Sentence Correction using RNN

Problem Statement:

Given a text message that contains corruputed english language(like social media text messages), we need to convert those messages to standard Standard English text.

Data Overview:

In this Modern World everything has become Digitalized. Everyone in the World has atleast one Mobile phone or computer.Due to evolution, messages to be delivered to people are done via Mobile phones or computers.These are fast and easy way for communication. Moreover most of the pepole in world now started to use short forms for conveying messages. So while performing various NLP based text Problems, the words need better text preprocessing to get better performance.So by using various RNN techniques we can improve the text or correct the incorrect text to a standard English so that it can help ML or DL to get better Performance on NLP based Models.

The dataset is around 2000 text messages which contains both incorrect and correct sentence.

Example1:

-->***Input***:'U wan me to "chop" seat 4 u nt?'

-->***Output:***'Do you want me to reserve seat for you or not?'

Example2:

-->***Input:***'Yup. U reaching. We order some durian pastry already. U come quick.'

-->***Output:***'Yeap. You reaching? We ordered some Durian pastry already. You come quick.'

Business Objective and Constraints:

Since this could be used for preprocessing there is not much requirement for very faster results.

In [215]:

```
#Importing necessary Libraries
import numpy as np
import pandas as pd
import datetime
import matplotlib.pyplot as plt
import seaborn as sns
import os
import pickle
import tqdm
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
%matplotlib inline
```

In [216]:

```
#In the text file the source text(corrupted text) are arranged as 1st line, 4th line, 7th line......etc.
#Same way the target text(standard English) are arranged as 2nd line,5th line,8th line......etc.
#So we need extract them from files,
#And store them in seperate list.
with open('sentencecorrection.txt', encoding="utf8") as f:#reading the files
    source=[]#for storing source text
    target=[]#for storing target text
    for i,j in enumerate(f):#looping over every lines
        if i%3==0:
            source.append(j)#appending the lines which contains source text
        if i%3==1:
            target.append(j)#appending the lines which contains target text
```

In [217]:

```
source[:4]#first four source text
```

Out[217]:

```
['U wan me to "chop" seat 4 u nt?\n',
 'Yup. U reaching. We order some durian pastry already. U come quick.\n',
 'They become more ex oredi... Mine is like 25... So horrible n they did less things than last
time...\n',
 "I'm thai. what do u do?\n"]
```

In [250]:

```
target[:4]#first four target text
```

Out[250]:

```
['Do you want me to reserve seat for you or not?\n',
 'Yeap. You reaching? We ordered some Durian pastry already. You come quick.\n',
 'They become more expensive already. Mine is like 25. So horrible and they did less things than I did l
ast time.\n',
 "I'm Thai. What do you do?\n"]
```

In [251]:

```
data={'source':source,'target':target}
df=pd.DataFrame(data)#creating DataFrame using the data
```

In [247]:

```
df.head(4)#displaying four rows
```

Out[247]:

	source	target
0	U wan me to "chop" seat 4 u nt?\n	Do you want me to reserve seat for you or not?\n
1	Yup. U reaching. We order some durian pastry a...	Yeap. You reaching? We ordered some Durian pas...
2	They become more ex oredi... Mine is like 25.....	They become more expensive already. Mine is li...
3	I'm thai. what do u do?\n	I'm Thai. What do you do?\n

In [239]:

```
print("No of data points:")
print(df.shape)#shape if the dataset
```

```
No of data points:
(2000, 2)
```

In [240]:

```
df.info()#info about the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   source  2000 non-null   object
 1   target  2000 non-null   object
dtypes: object(2)
memory usage: 31.4+ KB
```

In [241]:

```
df.isnull().sum()#checking for null values
```

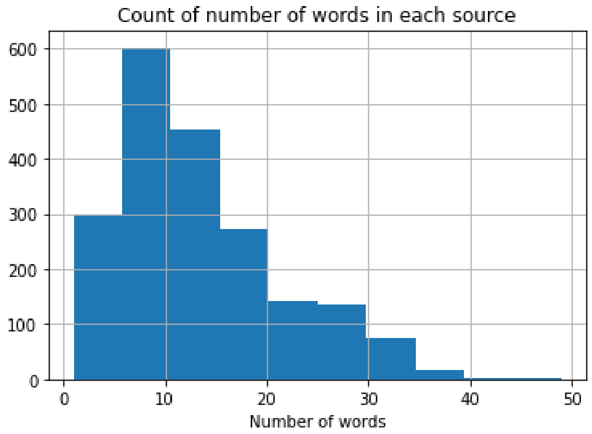
Out[241]:

```
source    0
target    0
dtype: int64
```

Words count for each sentence(Source):

In [242]:

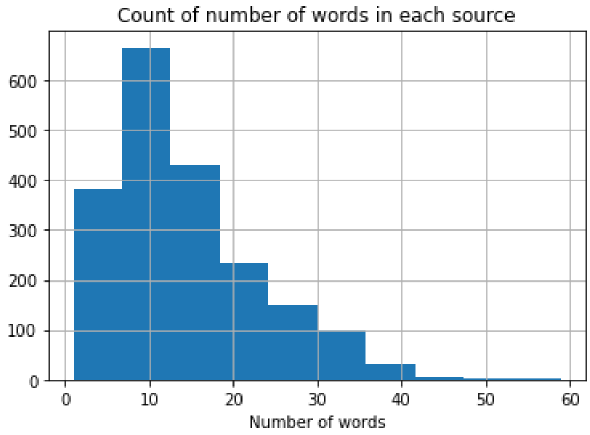
```
df['source'].str.split().apply(len).hist()#creating a histogram plot of count of words in the whole text
plt.title('Count of number of words in each source')
plt.xlabel('Number of words')
plt.show()
```



Word count for each sentence(Target):

In [225]:

```
df['target'].str.split().apply(len).hist()#creating a histogram plot of count of words in the whole text
plt.title('Count of number of words in each source')
plt.xlabel('Number of words')
plt.show()
```



Observation:

```
   -->From the above plots we can see the Count of no of number of words in source varies between
   0 to 50 and in target the Count of number of words varies between 0 to 60 and their distibution
   are mostly similar since the target is just correted sentence of the source.
```

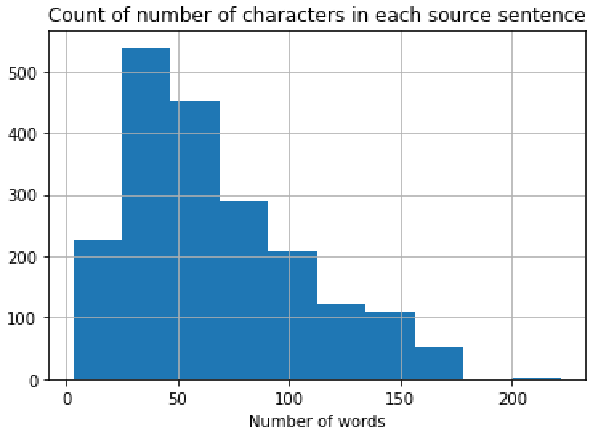
In [226]:

```
def length(text):#for calculating the length of the sentence
    return len(str(text))
```

Characters count for each sentence(Source):

In [227]:

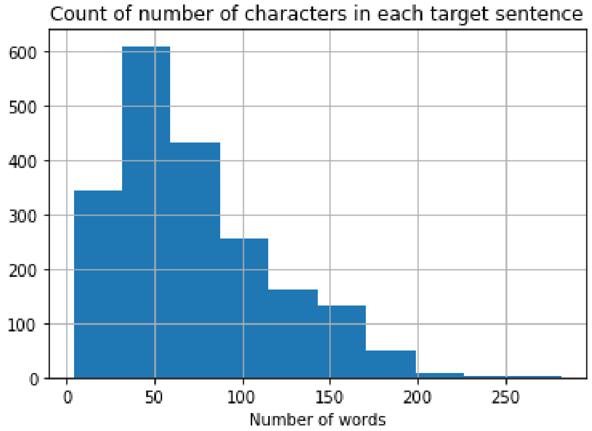
```
df['source'].apply(length).hist()#creating a histogram plot of count of words in the whole text data.
plt.title('Count of number of characters in each source sentence')
plt.xlabel('Number of words')
plt.show()
```



Characters count for each sentence(Target):

In [228]:

```
df['target'].apply(length).hist()#creating a histogram plot of count of words in the whole text data.
plt.title('Count of number of characters in each target sentence')
plt.xlabel('Number of words')
plt.show()
```



Observation:

```
From the above plots we can see that the Count of number of characters in each source sentence
varies between 0 to 2400
and similar way target sentence varies between 0 to 270.
```

In [229]:

```
print("No of data points remaining if we remove sentences of length greater than 150: ",len(df[df['source
print("No of data points remaining if we remove sentences of length greater than 160: ",len(df[df['source
print("No of data points remaining if we remove sentences of length greater than 170: ",len(df[df['source
print("No of data points remaining if we remove sentences of length greater than 180: ",len(df[df['source
```

```
No of data points remaining if we remove sentences of length greater than 150:  1907
No of data points remaining if we remove sentences of length greater than 160:  1977
No of data points remaining if we remove sentences of length greater than 170:  1996
No of data points remaining if we remove sentences of length greater than 180:  1997
```

In [230]:

```
print("No of data points remaining if we remove sentences of length greater than 160: ",len(df[df['target
print("No of data points remaining if we remove sentences of length greater than 170: ",len(df[df['target
print("No of data points remaining if we remove sentences of length greater than 180: ",len(df[df['target
print("No of data points remaining if we remove sentences of length greater than 190: ",len(df[df['target
print("No of data points remaining if we remove sentences of length greater than 200: ",len(df[df['target
print("No of data points remaining if we remove sentences of length greater than 210: ",len(df[df['target
```

```
No of data points remaining if we remove sentences of length greater than 160:  1885
No of data points remaining if we remove sentences of length greater than 170:  1937
No of data points remaining if we remove sentences of length greater than 180:  1958
No of data points remaining if we remove sentences of length greater than 190:  1978
No of data points remaining if we remove sentences of length greater than 200:  1990
No of data points remaining if we remove sentences of length greater than 210:  1995
```

Frequently occuring words in Source:

In [231]:

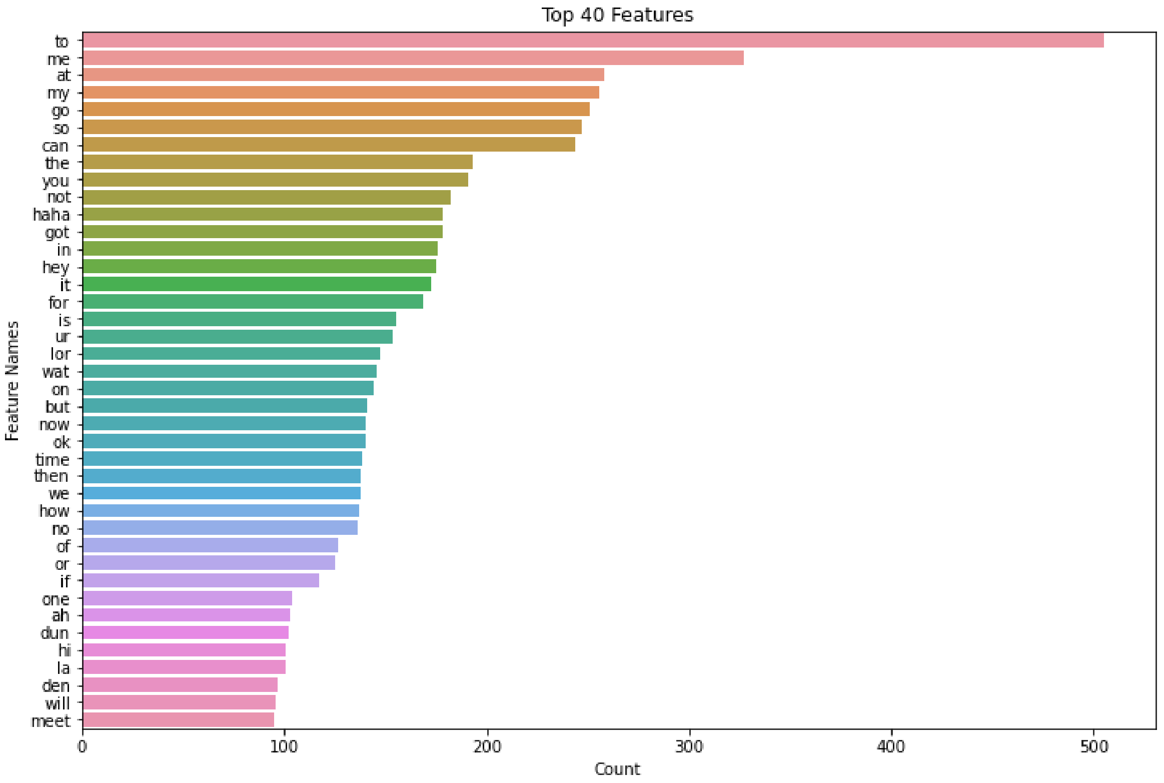
```
vect=CountVectorizer()#in the presence of stop words
output=vect.fit_transform(df['source'])
features=vect.get_feature_names()#here we are getting the unique feature names

#https://stackoverflow.com/questions/27488446/how-do-i-get-word-frequency-in-a-corpus-using-scikit-learn-
count=output.toarray().sum(axis=0)#here we are getting the count of unique words

df=pd.DataFrame(count,features)#Loading the feature and count to the DataFrame
df=df.sort_values(by=0,ascending=False)#Sorting the DataFrame to get the most occurances
df=df[:40]#Top 40 words with most word count
df
```

Out[231]:

	0
to	505
me	327
at	258
my	256
go	251
so	247
can	244
the	193
you	191
not	182
haha	178
got	178
in	176
hey	175
it	173
for	169
is	155
ur	154
lor	147
wat	146
on	144
but	141
now	140
ok	140
time	139
then	138
we	138
how	137
no	136
of	127
or	125
if	117
one	104
ah	103
dun	102
hi	101
la	101
den	97
will	96
meet	95

In [232]:

```
x=df.index#getting only the top 40 feature names
y=[df[0][i] for i in range(len(df))]#getting the count of top 40 feature names
plt.figure(figsize=(12,8))
sns.barplot(x=y,y=x)
plt.title('Top 40 Features')
plt.xlabel('Count')
plt.ylabel('Feature Names')
plt.show()
```



Frequently occuring words in Target:

In [243]:

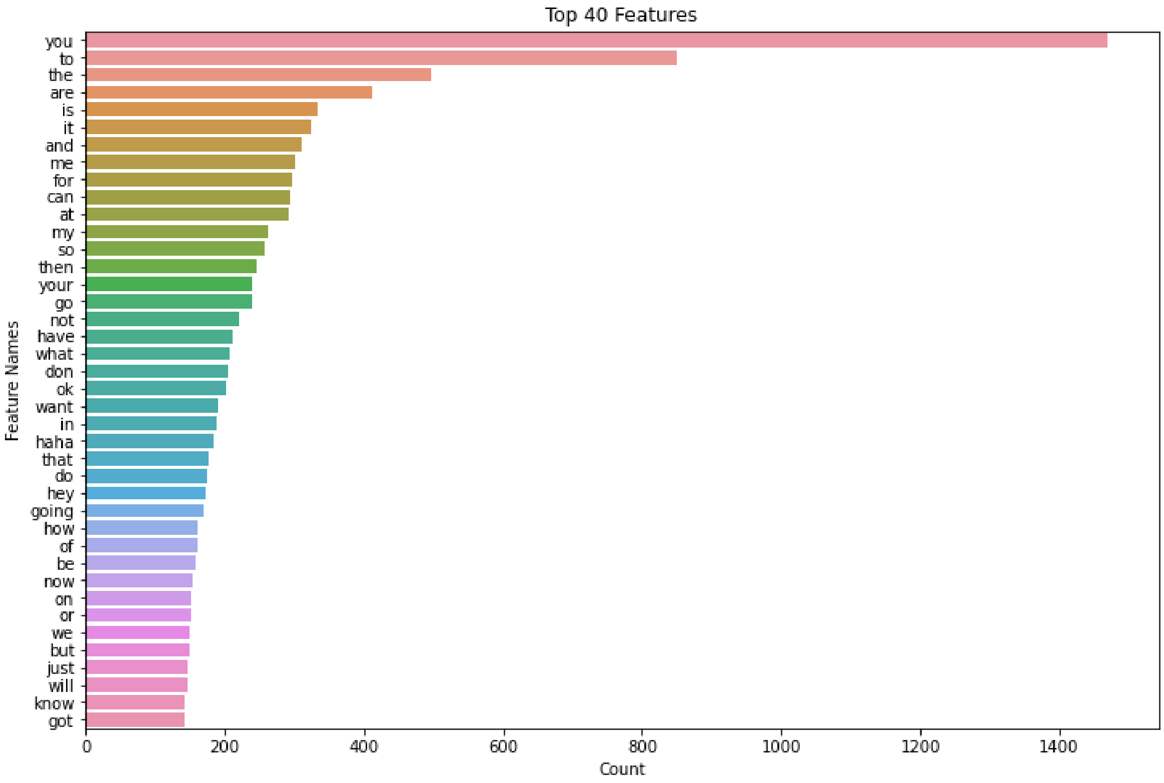
```
vect=CountVectorizer()#in the presence of stop words
output=vect.fit_transform(df['target'])
features=vect.get_feature_names()#here we are getting the unique feature names

#https://stackoverflow.com/questions/27488446/how-do-i-get-word-frequency-in-a-corpus-using-scikit-learn
count=output.toarray().sum(axis=0)#here we are getting the count of unique words

df=pd.DataFrame(count,features)#Loading the feature and count to the DataFrame
df=df.sort_values(by=0,ascending=False)#Sorting the DataFrame to get the most occurances
df=df[:40]#Top 40 words with most word count
df
```

Out[243]:

```
          0
  you  1469
   to   850
  the   498
  are   412
   is   334
   it   324
  and   310
   me   302
  for   297
  can   294
   at   291
   my   262
   so   258
 then   246
 your   239
   go   238
  not   221
 have   212
 what   206
  don   204
   ok   202
 want   190
   in   188
 haha   183
 that   178
   do   175
  hey   172
going   170
  how   161
   of   160
   be   158
  now   153
   on   152
   or   151
   we   149
  but   149
 just   148
 will   146
 know   142
  got   142
```

In [244]:

```
x=df.index#getting only the top 40 feature names
y=[df[0][i] for i in range(len(df))]#getting the count of top 40 feature names
plt.figure(figsize=(12,8))
sns.barplot(x=y,y=x)
plt.title('Top 40 Features')
plt.xlabel('Count')
plt.ylabel('Feature Names')
plt.show()
```



Observation:

```
By using the Count Vectorizer, we are getting the Top 40 Features that occur more frequently in
the source and target.
The above plots shows the words that are frequently occuring in the whole corpus.
```

Rare words in Source:

In [248]:

```
#here we are using TfidfVectorizer and fitting to source corpus and finding the rare words
from nltk.corpus import stopwords
vect=TfidfVectorizer()
output=vect.fit_transform(df['source'])
features=vect.get_feature_names()
idf_values=vect.idf_
df=pd.DataFrame(idf_values,features)
df=df.sort_values(by=0,ascending=False)
df=df[:40]
df
```

Out[248]:

	0
000pes	7.908255
matthew	7.908255
marketin	7.908255
marshmallow	7.908255
match	7.908255
math	7.908255
maths	7.908255
matric	7.908255
matriculation	7.908255
maxy	7.908255
melting	7.908255
maya	7.908255
mb	7.908255
meaningless	7.908255
measurement	7.908255
medic	7.908255
medical	7.908255
mel	7.908255
marine	7.908255
marina	7.908255
march	7.908255
map	7.908255
m1	7.908255
macadamia	7.908255
machine	7.908255
machines	7.908255
mad	7.908255
mahz	7.908255
mainland	7.908255
maintain	7.908255
maintenance	7.908255
makeup	7.908255
malaysian	7.908255
mama	7.908255
manager	7.908255
mangosteen	7.908255
manners	7.908255
melnite	7.908255
mem	7.908255
monkees	7.908255

Rare words in Target:

In [252]:

```
#here we are using TfidfVectorizer and fitting to target corpus and finding the rare words
from nltk.corpus import stopwords
vect=TfidfVectorizer()
```

```
output=vect.fit_transform(df['target'])
features=vect.get_feature_names()
idf_values=vect.idf_
df=pd.DataFrame(idf_values,features)
df=df.sort_values(by=0,ascending=False)
df=df[:40]
df
```

Out[252]:

	0
00	7.908255
malayu	7.908255
materials	7.908255
match	7.908255
mass	7.908255
marshmallow	7.908255
marketing	7.908255
marine	7.908255
marina	7.908255
march	7.908255
map	7.908255
manners	7.908255
mangosteen	7.908255
manager	7.908255
manage	7.908255
malaysian	7.908255
mathematics	7.908255
makeup	7.908255
maintenance	7.908255
maintain	7.908255
mainland	7.908255
mad	7.908255
machines	7.908255
machine	7.908255
macaroni	7.908255
macadamias	7.908255
mac	7.908255
m1	7.908255
lush	7.908255
lungs	7.908255
math	7.908255
matthew	7.908255
picnic	7.908255
merchandiser	7.908255
ming	7.908255
min	7.908255
mimi	7.908255
mike	7.908255
mid	7.908255
mick	7.908255

Observation:

```
By using idf_values we could get the rare words in the whole corupus.Similarly we are getting
the rare words for
both source and text.
```