_df = temp_df_w_yes[:100]
no_df = temp_df_w_no[:100]

#Appending 100 yes and 100 no replies data
temp_df_wo_no_yes.append(yes_df, ignore_index=True)
temp_df_wo_no_yes.append(no_df, ignore_index=True)
data = temp_df_wo_no_yes

data.head()
```

	question	reply	reply_length
3	what is the battery made by volta credited to be?	start_ the first cell _end	14
4	what important electrical unit was named in ho...	start_ the volt _end	8
5	what important electrical unit was named in ho...	start_ volt _end	4
6	where did volta enter retirement?	start_ spain _end	5
10	for how many years did alessandro volta live?	start_ 53 _end	2

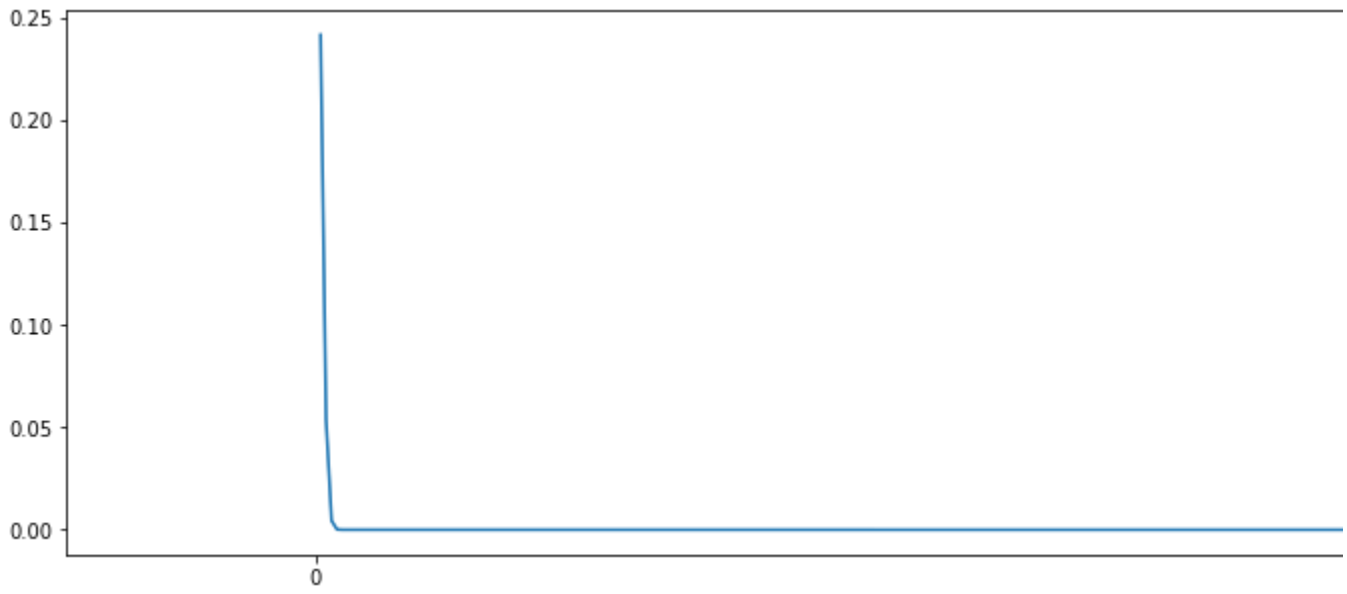
```
import matplotlib
matplotlib.rc('figure', figsize=[50,5])
#Getting the length of each reply
data["reply Length"]= data["reply"].str.len()
len_reply = data["reply Length"].values

#converting datatype to string
len_str_arr = []
for num in len_reply:
  len_str_arr.append(str(num))

no_reply_counter = 0
for num in len_reply:
  if num == 0:
    no_reply_counter += 1

temp_arr = []
for indx in len_reply:
  if indx == 0:
    continue
  else:
    temp_arr.append(indx)

print("number of email don' have any reply", end = ' : ')
```

```
print(no_reply_counter)
print('*' * 50)

print('Max length', end = ' : ')
print(max(list(len_reply)))
print('*' * 50)

print('Min Length', end = ' : ')
print(min(list(temp_arr)))
print('*' * 50)

print("Average Length", end = ' : ')
print((sum(list(len_reply)))/(len(list(len_reply))))
print('*' * 50)

print("No of unique reply", end = ' : ')
print(len(list(set(len_reply))))
print('*' * 50)


print('-' * 50)
print("Distribution of the words")
print('-' * 50)

plt.hist(len_str_arr, 200,
         histtype='bar',
         facecolor='r',
         alpha=0.5)
plt.show()
```

```
number of email don' have any reply : 0
**************************************************
Max length : 932
**************************************************
Min Length : 12
**************************************************
Average Length : 49.566612289287654
**************************************************
No of unique reply : 168
**************************************************
--------------------------------------------------
Distribution of the words
--------------------------------------------------
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_g
  after removing the cwd from sys.path.
```

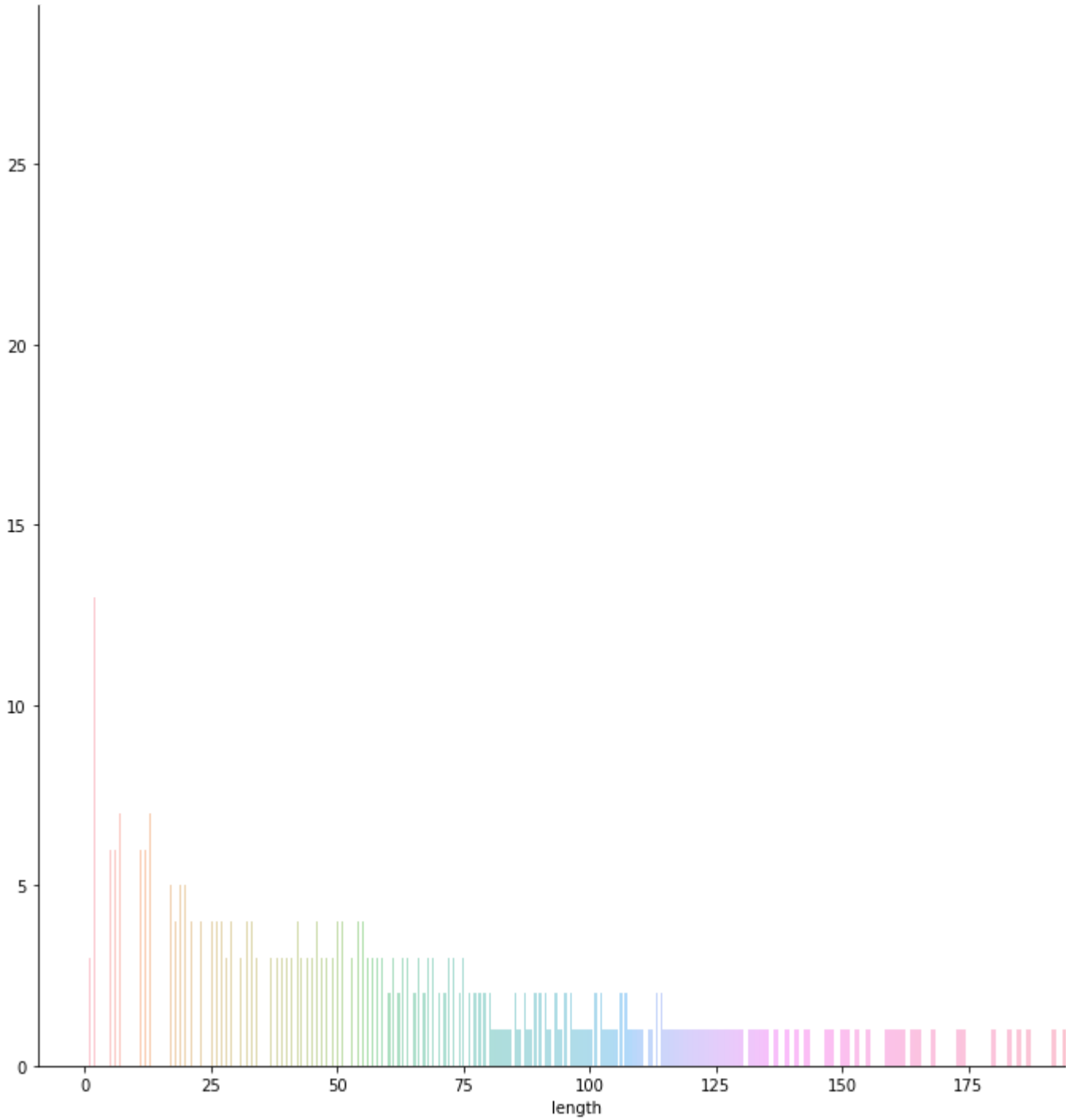



```
import numpy as np
import scipy
import matplotlib.pyplot as plt
import seaborn as sns

x = len_arr # generate samples from normal distribution (discrete data)
norm_cdf = scipy.stats.norm.pdf(x) # calculate the cdf - also discrete

# plot the cdf
sns.lineplot(x=x, y=norm_cdf)
plt.show()
```




```
import seaborn as sns
import matplotlib.pyplot as plt
#lets zoom into it
copy_len_arr = (len_arr).copy()

copy_len_arr.sort()

#take only top 200
temp_len_arr = []
for len in copy_len_arr:
  if len > 200:
    break
  else:
    temp_len_arr.append(len)

temp_df = pd.DataFrame(temp_len_arr)
temp_df.columns = ['length']

sns.FacetGrid(temp_df, hue="length", height=10).map(sns.distplot, "length")
plt.show();
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/nonparametric/kde.py:487: RuntimeWarn
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
/usr/local/lib/python3.6/dist-packages/statsmodels/nonparametric/kdetools.py:34: Runtime
  FAC1 = 2*(np.pi*bw/RANGE)**2
/usr/local/lib/python3.6/dist-packages/numpy/core/_methods.py:217: RuntimeWarning: Degre
  keepdims=keepdims)
/usr/local/lib/python3.6/dist-packages/numpy/core/_methods.py:209: RuntimeWarning: inval
  ret = ret.dtype.type(ret / rcount)
```



From the above graph and average off all reply we'll take 20 as the maximum lim

```
data.head(2)
```

	question	reply	reply_length	reply Le
3	what is the battery made by volta credited to be?	start_ the first cell _end	14	
4	what important electrical unit was named in ho...	start_ the volt _end	8	

```
data = data.sample(frac=1)
```

```
data.head()
```

	question	reply	rep
1085	how is the climate in the city?	start_ the city is hot and humid _end	
579	where is harvesting wild turtles legal?	start_ florida _end	
2238	what expired on march?	start_ monroe's presidency _end	
693	what are the three segments of an ant?	start_ the head, mesosoma and metasoma are the...	
2080	who is the head of state of ghana?	start_ john agyekum kufuor _end	

From above graphs and data we observed that **maximun reply have less words.**

and **average length** if reply is nearly **25** words

We have **169** unique reply

Check for emails

```
all_questions = data['question'].values
```

```
#Preprocessing questions
#Remove the full stops from the dataframe replys

bad_chars = [';', ':', '!', "*",'.',')','(','?','-','--']

preprocessed_question = []
for question in all_questions:
  for i in bad_chars :
    question = question.replace(i, '')
  preprocessed_question.append(question)
```

```
data['question'] = preprocessed_question
```

```
data['question'] = preprocessed_question
```

Words stemming

```
#We'll not use Word steemming here as it can create problem in understanding grammar.
# #Performing stammering here
# # We are having less data
# from nltk.stem import PorterStemmer
# porter = PorterStemmer()
# stemmed_sent = []
# for sent in preprocessed_question:
#   word_arr = []
#   for word in sent.split():
#     word_arr.append(porter.stem(word))
#   temp_str = ""
#   for words in word_arr:
#     temp_str += words + " "
#   stemmed_sent.append(temp_str)

# data['question'] = stemmed_sent

#trimming the string the dataframe
#because after removal of the special character trimming might have lost
data = data.applymap(lambda x: x.strip() if isinstance(x, str) else x)
```

```
data.head()
```

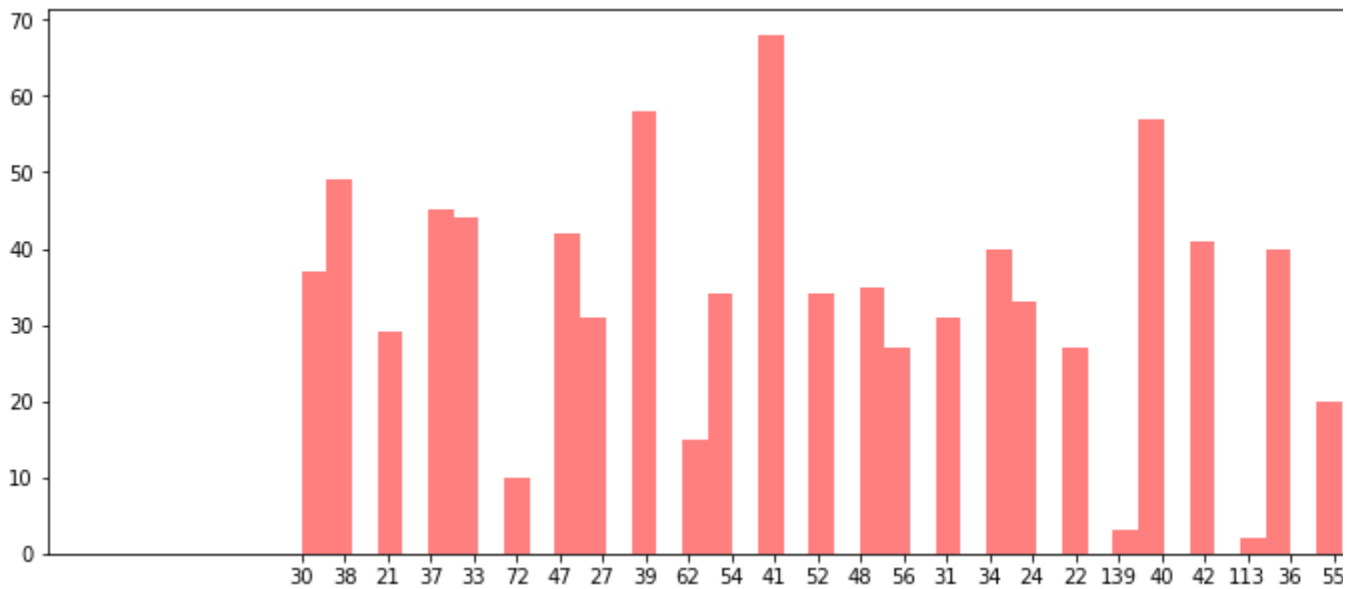
	question	reply	repl
1085	how is the climate in the city	start_ the city is hot and humid _end	
579	where is harvesting wild turtles legal	start_ florida _end	
2238	what expired on march	start_ monroe's presidency _end	
693	what are the three segments of an ant	start_ the head, mesosoma and metasoma are the...	
2080	who is the head of state of ghana	start_ john agyekum kufuor _end	

```
del len
```

```
import matplotlib
matplotlib.rc('figure', figsize=[50,5])
#Getting the length of each reply
data["question_length"]= data["question"].str.len()
len_questions = data["question_length"].values

#converting datatype to string
```

```
len_str_arr = []
for num in len_questions:
  len_str_arr.append(str(num))

no_reply_counter = 0
for num in len_questions:
  if num == 0:
    no_reply_counter += 1

temp_arr = []
for indx in len_questions:
  if indx == 0:
    continue
  else:
    temp_arr.append(indx)

print("number of email don' have any reply", end = ' : ')
print(no_reply_counter)
print('*' * 50)

print('Max length', end = ' : ')
print(max(list(len_questions)))
print('*' * 50)

print('Min Length', end = ' : ')
print(min(list(temp_arr)))
print('*' * 50)

print("Average Length", end = ' : ')
print((sum(list(len_questions)))/(len(list(len_questions))))
print('*' * 50)

print("No of unique reply", end = ' : ')
print(len(list(set(len_questions))))
print('*' * 50)


print('-' * 50)
print("Distribution of the words")
print('-' * 50)

plt.hist(len_str_arr, 200,
         histtype='bar',
         facecolor='r',
         alpha=0.5)
plt.show()
```

```
number of email don' have any reply : 0
**************************************************
Max length : 246
**************************************************
Min Length : 4
**************************************************
Average Length : 48.07340946166395
**************************************************
No of unique reply : 119
**************************************************
--------------------------------------------------
Distribution of the words
--------------------------------------------------
```



From the above graph we can observe that we have minimum of **4** character in the email

maximum length of the questions is **270**

and we have unique distribution of **156** questions.

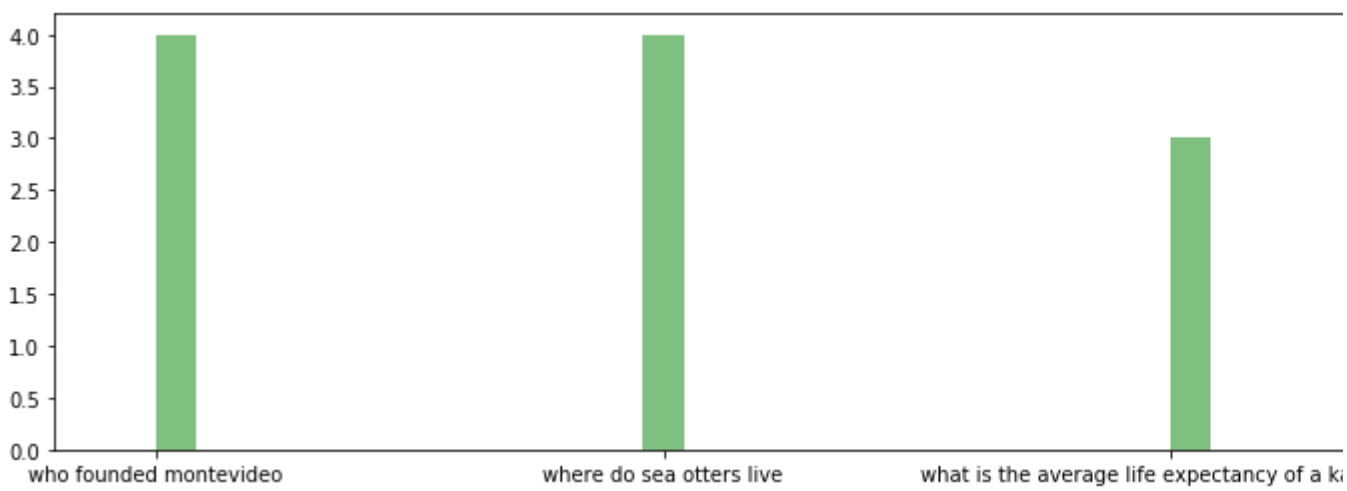
Top Unique distribution of questions in dataset

```
question_list = list(data['question'].values)
questions_dict = {i:question_list.count(i) for i in question_list}
```

```
from collections import OrderedDict
questions_dict_sorted = OrderedDict(sorted(questions_dict.items(), key=lambda x: x[1]))
```

```
questions = []
keys = []
for item in questions_dict_sorted.items():
```

```
for item in questions_dict_sorted.items():
  questions.append(item[0])
  keys.append(item[1])


questions.reverse()
keys.reverse()


#top 10 reply
top_5_questions = questions[:5]
top_5_keys = keys[:5]


temp_questions_list = []
i = 0
for key in top_5_keys:
  for _ in range(0,key):
    temp_questions_list.append(top_5_questions[i])
  i += 1


import matplotlib
matplotlib.rc('figure', figsize=[20,4])
plt.hist(temp_questions_list, 50,
         histtype='bar',
         facecolor='g',
         alpha=0.5)
plt.show()
```



Here we can see the top 5 repeated questions,

It's not looking usefull for any interpretation

```
print(data.head(5))
print(data.tail(5))
```

```
                                      question  ... question_length
1085          how is the climate in the city  ...              30
579   where is harvesting wild turtles legal  ...              38
2238                   what expired on march  ...              21
693    what are the three segments of an ant  ...              37
2080       who is the head of state of ghana  ...              33

[5 rows x 5 columns]
                                                question  ... question_length
3      what is the battery made by volta credited to be  ...              48
260                 what is the smallest species of fox  ...              35
105   what is the name of a university or similar in...  ...              97
1459  where do most people in urban saint petersburg...  ...              51
1600            what is the official language of turkey  ...              39

[5 rows x 5 columns]
```

Here we can **combine the Article Title with the question** which can help in getting better suggestions

Modelling Approaches:

here before modeling we can stammer the sentences : Playing

- ***Seq2seq model***
- ***Bi-Directional LSTM***
- ***Attention Layer***
- ***Beam Search***

Modeling

```
# Vocabulary of question
all_email_words=[]
for quest in data.question:
    for word in quest.split():
        if word not in all_email_words:
            all_email_words.append(word)

# Vocabulary of reply
all_reply_words=[]
for mar in data.reply:
    for word in mar.split():
        if word not in all_reply_words:
            all_reply_words.append(word)


all_email_words = list(set(all_email_words))
```

```
all_reply_words = list(set(all_reply_words))
```

```
data.head()
```

	question	reply	repl
1085	how is the climate in the city	start_ the city is hot and humid _end	
579	where is harvesting wild turtles legal	start_ florida _end	
2238	what expired on march	start_ monroe's presidency _end	
693	what are the three segments of an ant	start_ the head, mesosoma and metasoma are the...	
2080	who is the head of state of ghana	start_ john agyekum kufuor _end	

```
len("hello")
```

```
5
```

```
import numpy as np
lenght_list=[]
for l in data.question.values:
    lenght_list.append(len(l.split(' ')))
max_length_src = max(lenght_list)
max_length_src
```

```
46
```

```
#getting the max length of the list
lenght_list=[]
for l in data.reply.values:
    lenght_list.append(len(l.split(' ')))
max_length_tar = np.max(lenght_list)
max_length_tar
```

```
158
```

```
#All input words
input_words = sorted(list(all_email_words))
target_words = sorted(list(all_reply_words))

# get the length of the vocabalary // Kitne words // this will help while performing embeddin
num_encoder_tokens = len(all_email_words) + 1
num_decoder_tokens = len(all_reply_words) + 1

num_encoder_tokens, num_decoder_tokens
```

```
# from word to token we can get
input_token_index = dict([(word, i+1) for i, word in enumerate(input_words)])
target_token_index = dict([(word, i+1) for i, word in enumerate(target_words)])
```

```
input_token_index['three']
```

```
2597
```

```
# from token to word we can get
reverse_input_char_index = dict((i, word) for word, i in input_token_index.items())
reverse_target_char_index = dict((i, word) for word, i in target_token_index.items())
```

```
# reverse_input_char_index[3940]
```

```
from sklearn.utils import shuffle
data = shuffle(data)
data.head(2)
```

	question	reply	reply_le
1309	how many civilians died in the 1998 us embassy...	start_ over two hundred _end	
492	what is the order of santiago	start_ a spanish knightly order _end	

```
# Train - Test Split
from sklearn.model_selection import train_test_split
X, y = data.question, data.reply
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1)
X_train.shape, X_test.shape
```

```
((1655,), (184,))
```

```
y_train.head()
```

```
631                                 start_ napoleon _end
2695    start_ in 1990, goh chok tong succeeded lee as...
1100    start_ it lies on a plain, approximately eight...
2727    start_ that his children had been educated wit...
753     start_ yes, the gendarmenmarkt borders the fre...
Name: reply, dtype: object
```

```
X_train.head()
```

```
     631                                 who made volta a count
questions = X.values
replies =  y.values

all_txt_data = questions + replies


all_words = list(set(all_email_words + all_reply_words))
eng_words = list(set(all_email_words))


vocab_len = len(eng_words)
latent_dim = 300


import pickle
glove_pickel = open("drive/My Drive/DonarChoose/glove_vectors","rb")
glove_ = pickle.load(glove_pickel)


vector = glove_.get(word)


embedded_matrix = np.zeros((vocab_len, latent_dim))
for word, i in input_token_index.items():
    vector = glove_.get(word)
    if vector is not None:
      embedded_matrix[i] = vector
print(embedded_matrix.shape)
```

```
(2888, 300)
```

```
def generate_batch(X, y, batch_size=1):
    while True:
      # in every batch we are sending (batch_size) of sentences
        for j in range(0, len(X), batch_size):
            encoder_input_data = np.zeros((batch_size, max_length_src),
                    dtype='float32')
            decoder_input_data = np.zeros((batch_size, max_length_tar),
                    dtype='float32')
            decoder_target_data = np.zeros((batch_size, max_length_tar,
                    num_decoder_tokens), dtype='float32')

            # for all the sentences in the batch
            for (i, (input_text, target_text)) in enumerate(zip(X[j:j+ batch_size], y[j:j + b

                # for words in each sentence //question
                for (t, word) in enumerate(input_text.split()):  # question wale text ko toke
                    encoder_input_data[i,t] = input_token_index[word]  # encoder input seq

                # for each word in   Sentence
                for (t, word) in enumerate(target_text.split()):  # t is the length of the se
                    if t < len(target_text.split()) -1:
```

```
                        decoder_input_data[i,t] = target_token_index[word]  # decoder input s
                    if t > 0: #>0 cz we have to remove word start_
                        #one-hot encoding
                        #and making it to the output shape from decoder
                        decoder_target_data[i, t - 1,
                                target_token_index[word]] = 1.

            return ([encoder_input_data, decoder_input_data],
                   decoder_target_data)  # this will help multiple return// like closure in s
```

```
#https://stackoverflow.com/questions/56097089/how-to-fix-name-embedding-is-not-defined-in-ker
from keras.layers import Dense, Dropout, BatchNormalization, Input, Flatten, concatenate,Embe
embedded_layer = Embedding(vocab_len,latent_dim,weights=[embedded_matrix],input_length=1500,t


from keras.layers import Dense, Dropout, Embedding, LSTM, Bidirectional,Concatenate,TimeDistr
from keras.layers import Input, LSTM, Embedding, Dense
from keras.models import Model

# Encoder
latent_dim = 300 # how much dimension of output we want form the embedding layer
encoder_inputs = Input(shape=(None,)) # mentioning the input shape row and collumns are still

# **num_encoder_tokens** vocabulary size we are having

# enc_emb =  Embedding(num_encoder_tokens, latent_dim, mask_zero = True)(encoder_inputs) #Emb
enc_emb = embedded_layer(encoder_inputs)


#training the LSTM on encoder
encoder_lstm = Bidirectional(LSTM(128, return_sequences=True, return_state=True))
encoder_output, forward_h, forward_c, backward_h, backward_c = encoder_lstm(enc_emb)
state_h = Concatenate()([forward_h, backward_h])
state_c = Concatenate()([forward_c, backward_c])


encoder_states = [state_h, state_c] #output and the cell state


num_decoder_tokens
```

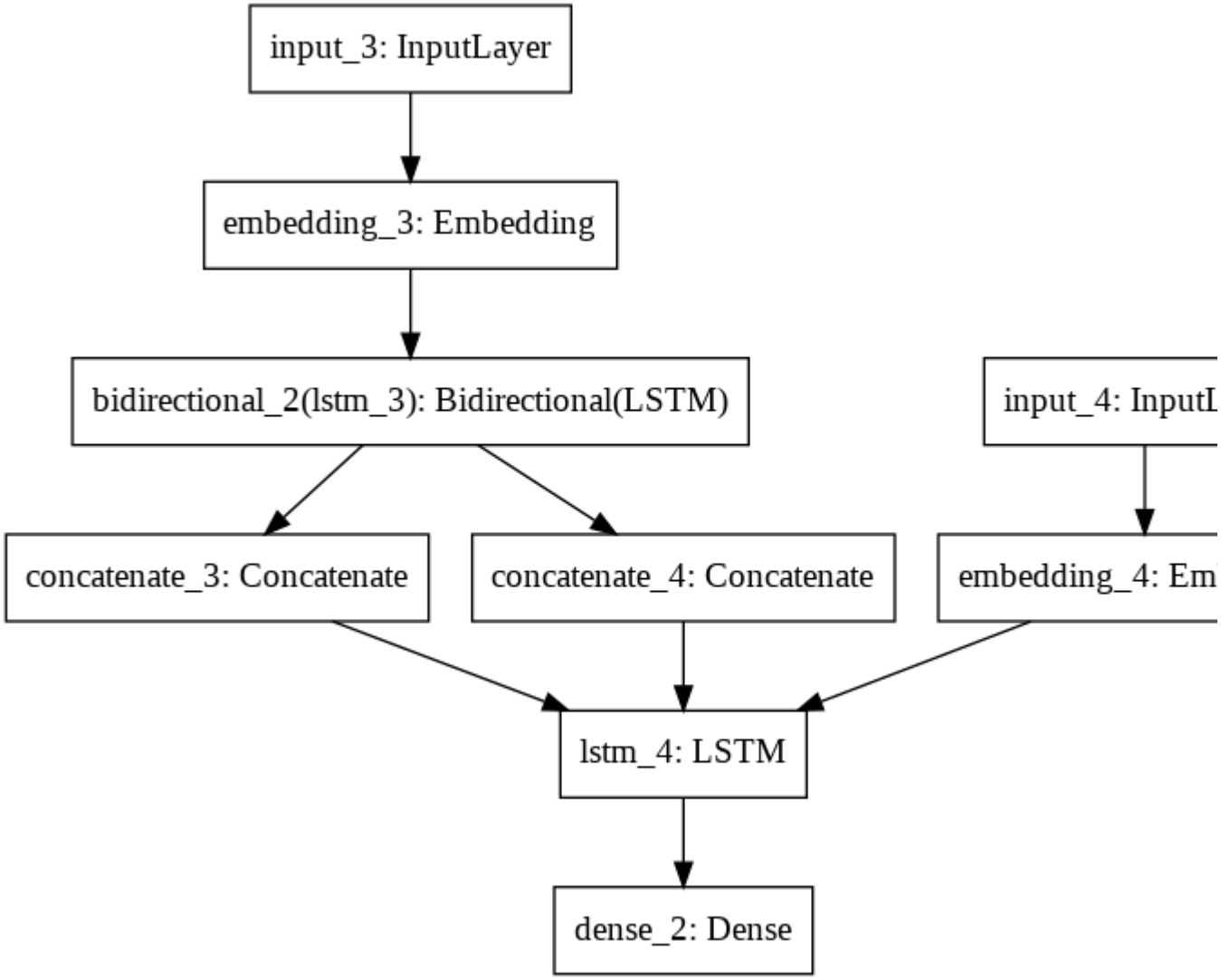
3776

```
import tensorflow as tf
from tensorflow.keras import layers, models
#  Set up the decoder, using `encoder_states` as initial state.
decoder_inputs = Input(shape=(None,)) #Reply's input
dec_emb_layer = Embedding(num_decoder_tokens, latent_dim, mask_zero = True)
dec_emb = dec_emb_layer(decoder_inputs) # Reply sikhave hum Embedding layer ko
```

```
dec_emb = dec_emb_layer(decoder_inputs) # Reply Lookup from Embedding layer to

# We set up our decoder to return full output sequences,
# and to return internal states as well. We don't use the
# return states in the training model, but we will use them in inference.


decoder_lstm = LSTM(256, return_sequences=True, return_state=True)
decoder_output, decoder_h, decoder_c = decoder_lstm(dec_emb,initial_state=encoder_states)
decoder_states = [decoder_h, decoder_c]

# logits = Dense(num_decoder_tokens, name='logits')(drop_layer)
# logits = Lambda(lambda x: x / temperature, name='Temperature')(logits) # this layer brings


decoder_dense = (Dense(num_decoder_tokens, activation='softmax'))
decoder_outputs = decoder_dense(decoder_output)

# Define the model that will turn
# `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
model = Model([encoder_inputs, decoder_inputs], decoder_outputs)


model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['acc'])


print(model.summary())
```

```
Model: "model_2"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_3 (InputLayer)            (None, None)         0
__________________________________________________________________________________________________
embedding_3 (Embedding)         (None, 1500, 300)    866400      input_3[0][0]
__________________________________________________________________________________________________
input_4 (InputLayer)            (None, None)         0
__________________________________________________________________________________________________
bidirectional_2 (Bidirectional) [(None, 1500, 256),  439296      embedding_3[0][0]
__________________________________________________________________________________________________
embedding_4 (Embedding)         (None, None, 300)    1132800     input_4[0][0]
__________________________________________________________________________________________________
concatenate_3 (Concatenate)     (None, 256)          0           bidirectional_2[0][1]
                                                                 bidirectional_2[0][3]
__________________________________________________________________________________________________
concatenate_4 (Concatenate)     (None, 256)          0           bidirectional_2[0][2]
                                                                 bidirectional_2[0][4]
__________________________________________________________________________________________________
lstm_4 (LSTM)                   [(None, None, 256),  570368      embedding_4[0][0]
                                                                 concatenate_3[0][0]
                                                                 concatenate_4[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, None, 3776)   970432      lstm_4[0][0]
==================================================================================================
Total params: 3,979,296
Trainable params: 3,112,896
Non-trainable params: 866,400
__________________________________________________________________________________________________
None
```

```
from keras.utils import plot_model
plot_model(model, to_file='multilayer_perceptron_graph.png')
```




```
y_train.head()
```

```
631                                 start_ napoleon _end
2695    start_ in 1990, goh chok tong succeeded lee as...
1100    start_ it lies on a plain, approximately eight...
2727    start_ that his children had been educated wit...
753     start_ yes, the gendarmenmarkt borders the fre...
Name: reply, dtype: object
```

```
#Will Crash if data is very big,because we have given batch size as len(X_train) which will c
enc_dcd_inp_data, decdr_trgt_data = generate_batch(X_train,y_train,batch_size = len(X_train))
```

```
model.fit(enc_dcd_inp_data,decdr_trgt_data,batch_size=100,epochs=50,verbose=1) # model.fit is
```

```
Epoch 1/50
1655/1655 [==============================] - 14s 8ms/step - loss: 7.0074 - acc: 0.1299
Epoch 2/50
1655/1655 [==============================] - 12s 7ms/step - loss: 6.3962 - acc: 0.1542
Epoch 3/50
1655/1655 [==============================] - 12s 7ms/step - loss: 6.2579 - acc: 0.1559
Epoch 4/50
1655/1655 [==============================] - 12s 7ms/step - loss: 6.1234 - acc: 0.1562
Epoch 5/50
1655/1655 [==============================] - 12s 7ms/step - loss: 5.9971 - acc: 0.1660
Epoch 6/50
1655/1655 [==============================] - 12s 7ms/step - loss: 5.8673 - acc: 0.1794
Epoch 7/50
1655/1655 [==============================] - 11s 7ms/step - loss: 5.7430 - acc: 0.1837
Epoch 8/50
1655/1655 [==============================] - 12s 7ms/step - loss: 5.6226 - acc: 0.1869
Epoch 9/50
1655/1655 [==============================] - 12s 7ms/step - loss: 5.5110 - acc: 0.1883
Epoch 10/50
1655/1655 [==============================] - 12s 7ms/step - loss: 5.3809 - acc: 0.1917
Epoch 11/50
1655/1655 [==============================] - 12s 7ms/step - loss: 5.2556 - acc: 0.1966
Epoch 12/50
1655/1655 [==============================] - 12s 7ms/step - loss: 5.1298 - acc: 0.2010
Epoch 13/50
1655/1655 [==============================] - 12s 7ms/step - loss: 5.0032 - acc: 0.2061
Epoch 14/50
1655/1655 [==============================] - 12s 7ms/step - loss: 4.8718 - acc: 0.2135
Epoch 15/50
1655/1655 [==============================] - 11s 7ms/step - loss: 4.7430 - acc: 0.2207
Epoch 16/50
1655/1655 [==============================] - 12s 7ms/step - loss: 4.6127 - acc: 0.2277
Epoch 17/50
1655/1655 [==============================] - 12s 7ms/step - loss: 4.4863 - acc: 0.2346
Epoch 18/50
1655/1655 [==============================] - 12s 7ms/step - loss: 4.3651 - acc: 0.2404
Epoch 19/50
1655/1655 [==============================] - 12s 7ms/step - loss: 4.2369 - acc: 0.2517
Epoch 20/50
1655/1655 [==============================] - 11s 7ms/step - loss: 4.1084 - acc: 0.2615
Epoch 21/50
1655/1655 [==============================] - 12s 7ms/step - loss: 3.9880 - acc: 0.2735
Epoch 22/50
1655/1655 [==============================] - 12s 7ms/step - loss: 3.8711 - acc: 0.2832
Epoch 23/50
1655/1655 [==============================] - 12s 7ms/step - loss: 3.7462 - acc: 0.2935
Epoch 24/50
1655/1655 [==============================] - 12s 7ms/step - loss: 3.6304 - acc: 0.3082
Epoch 25/50
1655/1655 [==============================] - 12s 7ms/step - loss: 3.4962 - acc: 0.3322
Epoch 26/50
1655/1655 [==============================] - 11s 7ms/step - loss: 3.3877 - acc: 0.3432
Epoch 27/50
1655/1655 [==============================] - 12s 7ms/step - loss: 3.2621 - acc: 0.3678
Epoch 28/50
1655/1655 [==============================] - 12s 7ms/step - loss: 3.1475 - acc: 0.3860
Epoch 29/50
```

```
1655/1655 [==============================] - 12s 7ms/step - loss: 3.0247 - acc: 0.4101
Epoch 30/50
1655/1655 [==============================] - 12s 7ms/step - loss: 2.9100 - acc: 0.4339
Epoch 31/50
1655/1655 [==============================] - 12s 7ms/step - loss: 2.7982 - acc: 0.4602
Epoch 32/50
1655/1655 [==============================] - 12s 7ms/step - loss: 2.6809 - acc: 0.4860
Epoch 33/50
1655/1655 [==============================] - 12s 7ms/step - loss: 2.5680 - acc: 0.5050
Epoch 34/50
1655/1655 [==============================] - 12s 7ms/step - loss: 2.4659 - acc: 0.5293
Epoch 35/50
1655/1655 [==============================] - 12s 7ms/step - loss: 2.3671 - acc: 0.5506
Epoch 36/50
1655/1655 [==============================] - 11s 7ms/step - loss: 2.2514 - acc: 0.5772
Epoch 37/50
1655/1655 [==============================] - 11s 7ms/step - loss: 2.1550 - acc: 0.5975
Epoch 38/50
1655/1655 [==============================] - 11s 7ms/step - loss: 2.0616 - acc: 0.6162
Epoch 39/50
1655/1655 [==============================] - 12s 7ms/step - loss: 1.9552 - acc: 0.6418
Epoch 40/50
1655/1655 [==============================] - 12s 7ms/step - loss: 1.8677 - acc: 0.6592
Epoch 41/50
1655/1655 [==============================] - 11s 7ms/step - loss: 1.7791 - acc: 0.6765
Epoch 42/50
1655/1655 [==============================] - 12s 7ms/step - loss: 1.6985 - acc: 0.6911
Epoch 43/50
1655/1655 [==============================] - 11s 7ms/step - loss: 1.6099 - acc: 0.7092
Epoch 44/50
1655/1655 [==============================] - 12s 7ms/step - loss: 1.5336 - acc: 0.7261
Epoch 45/50
1655/1655 [==============================] - 11s 7ms/step - loss: 1.4563 - acc: 0.7429
Epoch 46/50
1655/1655 [==============================] - 12s 7ms/step - loss: 1.3744 - acc: 0.7623
Epoch 47/50
1655/1655 [==============================] - 12s 7ms/step - loss: 1.3130 - acc: 0.7664
Epoch 48/50
1655/1655 [==============================] - 12s 7ms/step - loss: 1.2444 - acc: 0.7812
Epoch 49/50
1655/1655 [==============================] - 12s 7ms/step - loss: 1.1732 - acc: 0.8012
Epoch 50/50
1655/1655 [==============================] - 11s 7ms/step - loss: 1.1150 - acc: 0.8132
<keras.callbacks.History at 0x7f837e724748>
```

Observation: After removing recurrent yes and no replies trianing accuracy is slowly increasing else it

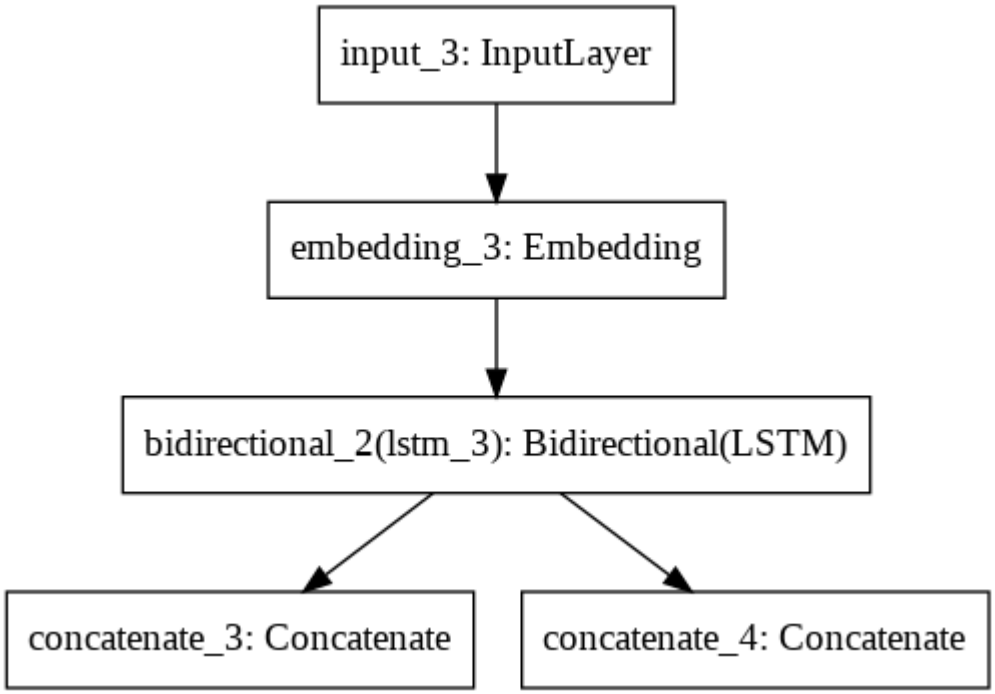
```
# model.fit_generator(generator = generate_batch(X_train, y_train, batch_size = 50),
#                     steps_per_epoch = 100,
#                     epochs=5,
#                     validation_data = generate_batch(X_test, y_test, batch_size = 50),
#                     validation_steps = 100)


#trying softmax layer with temperature
```

```
# def softmax_with_temperature(logits, temperature=1):
#   logits = logits / temperature
#   return np.exp(logits) / np.sum(np.exp(logits))


# import tensorflow as tf
# from tensorflow.keras import layers, models
# #  Set up the decoder, using `encoder_states` as initial state.
# decoder_inputs = Input(shape=(None,)) #Reply's input
# dec_emb_layer = Embedding(num_decoder_tokens, latent_dim, mask_zero = True)
# dec_emb = dec_emb_layer(decoder_inputs) # Reply sikhaye hum Embedding layer ko

# # We set up our decoder to return full output sequences,
# # and to return internal states as well. We don't use the
# # return states in the training model, but we will use them in inference.


# decoder_lstm = LSTM(256, return_sequences=True, return_state=True)
# decoder_output, decoder_h, decoder_c = decoder_lstm(dec_emb,initial_state=encoder_states)
# decoder_states = [decoder_h, decoder_c]

# logit_layer = Dense(698, name='logits')(decoder_output)

# decoder_dense = TimeDistributed(Dense(num_decoder_tokens, activation='softmax'))
# decoder_outputs = decoder_dense(logit_layer)

# # Define the model that will turn
# # `encoder_input_data` & `decoder_input_data` into `decoder_target_data`
# model = Model([encoder_inputs, decoder_inputs], decoder_outputs)


# model_sans_softmax = Model(inputs=model.input, outputs=model.get_layer("logits").output)
# model_logits = model_sans_softmax.predict(a)



# unsoftened_train_prob = softmax_with_temperature(model_logits, 1)


# unsoftened_train_prob.shape


decoder_outputs.shape
```

Inference

```
#Making new model so that we can predict the output
# taking all input and output of the encoder from the previous model to here
    #encoder_inputs: It's the tokenized and padded words from the question
encoder_model = Model(encoder_inputs, encoder_states) # this is not used for the input in the
```

```
from keras.utils import plot_model
plot_model(encoder_model, to_file='multilayer_perceptron_graph.png')
```



```
#giving input to decoder
decoder_state_input_h = Input(shape=(256,)) # this will be the input form the encoders # cz 2
decoder_state_input_c = Input(shape=(256,)) #cell state after the encoder

decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]

#Reply words tokenized
dec_emb2= dec_emb_layer(decoder_inputs) #Get the embeddings of the decoder sequence

#decoder ka output and cell state nikal liya
    # decoder_lstm = LSTM(256, return_sequences=True, return_state=True)
        # here dec_emb2 we passed the output from the embedding later which for trained on to
decoder_outputs, state_h2, state_c2 = decoder_lstm(dec_emb2, initial_state=decoder_states_inp

#added both states from the decoder
decoder_states = [state_h2, state_c2]

# Passing the output from the decoder model to the dense layer
decoder_output_dense = decoder_dense(decoder_outputs)
```

```
# Final decoder model
```

```
# decoder_inputs : Embedding layer ko Reply sentence tokenized
# decoder_states_inputs : this is from the encoder
# decoder_outputs2 : #ye decoder ka output he //predicted values from the decoder not the cel
# decoder_states2 : #ye output(h) & cell state h from decoder # not form LSTm this is coming


decoder_model = Model(
    [decoder_inputs] + decoder_states_inputs,
    [decoder_output_dense] + decoder_states)


print(decoder_model.summary())
```

```
Model: "model_4"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            (None, None)         0
__________________________________________________________________________________________________
embedding_4 (Embedding)         (None, None, 300)    1132800     input_4[0][0]
__________________________________________________________________________________________________
input_5 (InputLayer)            (None, 256)          0
__________________________________________________________________________________________________
input_6 (InputLayer)            (None, 256)          0
__________________________________________________________________________________________________
lstm_4 (LSTM)                   [(None, None, 256),  570368      embedding_4[1][0]
                                                                 input_5[0][0]
                                                                 input_6[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, None, 3776)   970432      lstm_4[1][0]
==================================================================================================
Total params: 2,673,600
Trainable params: 2,673,600
Non-trainable params: 0
__________________________________________________________________________________________________
None
```

```
from keras.utils import plot_model
plot_model(decoder_model, to_file='multilayer_perceptron_graph.png')
```


```
def decode_sequence(input_seq):
    # Encode the input as state vectors.
    states_value = encoder_model.predict(input_seq)
    # Generate empty target sequence of length 1.
    target_seq = np.zeros((1,1))
    # Populate the first character of target sequence with the start character.
    target_seq[0, 0] = target_token_index['start_']

    # Sampling loop for a batch of sequences
    # (to simplify, here we assume a batch of size 1).
    stop_condition = False
    decoded_sentence = ''
    while not stop_condition:
        output_tokens, h, c = decoder_model.predict([target_seq] + states_value)

        # Sample a token
        sampled_token_index = np.argmax(output_tokens[0, -1, :]) #need to check this logic
        sampled_char = reverse_target_char_index[sampled_token_index]
        decoded_sentence += ' '+sampled_char

        # Exit condition: either hit max length
        # or find stop character.
        if (sampled_char == '_end' or
           len(decoded_sentence) > 50):
            stop_condition = True

        # Update the target sequence (of length 1).
        target_seq = np.zeros((1,1))
        target_seq[0, 0] = sampled_token_index

        # Update states
        states_value = [h, c]

    return decoded_sentence


def generate_batch(X, y, batch_size):
    while True:
      # in every batch we are sending (batch_size) of sentences
        for j in range(0, len(X), batch_size):
            encoder_input_data = np.zeros((batch_size, max_length_src),
                    dtype='float32')
            decoder_input_data = np.zeros((batch_size, max_length_tar),
                    dtype='float32')
            decoder_target_data = np.zeros((batch_size, max_length_tar,
```

```
        num_decoder_tokens), dtype='float32')

    # for all the sentences in the batch
    for (i, (input_text, target_text)) in enumerate(zip(X[j:j
        + batch_size], y[j:j + batch_size])):

        # for words in each sentence //question
        for (t, word) in enumerate(input_text.split()):  # question wale text ko toke
            encoder_input_data[i,t] = input_token_index[word]  # encoder input seq

        # for each word in   Sentence
        for (t, word) in enumerate(target_text.split()):  # t is the length of the se
            if t < len(target_text.split()) - 1:
                decoder_input_data[i,t] = target_token_index[word]  # decoder input s
            if t >=0 :
                #one-hot encoding
                #and making it to the output shape from decoder
                decoder_target_data[i, t,
                        target_token_index[word]] = 1.

    yield ([encoder_input_data, decoder_input_data],
           decoder_target_data)  # this will help multiple return// like closure in s
```

```
X_test.shape
```

```
(184,)
```

```
temp_arr = []
for index in range(1,184):
  x =index
  train_gen = generate_batch([X_test.values[x]], [y_test.values[x]], batch_size = 10)

  (input_seq, actual_output), end_temp = next(train_gen)

  predicted_sentence = decode_sequence(input_seq)
  print('question/email: ',X_train.values[x])
  reply = y_train.values[x].replace('start_', '')
  reply = reply.replace('_end','.')
  print('original reply: ',reply)
  predicted_txt = predicted_sentence.replace('_end','.')
  print('predicted reply: ',predicted_txt)
  temp_arr.append(predicted_txt)
  print("\n")
  print("*"*50)
  print("\n")
```

```
question/email:  when did goh chok tong succeed lee as prime minister
original reply:   in 1990, goh chok tong succeeded lee as prime minister .
predicted reply:   the three heaviest cats in the world are two are two


**************************************************


question/email:  what contributes to frequent flooding in jakarta
original reply:   it lies on a plain, approximately eight meters above the sea level .
predicted reply:   the reichstag building .


**************************************************


question/email:  what was roosevelt's justification for supporting desegregation of schc
original reply:   that his children had been educated with other races and there was not
predicted reply:   the capital of the drum is an western part is the light


**************************************************


question/email:  does the gendarmenmarkt border the french cathedral
original reply:   yes, the gendarmenmarkt borders the french cathedral .
predicted reply:   loons or divers, grebes, gallinules, and coots .


**************************************************


question/email:  where is qatar's telecommunication system centered
original reply:   doha .
predicted reply:   the first novel in the white engine .


**************************************************


question/email:  how many swedish speakers were reported in canada in 2001
original reply:   there are 16,915 reported swedish speakers in canada .
predicted reply:   a holt .


**************************************************


question/email:  how many swedish speakers were reported in canada in 2001
original reply:   16,91500 .
predicted reply:   the reichstag building .


**************************************************


question/email:  what are deposited in holes dug into mud or sand
```

```
original reply:   eggs .
predicted reply:   the volt .


**************************************************


question/email:  is montreal's economy the third largest of all cities in canada
original reply:   no, montreal's economy is the second largest of all cities in canada .
predicted reply:   he did not attend .


**************************************************


question/email:  when did qatar become an independent state
original reply:   september 3, 1971 .
predicted reply:   by their hardened, often darkened head, the presence


**************************************************


question/email:  what was the name of james watt's mother
original reply:   agnes muirhead .
predicted reply:   a "melburnian" .


**************************************************


question/email:  what makes san francisco among the topten north american destinations f
original reply:   a large hotel infrastructure and a world-class convention facility in
predicted reply:   3 .


**************************************************


question/email:  why did coolidge not attend law school
original reply:   it was too expensive .
predicted reply:   japan .


**************************************************


question/email:  how many noun classes does swahili have
original reply:   swahili has sixteen noun classes .
predicted reply:   78 million .


**************************************************


question/email:  is modern korean written in columns or rows
original reply:   it is written in rows .
predicted reply:   felicita mazz .
```

```
**************************************************


question/email:  what prompted the city to upgrade its building codes
original reply:   the threat of major earthquakes .
predicted reply:   in the battle of trenton .


**************************************************


question/email:  what is ottawa's junior ice hockey team
original reply:   the ottawa 67's .
predicted reply:   yes, there are more than more million .


**************************************************


question/email:  who took a hard line that reduced violence by groups like the ku klux k
original reply:   grant .
predicted reply:   a adult volume can generally be increased by making


**************************************************


question/email:  where is irving berlin's piano located
original reply:   in the smithsonian museum .
predicted reply:   the celsius of david, completed called by the light


**************************************************


question/email:  what is "tobogganing"
original reply:   tobogganing is when penguins slide on their bellies across the snow .
predicted reply:   michael faraday .


**************************************************


question/email:  how many suborders are turtles divided into
original reply:   three .
predicted reply:   michael faraday .


**************************************************


question/email:  what does avogadro's law state
original reply:   avogadro's law states that the relationship between the masses of the
predicted reply:   the three are cats in the bass drum are an identical
```

```
**************************************************


question/email:  what suborder of turtles is extinct
original reply:   paracryptodira .
predicted reply:   the united states of the kangaroo .


**************************************************


question/email:  for how many years did charlesaugustin de coulomb live
original reply:   70 .
predicted reply:   176215 km©÷ .


**************************************************


question/email:  how long is an adult cougar's paw print
original reply:   4 inches .
predicted reply:   felicita mazz .


**************************************************


question/email:  what used singapore as a strategic trading post along the spice route
original reply:   british .
predicted reply:   the capital of queen nefertiti is in the altes museum


**************************************************


question/email:  when was the six day war
original reply:   1967 .
predicted reply:   chinese is not chinese .


**************************************************


question/email:  did nikola tesla use a technique called picture thinking
original reply:   nikola tesla used a technique called picture thinking .
predicted reply:   the leopard has rosettes rather than cheetah's simple


**************************************************


question/email:  when did canada have one of the largest armed forces in the world
original reply:   world war ii .
predicted reply:   in 1819 .


**************************************************
```