BERT as Embedding

Task: Finding Semantic Textual Similarity

```
# Ignore all your warnings
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

# Loading Libraries
import datetime
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import f1_score
import pylab
import scipy.stats as stats
from scipy.stats import boxcox

import re
import pickle

from tqdm import tqdm
import os
from wordcloud import WordCloud
from matplotlib_venn import venn2
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from textblob import TextBlob

import nltk
from nltk.probability import FreqDist
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import string
eng_stopwords = stopwords.words('english')
import gc
from bs4 import BeautifulSoup
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
import tensorflow

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder
```

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import normalize, StandardScaler
from scipy import sparse as sp
from scipy.sparse import hstack

from sklearn.metrics import pairwise_distances
from sklearn.preprocessing import minmax_scale
from sklearn.metrics.pairwise import cosine_similarity

from transformers import BertModel, BertTokenizer, BertForSequenceClassificatio
n
from torch.utils.data import Dataset, DataLoader
import torch
```

Workflow

1. Problem Statement and Dataset Description
2. Dataset Loading
3. Machine learning Formulation
4. Exploratory Data Analyses
5. Preprocessing
6. Feature engineering and Modeling (Finding Similarity)
7. Results and Conclusion

. . . . . .

1. Problem Statement

Given two paragraphs, quantify the degree of similarity between the two text-based on Semantic similarity. Semantic Textual Similarity (STS) assesses the degree to which two sentences are semantically equivalent to each other. The STS task is motivated by the observation that accurately modelling the meaning similarity of sentences is a foundational language understanding problem relevant to numerous applications including machine translation (MT), summarization, generation, question-answering (QA), short answer grading, semantic search.

STS is the assessment of pairs of sentences according to their degree of semantic similarity. The task involves producing real-valued similarity scores for sentence pairs.

About Dataset

- The data contains a pair of paragraphs. These text paragraphs are randomly sampled from a raw dataset.
- Each pair of the sentence may or may not be semantically similar. The candidate is to predict a value between 0-1 indicating a degree of similarity between the pair of text paras.
- 1 means highly similar
- 0 means highly dissimilar

Note: The given dataset does not contain any label.

. . . . . .

2. Dataset Loading

```python
# Loading Dataset
data = pd.read_csv("Text_Similarity_Dataset.csv",delimiter=',')
data.sample(10)
```

	Unique_ID	text1	text2
2088	2088	china had role in yukos split-up china lent ru...	yahoo celebrates a decade online yahoo one of...
2582	2582	school tribute for tv host carson more than 1 ...	plaid mp s cottage arson claim a plaid cymru m...
18	18	mobile multimedia slow to catch on there is no...	text messages aid disaster recovery text messa...
3878	3878	wenger steps up row arsene wenger has stepped ...	bellamy under new fire newcastle boss graeme s...
1586	1586	sundance to honour foreign films international...	paraguay novel wins us book prize a novel set ...
2276	2276	wada will appeal against ruling the world anti...	uk athletics agrees new kit deal uk athletics ...
558	558	iranian mps threaten mobile deal turkey s bigg...	rank set to sell off film unit leisure group...
3685	3685	fears raised over ballet future fewer children...	consumer concern over rfid tags consumers are ...
1267	1267	tv calls after carroll error spurs boss martin...	parmalat sues 45 banks over crash parmalat has...
3214	3214	the year search became personal the odds are t...	kennedy begins pre-election tour liberal democ...

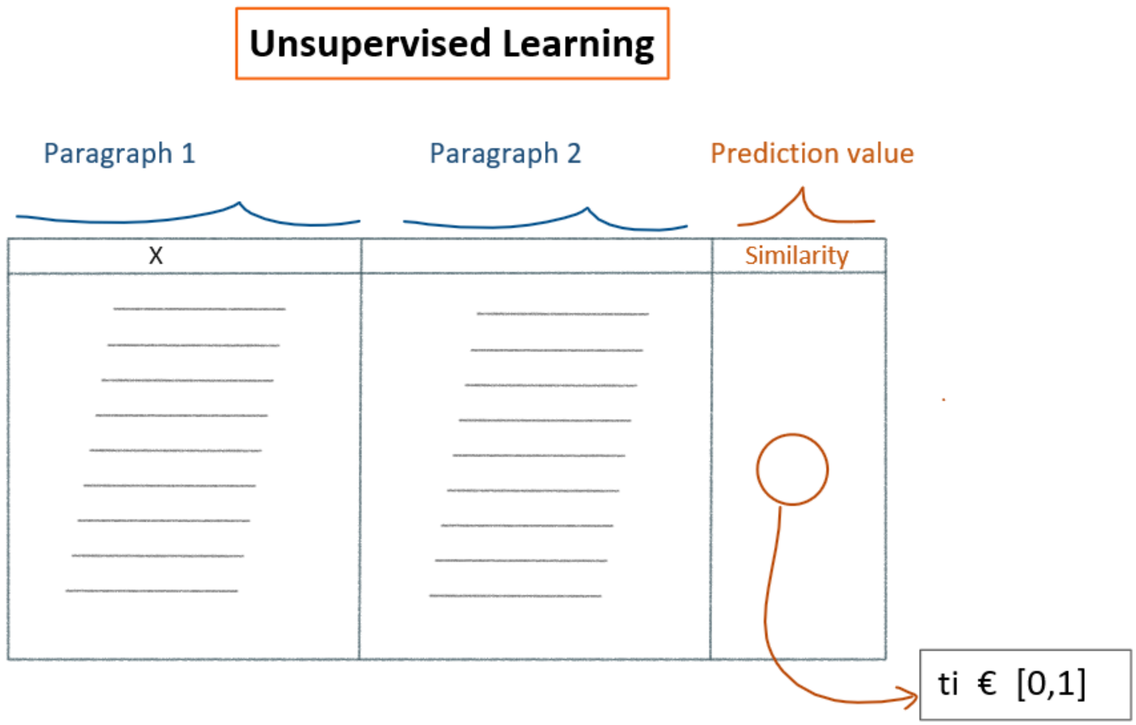
. . . . . .

3. Machine learning Formulation

- The given dataset does not contain any label. Therefore, can be treated as an unsupervised learning problem.
- However, this does not imply that supervised techniques are not applicable.

So the task is to find the degree of similarity between the pair of text paras i.e predict a value between 0-1.

```python
from IPython.display import Image
Image(filename='Capture.png')
```



3.1 Approaches to solve the Problem

Approach 1: Using Avg Word2Vec of text1 and text2 and find similarity

1. Find the word embeddings of each word and average it on total words of text.
2. Find Cosine Similarity between embeddings of text1 and text2.
3. Scale the Cosine Similarity result between [0,1]

Approach 2: Using Avg Tfidf Word2Vec of text1 and text2 and find similarity

1. Fit TfIDF on combination of text1 and text2 and create dictionary of vocabulary and idf_scores
2. Find the word embeddings of each word and multiply the embedding of that word with its corresponding idf_score and average the embeddings of total words present in text.
3. Following above step find the embeddings of text1 and text2.
4. Find Cosine Similarity between embeddings of text1 and text2.
5. Scale the Cosine Similarity result between [0,1]

Approach 3:Using Spacy NLP pipeline to find similarity

1. Find the Similarity between embeddings of text1 and text2 using spacy NLP pipeline.
2. Scale the Similarity result between [0,1]

Approach 4: Using Bert pretrained model as feature extraction (pytorch framework GPU)

1. Format the text into desired format of Bert model.
2. Find the word 784 dim embeddings for each token and from the embedding of [cls] token.
3. Average the embeddings of total tokens passed to bert model.
4. Following above step find the embeddings of text1 and text2.
5. Find Cosine Similarity between embeddings of text1 and text2.
6. Scale the Cosine Similarity result between [0,1]

4. Exploratory Data Analyses

```
# Shape of Dataset
print(f"Shape of dataset: {data.shape}\n")

# Any Unique Values
print(f'Number of Duplicate values: {data.Unique_ID.duplicated().sum()}\n')

print(f" Any null Column{data.isnull().any()}")
```

```
Shape of dataset: (4023, 3)

Number of Duplicate values: 0

 Any null ColumnUnique_ID    False
text1        False
text2        False
dtype: bool
```

4.0 Utility Function

```
# Utility function to plot lineplot and distplot using seaborn
def plot_sns(data,feature,color='lightblue',title=None,subtitle=None):

    """
    Utility function to plot lineplot and distplot using seaborn

    plot_sns(data,feature,color='lightblue',title=None,subtitle=None):

    data = data
    feature = coulum name
    color = color of plot
    title = Either 'length' or 'number' based on which to plot. Otherwise by de
fault='None'
    subtitle = Either 'train_df' or 'test_df'. Otherwise by default='None'

    """
    f, (ax1, ax2) = plt.subplots(1, 2, figsize=(24, 6))

    # line plot
    sns.lineplot(np.arange(len(data)),data,ax=ax1,color=color)
    if title=='number':
        ax1.set(xlabel=f"Idx of {feature}", ylabel=f"Number of words in {featur
e}", title=f'Number of words in {feature} in {subtitle}\n')
    elif title=='length':
        ax1.set(xlabel=f"Idx of {feature}", ylabel=f"Length of {feature}", titl
e=f'Length of {feature} in {subtitle}\n')
    ax1.grid()

    # distribution plot
    sns.distplot(data,ax=ax2,color=color)
    if title=='number':
        ax2.set(xlabel=f"Idx of {feature}", ylabel=f"Number of words in {featur
e}", title=f'Number of words in {feature} in {subtitle}\n')
```

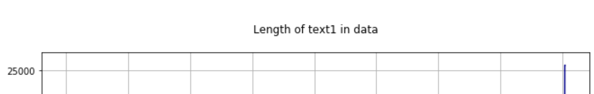
```
    elif title=='length':
        ax2.set(xlabel=f"Idx of {feature}", ylabel=f"Length of {feature}", titl
e=f'Length of {feature} in {subtitle}\n')
    ax2.grid()
    plt.show()

#==============================================================================
==============================================================================
==========================
# Utility function to plot bar graph for both train and test using seaborn
def plot_bar(train_data,test_data,feature=None,x_label=None, y_label=None):

    f, (ax1, ax2) = plt.subplots(1, 2, figsize=(24, 6))

    # for train_df
    sns.barplot(train_data,np.arange(len(train_data)),ax=ax1)
    ax1.set(xlabel=f"{x_label}", ylabel=f"{y_label} {feature}", title='train_df
\n')
    ax1.grid()

    # for test_df
    sns.barplot(test_data,np.arange(len(test_data)),ax=ax2)
    ax2.set(xlabel=f"{x_label}", ylabel=f"{y_label} {feature}", title='test_df
\n')
    ax2.grid()
    plt.show()

#==============================================================================
==============================================================================
==========================
# Utility function to plot requency of most popular words
def word_frequency_plot(dataframe, title=None):
    list_of_all_words = []
    for sent in dataframe:
        list_of_all_words.extend(sent.split())

    top_50_words = pd.Series(list_of_all_words).value_counts()[:50]
    top_50_words_prob_dist = top_50_words.values/sum(top_50_words.values)

    #  plot of frequency of polpular words in train
    plt.figure(figsize=(16,7))
    sns.barplot(top_50_words.index, top_50_words_prob_dist)
    plt.xlabel("words")
    plt.ylabel("frequency")
    plt.title(f"Frequency of most popular words {title}\n")
    plt.xticks(rotation=70)
    plt.grid()
    plt.show()

#==============================================================================
==============================================================================
==========================
# Utility function to check if feature or variable follows Normal distribution
 using Q-Q Plot
def q_q_plot(train_data, test_data, feature_name=None):
```

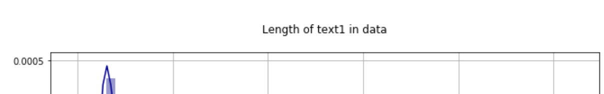
```
    """
    # code refer: https://stackoverflow.com/a/13865874
    """
    f, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 5))

    measurements = train_data
    stats.probplot(measurements, dist="norm", plot=ax1)
    ax1.set(title=f'train : Q-Q Plot for {feature_name} \n')

    measurements = test_data
    stats.probplot(measurements, dist="norm", plot=ax2)
    ax2.set(title=f'test : Q-Q Plot for {feature_name} \n')
    plt.show()

#===============================================================================
================================================================================
==========================
# Utility function for box plot
def box_plot(data, feature_name=None):

    # for train data
    plt.figure(figsize=(26,4))
    sns.violinplot(data,color='darkred')
    plt.title(f'Train : violinplot Plot for {feature_name} \n')
    plt.xlabel(f"{feature_name}")
    plt.ylabel(f"Distribution")
    plt.grid()
    plt.show()
```

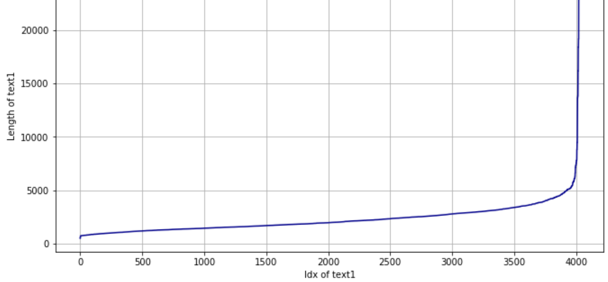
4.1. EDA: text1

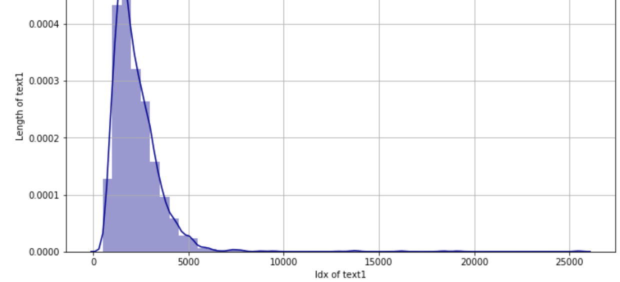
4.1.1. Length of text1

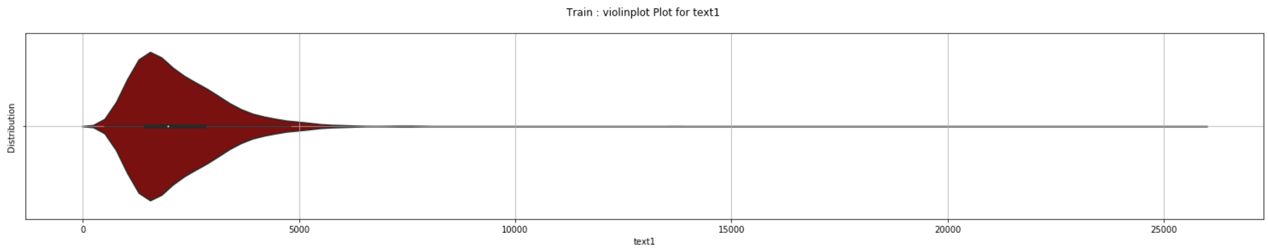
```
# Length of text1
data['len_text1'] = data['text1'].apply(lambda x: len(x))

# plot
plot_sns(sorted(data['len_text1']),"text1",color='darkblue',title='length',subt
itle='data')

# Box plot of Length of question_title in train and test
box_plot(sorted(data['len_text1']), "text1")
```









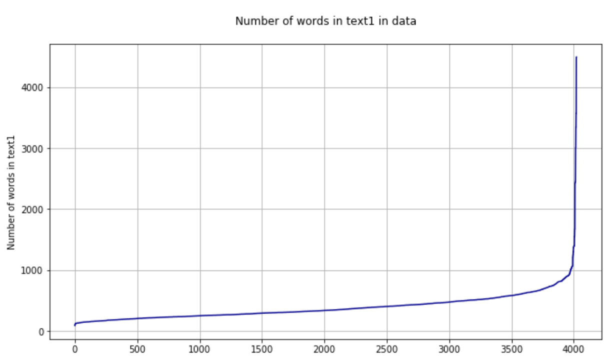


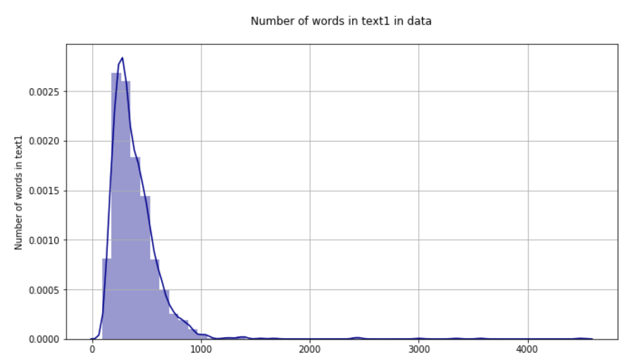
Observation:

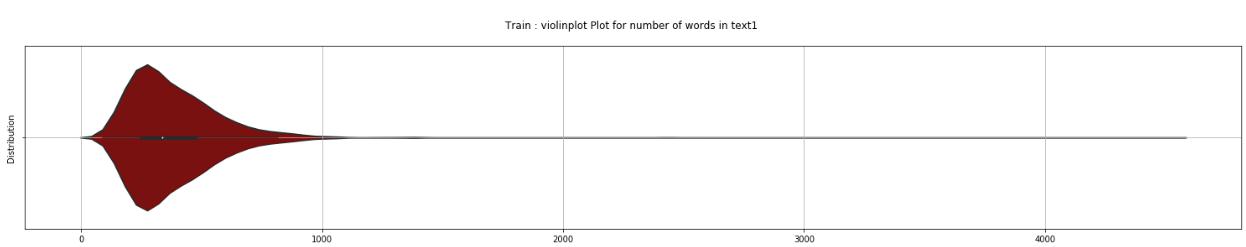
- Maximum length of text1 is upto 25000.
- Distribution is highly skewed toward right(looks like it following log normal distribution)

4.1.2. Number of words in text 1

```python
# Number of words of text1
data['n_text1'] = data['text1'].apply(lambda x: len(x.split()))

# plot
plot_sns(sorted(data['n_text1']),"text1",color='darkblue',title='number',subtit
le='data')

# Box plot of Length of question_title in train and test
box_plot(sorted(data['n_text1']), "number of words in text1")
```







Observation:

- Maximum number of words in text1 is upto 4500 words.
- Distribution is highly skewed toward right.

4.1.3. WordCloud of text1

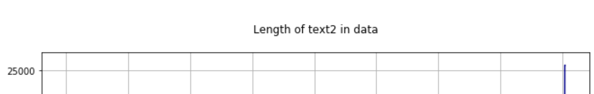
```
# refer: https://www.datacamp.com/community/tutorials/wordcloud-python

# For train_df
text_train = " ".join(word for word in data['text1'])

# Create and generate a word cloud image:
wordcloud = WordCloud().generate(text_train)

# Display the generated image:
plt.figure(figsize=(9,6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title("WordCloud of text1 \n")
plt.axis("off")
plt.show()
```

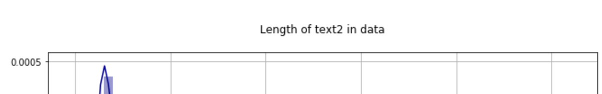
WordCloud of text1

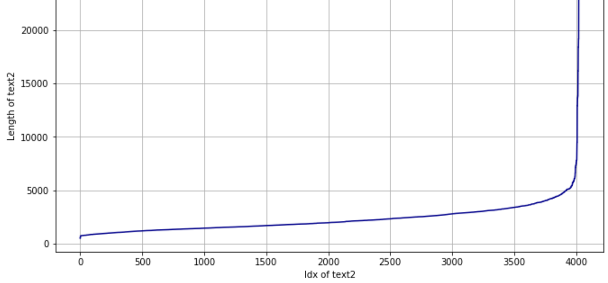
4.2. EDA: text2

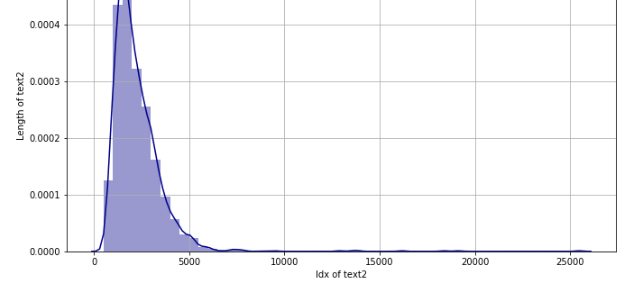
4.2.1. Length of text2

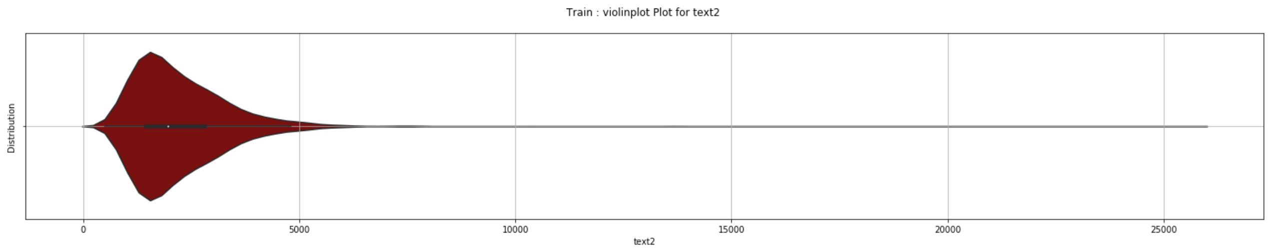
```
# Length of text1
data['len_text2'] = data['text2'].apply(lambda x: len(x))

# plot
plot_sns(sorted(data['len_text2']),"text2",color='darkblue',title='length',subt
itle='data')

# Box plot of Length of question_title in train and test
box_plot(sorted(data['len_text2']), "text2")
```





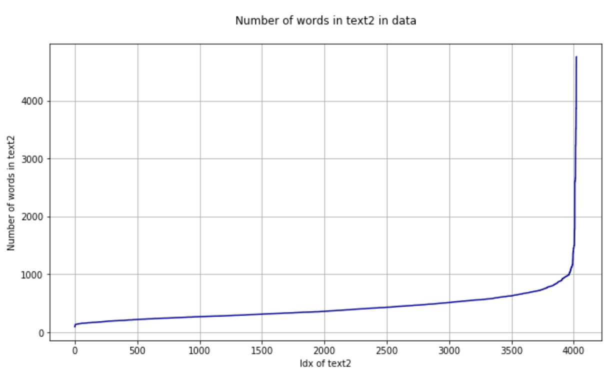


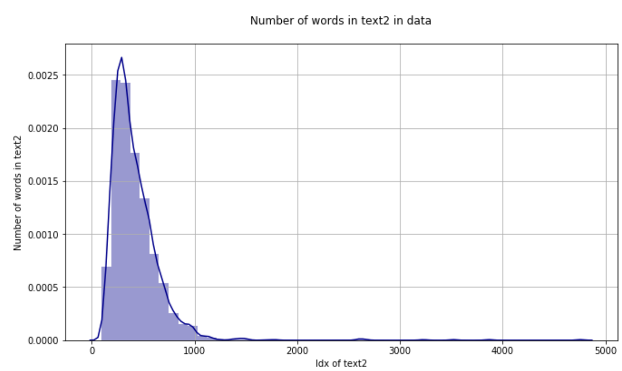


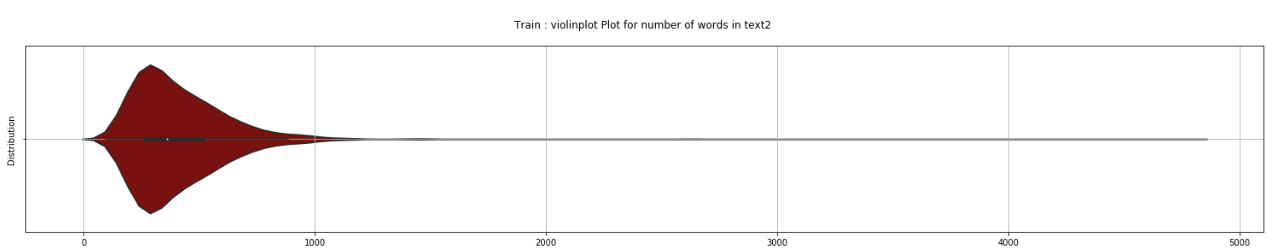


4.2.2. Number of words in text2

```
# Number of words of text1
data['n_text2'] = data['text2'].apply(lambda x: len(x.split(" ")))

# plot
plot_sns(sorted(data['n_text2']),"text2",color='darkblue',title='number',subtit
le='data')

# Box plot of Length of question_title in train and test
box_plot(sorted(data['n_text2']), "number of words in text2")
```







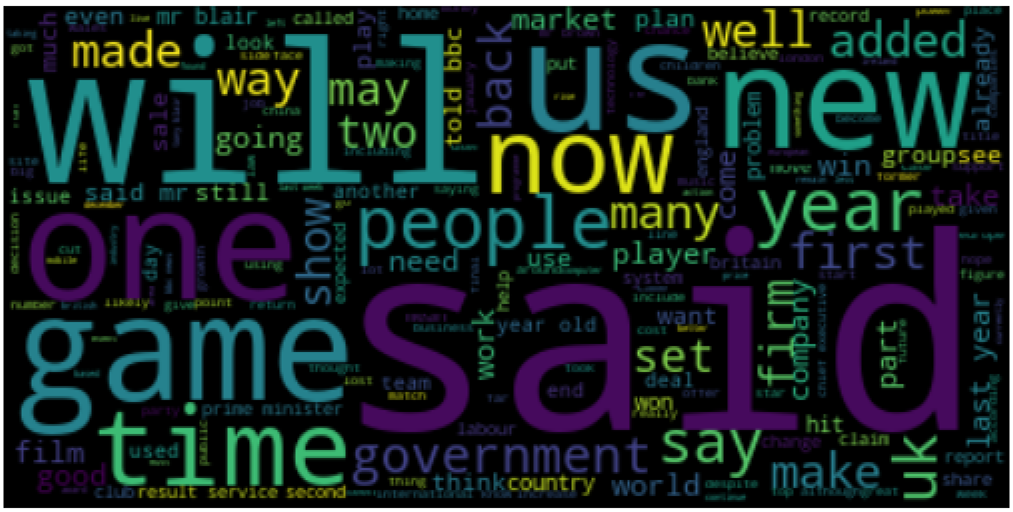
Observation:

Observation is almost same as text1 observations.

4.2.3. WordCloud of text2

```
# refer: https://www.datacamp.com/community/tutorials/wordcloud-python

# For train_df
text_train = " ".join(word for word in data['text2'])

# Create and generate a word cloud image:
wordcloud = WordCloud().generate(text_train)

# Display the generated image:
plt.figure(figsize=(9,6))
plt.imshow(wordcloud, interpolation='bilinear')
plt.title("WordCloud of text2 \n")
plt.axis("off")
plt.show()
```

WordCloud of text2



5. Preprocessing

Utility Function

```
# Preprocessing Functions
# credit : https://www.kaggle.com/urvishp80/quest-encoding-ensemble

mispell_dict = {"aren't" : "are not","can't" : "cannot","couldn't" : "could no
t","couldnt" : "could not","didn't" : "did not","doesn't" : "does not",
                "doesnt" : "does not","don't" : "do not","hadn't" : "had not",
"hasn't" : "has not","haven't" : "have not","havent" : "have not",
                "he'd" : "he would","he'll" : "he will","he's" : "he is","i'd"
: "I would","i'd" : "I had","i'll" : "I will","i'm" : "I am",
                "isn't" : "is not","it's" : "it is","it'll":"it will","i've" :
"I have","let's" : "let us","mightn't" : "might not",
                "mustn't" : "must not","shan't" : "shall not","she'd" : "she wo
uld","she'll" : "she will","she's" : "she is","shouldn't" : "should not",
                "shouldnt" : "should not","that's" : "that is","thats" : "that
 is","there's" : "there is","theres" : "there is","they'd" : "they would",
                "they'll" : "they will","they're" : "they are","theyre":  "they
are","they've" : "they have","we'd" : "we would","we're" : "we are",
                "weren't" : "were not","we've" : "we have","what'll" : "what wi
```

```
ll","what're" : "what are","what's" : "what is","what've" : "what have",
                "where's" : "where is","who'd" : "who would","who'll" : "who wi
ll","who're" : "who are","who's" : "who is","who've" : "who have",
                "won't" : "will not","wouldn't" : "would not","you'd" : "you wo
uld","you'll" : "you will","you're" : "you are","you've" : "you have",
                "'re": " are","wasn't": "was not","we'll":" will","didn't": "di
d not","tryin'":"trying"}


imp_keywords = ("R", "r", "C","c", "os", "OS")


# Counting the numeric feature and removing it
def count_digits_and_remove(text):
    """
    counting the number of occurance of digit

    return : text , (n_total_digit, n_2_digit, n_3_digit, n_4_digit, n_5_plus_d
igit)

    n_total_digit  : Total occurance of numeric feature
    n_2_digit      : Number of time 2 digit numeric feature occur
    n_2_digit      : Number of time 3 digit numeric feature occur
    n_2_digit      : Number of time 4 digit numeric feature occur
    n_5_plus_digit : Number of time more than 4 digit numeric feature occur

    """
    digits = re.findall(r'[0-9]+',text)

    n_total_digit = []
    n_2_digit = []
    n_3_digit = []
    n_4_digit = []
    n_5_plus_digit = []

    n_total_digit.append(len(digits))
    for digit in digits:
        if len(digit)==2:
            n_2_digit.append(digit)

        elif len(digit)==3:
            n_3_digit.append(digit)

        elif len(digit)==4:
            n_4_digit.append(digit)

        elif len(digit)>4:
            n_5_plus_digit.append(digit)

    # remove all the numbers
    text = re.sub(r'[0-9]+'," ",text)

    return text , (len(n_total_digit), len(n_2_digit), len(n_3_digit), len(n_4_
digit), len(n_5_plus_digit))
```

```python
#==============================================================================
========================================================
# Counting number of non_alpha_numeric character and removing all the special c
haracter words
def count_non_alpha_numeric_and_remove(text):
    """ Counting number of non_alpha_numeric character(for programinng context)
"""

    # finding all the all the non_alpha_numeric char
    n_special_char = (re.findall(r"[^A-Za-z0-9 :]", text))

    # removing it.
    text = re.sub(r"[^A-Za-z0-9]", " ",text)

    return text ,len(n_special_char)

#==============================================================================
========================================================
# Counting the number of all capital word(maybe it would be corelated with labe
ls) and coverting into lower string character
def count_all_cap_words_and_lower_it(text):

    """Finding the number of all capital word and lower it"""

    # Find all the capital words
    n_all_capital_words = (re.findall(pattern = r'([A-Z]([A-Z])+)',string=text
))

    # converting into string into lower char string
    text = text.lower()

    return text , len(n_all_capital_words)

#==============================================================================
========================================================
def _get_mispell(mispell_dict):
    mispell_re = re.compile('(%s)' % '|'.join(mispell_dict.keys()))
    return mispell_dict, mispell_re

def replace_typical_misspell(text):

    """De-Concatenation of words and correction of misspelled words"""
    mispellings, mispellings_re = _get_mispell(mispell_dict)

    def replace(match):
        return mispellings[match.group(0)]

    return mispellings_re.sub(replace, text)

#==============================================================================
========================================================

# Return the number of links and text without html tags
# Also return the counts of 'number of lines'  and remove it
def strip_html(text):
```

```
    """
    Return the number of links and clean text (without html tags)
    Also return the counts of 'number of lines'  and remove it

    """
    # finding http links using regex and counting it and remove it
    n_links = (re.findall(r'http[s]?://\S+',text))
    text = re.sub(r'http[s]?://\S+'," ",text)

    # finding number of lines using regex and counting it and remove it
    n_lines = re.findall(r'\n',text)
    text = re.sub(r'\n', " ",text)

    return  text, len(n_links) , len(n_lines)

#===============================================================================
========================================================
# function to remove all the stopwords and words having lengths less than 3
def remove_stop_words_and_punc(text) :

    """
    Remove all the stopwords
    Remove all the words whose length is less than 3 and not belong to importan
t keywords(e.g. 'C','R','OS' etc)

    """
    # removing the words from the stop words list: 'no', 'nor', 'not'
    stops = set(stopwords.words("english"))
    stops.remove('no')
    stops.remove('nor')
    stops.remove('not')

    text= text.split()
    text = [w for w in text if not w in stops]

    # Removing the words having length less than 3 and not the imp_keyword
    clean_text = []
    for word in text:
        if word not in imp_keywords and len(word)<3:
            pass
        else:
            clean_text.append(word)


    clean_text = " ".join(clean_text)
    return(clean_text)

#===============================================================================
========================================================
# function for stemming of words in text
def stem(text):
    stemmer = PorterStemmer()
    result = " ".join([ stemmer.stem(word) for word in text.split(" ")])
    return result
```

```
#=========================================================================================================================
# Final text cleaning funtion
def clean_text(text, extra_features=True, strip_html_fn=True, count_all_cap_words_and_lower_it_fn=True, replace_typical_misspell_fn=True, count_digits_and_remove_fn=True,
               count_non_alpha_numeric_and_remove_fn=True, remove_stop_words_and_punc_fn=True,stem_fn=True):
    """
    This function sequentially execute all the cleaning and preprocessing function and finaly gives cleaned text.
    Input: Boolean values of extra_features, strip_html, count_all_cap_words_and_lower_it, replace_typical_misspell, count_non_alpha_numeric_and_remove, remove_stop_words_and_punc, stem
            (by default all the input values = True)

    return: clean text

    """
    if strip_html_fn:
        # remove html tags
        clean_text, n_links, n_lines = strip_html(text)

    if count_all_cap_words_and_lower_it_fn:
        # Find all the capital words and covert all chars of text into lower char string
        clean_text, n_all_capital_words= count_all_cap_words_and_lower_it(clean_text)

    if replace_typical_misspell_fn:
        # de-concatenation of words
        clean_text = replace_typical_misspell(clean_text)

    if count_digits_and_remove_fn:
        # count the numbers and remove it
        clean_text, (n_total_digit, n_2_digit, n_3_digit, n_4_digit, n_5_plus_digit) = count_digits_and_remove(clean_text)

    if count_non_alpha_numeric_and_remove_fn:
        # Count the number of non alpha numeric character and remove it
        clean_text, n_non_alpha_char = count_non_alpha_numeric_and_remove(clean_text)

    if remove_stop_words_and_punc_fn:
        # removing Stopwords and the words length less than 3(As these words mostly tend to redundant words) excpect 'C' and 'R'and 'OS' <-- programing keywords
        clean_text = remove_stop_words_and_punc(clean_text)

    if stem_fn:
        # stemming ( use only for BOW or TFIDF represention. Not effective for word embedding like w2v or glove)
        clean_text = stem(clean_text)

    # return extra_features
```

```
    if extra_features:
        return clean_text, (n_links, n_lines, n_all_capital_words, n_non_alpha_
char, n_total_digit, n_2_digit, n_3_digit, n_4_digit, n_5_plus_digit)

    else:
        return clean_text

#================================================================================
==========================================================================
#================================================================================
==========================================================================
# This function is only for  word embedding pre processing
# This function will take dataframe and return cleaned dataframe (This fincion
 will only be used for BOW nd TFIDF representaion)
def clean_data_for_embeding(dataframe ):

    """
    This function will take dataframe and return cleaned dataframe along with e
xtra features.

    Input: dataframe which need to preprocess only for embeddings words
    Return: clean dataframe

    """
    cleaned_data = []

    for i in tqdm(range(dataframe.shape[0])):

        text = dataframe.iloc[i]

        cleaned_text = clean_text(text,extra_features= False, stem_fn=False) #
 There is no need of extra_features to calculate again as it is already calcula
ted
        cleaned_data.append(cleaned_text)

    return cleaned_data
```

5.0. Preprocessing for word embedding (without stemming)

Preprocessing utility function for word embedding (without stemming)

Only minor change in this function and above preprocessing function is this statement:

> "clean_text(text,extra_features= False, stem_fn=False)" at l
> ine 19

5.1. Preprocessing: text1

```
clean_text_1_for_embedding = clean_data_for_embeding(data['text1'])
data['clean_text1_for_embedding'] = clean_text_1_for_embedding
```

```
100%|██████████████████████████████████████████████████████████████████████| 4023/4023

[00:02<00:00, 1436.51it/s]
```

Sample

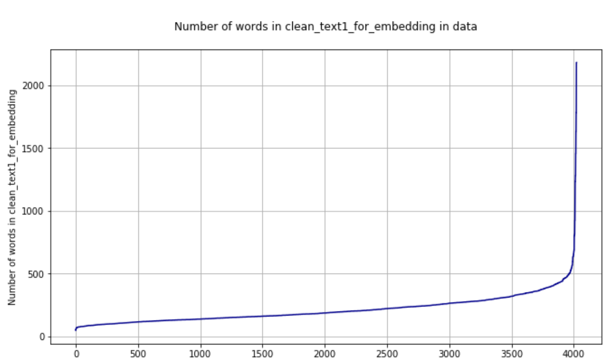
```
# Sample preproceesing
i=15

print(f"Before preproceesing: \n{'-'*20}\n{data['text1'][i]}\n")
print(f"After preproceesing: \n{'-'*20}\n{data['clean_text1_for_embedding'][i]}
")
```

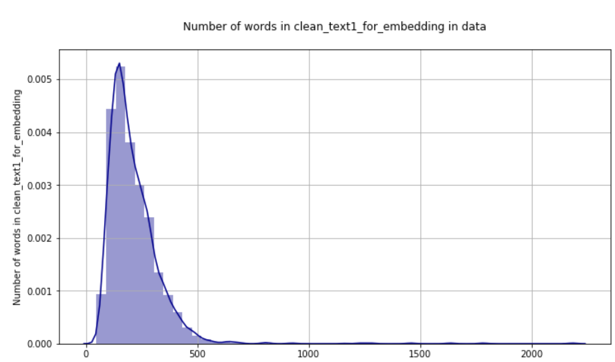
```
Before preproceesing:
--------------------
bond game fails to shake or stir for gaming fans  the word goldeneye evokes excited memories
not only of the james bond revival flick of 1995  but also the classic shoot-em-up that acco
mpanied it and left n64 owners glued to their consoles for many an hour.  adopting that hall
owed title somewhat backfires on this new game  for it fails to deliver on the promise of it
s name and struggles to generate the original s massive sense of fun. this however is not a
sequel  nor does it relate to the goldeneye film. you are the eponymous renegade spy  an age
nt who deserts to the bond world s extensive ranks of criminal masterminds  after being deem
ed too brutal for mi6. your new commander-in-chief is the portly auric goldfinger  last seen
in 1964  but happily running around bent on world domination. with a determination to justif
y its name which is even less convincing than that of tina turner s similarly-titled theme s
ong  the game literally gives the player a golden eye following an injury  which enables a d
egree of x-ray vision.  rogue agent signals its intentions by featuring james bond initially
and proceeding to kill him off within moments  squashed by a plummeting helicopter. the noti
on is of course to add a novel dark edge to a 007 game  but the premise simply does not get
the juices flowing like it needs to.  recent bond games like nightfire and everything or not
hing were very competent and did a fine job of capturing the sense of flair  invention and g
lamour of the film franchise. this title lacks that aura  and when the bond magic shines thr
ough  it feels like a lucky accident. the central problem is that the gameplay just is not g
ood enough. quite aside from the bizarre inability to jump  the even more bizarre glaring gr
aphical bugs and dubious enemy ai  the levels simply are not put together with much style or
imagination. admittedly the competition has been tough  even in recent weeks  with the likes
of halo 2 and half life 2 triumphing in virtually every department. what the game is good at
is enveloping you in noisy  dynamic scenes of violent chaos. as is the trend of late  you ar
e made to feel like you are in the midst of a really messy and fraught encounter. sadly that
sense of action is outweighed by the difficulty of navigating and battling within the chaos
meaning that frustration is often the outcome. and irregular save points mean you have to ba
cktrack each time you are killed. a minute red dot passes for a crosshair  although the coll
ision-detection is so suspect that the difficulties of aiming weapons are compensated for. s
hooting enemies from a distance can be tricky  and you will not always know you have picked
them off  since dead enemies vanish literally before they have fully hit the floor  and they
do so in some woefully uninspiring death animations. it is perhaps indicative of a lack of c
onfidence that the game maker s allow you several different weapons almost immediately and t
hrow you quickly into raging firefights - no time is risked with a measured build-up.  by fa
r the most satisfying element of the game is seeing old favourites like dr no  goldfinger  h
at-fiend oddjob and crazed russian sex beast xenia onatopp resurrected after all these years
and with their faces rendered in an impressively recognisable fashion.  there is a real thri
ll from doing battle with these legendary villains  and it is a testament to the power of th
e bond universe that they can cut such a dash. but the in-game niggles  combined with a stor
y and presentation that just do not feel sufficiently well thought-through  will make this a
disappointment for most. diehard fans of bond will probably find enough here to make it a wo
rthwhile purchase and try to ignore the failings. the game is weak  not completely unplayabl
e. then again  007 fanatics may also take umbrage at the cavalier blending of characters fro
m different eras. given james bond s healthy pedigree in past games  there is every reason t
o hope that this is just a blip  a commendable idea that just has not worked  that will be r
ectified when the character inevitably makes his return.  goldeneye: rogue agent is out now

After preproceesing:
--------------------
bond game fails shake stir gaming fans word goldeneye evokes excited memories not james bond
```

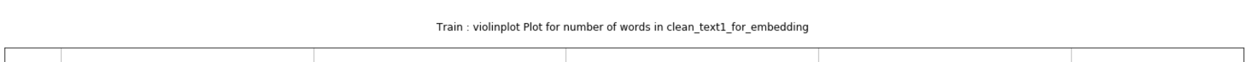
```
revival flick also classic shoot accompanied left owners glued consoles many hour adopting h
allowed title somewhat backfires new game fails deliver promise name struggles generate orig
inal massive sense fun however not sequel nor relate goldeneye film eponymous renegade spy a
gent deserts bond world extensive ranks criminal masterminds deemed brutal new commander chi
ef portly auric goldfinger last seen happily running around bent world domination determinat
ion justify name even less convincing tina turner similarly titled theme song game literally
gives player golden eye following injury enables degree ray vision rogue agent signals inten
tions featuring james bond initially proceeding kill within moments squashed plummeting heli
copter notion course add novel dark edge game premise simply not get juices flowing like nee
ds recent bond games like nightfire everything nothing competent fine job capturing sense fl
air invention glamour film franchise title lacks aura bond magic shines feels like lucky acc
ident central problem gameplay not good enough quite aside bizarre inability jump even bizar
re glaring graphical bugs dubious enemy levels simply not put together much style imaginatio
n admittedly competition tough even recent weeks likes halo half life triumphing virtually e
very department game good enveloping noisy dynamic scenes violent chaos trend late made feel
like midst really messy fraught encounter sadly sense action outweighed difficulty navigatin
g battling within chaos meaning frustration often outcome irregular save points mean backtra
ck time killed minute red dot passes crosshair although collision detection suspect difficul
ties aiming weapons compensated shooting enemies distance tricky not always know picked sinc
e dead enemies vanish literally fully hit floor woefully uninspiring death animations perhap
s indicative lack confidence game maker allow several different weapons almost immediately t
hrow quickly raging firefights time risked measured build far satisfying element game seeing
old favourites like goldfinger hat fiend oddjob crazed russian sex beast xenia onatopp resur
rected years faces rendered impressively recognisable fashion real thrill battle legendary v
illains testament power bond universe cut dash game niggles combined story presentation not
feel sufficiently well thought make disappointment diehard fans bond probably find enough ma
ke worthwhile purchase try ignore failings game weak not completely unplayable fanatics may
also take umbrage cavalier blending characters different eras given james bond healthy pedig
ree past games every reason hope blip commendable idea not worked rectified character inevit
ably makes return goldeneye rogue agent
```

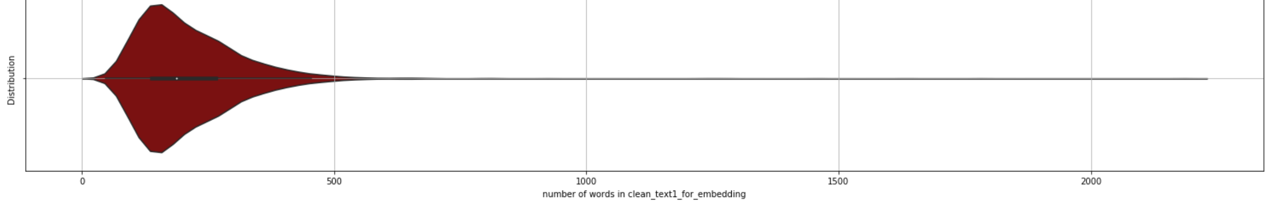
5.1.1. Number of words in clean_text1_for_embedding

```python
# Number of words of text1
data['n_word_clean_text1_for_embedding'] = data['clean_text1_for_embedding'].ap
ply(lambda x: len(x.split()))

# plot
plot_sns(sorted(data['n_word_clean_text1_for_embedding']),"clean_text1_for_embe
dding",color='darkblue',title='number',subtitle='data')

# Box plot of Length of question_title in train and test
box_plot(sorted(data['n_word_clean_text1_for_embedding']), "number of words in
 clean_text1_for_embedding")
```





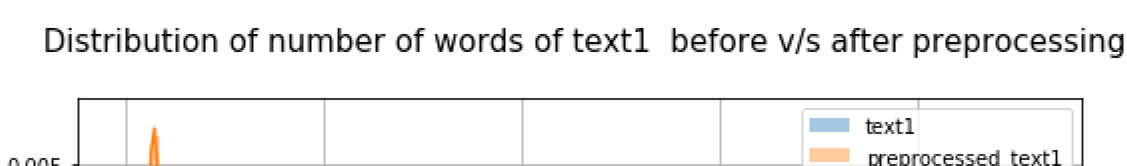


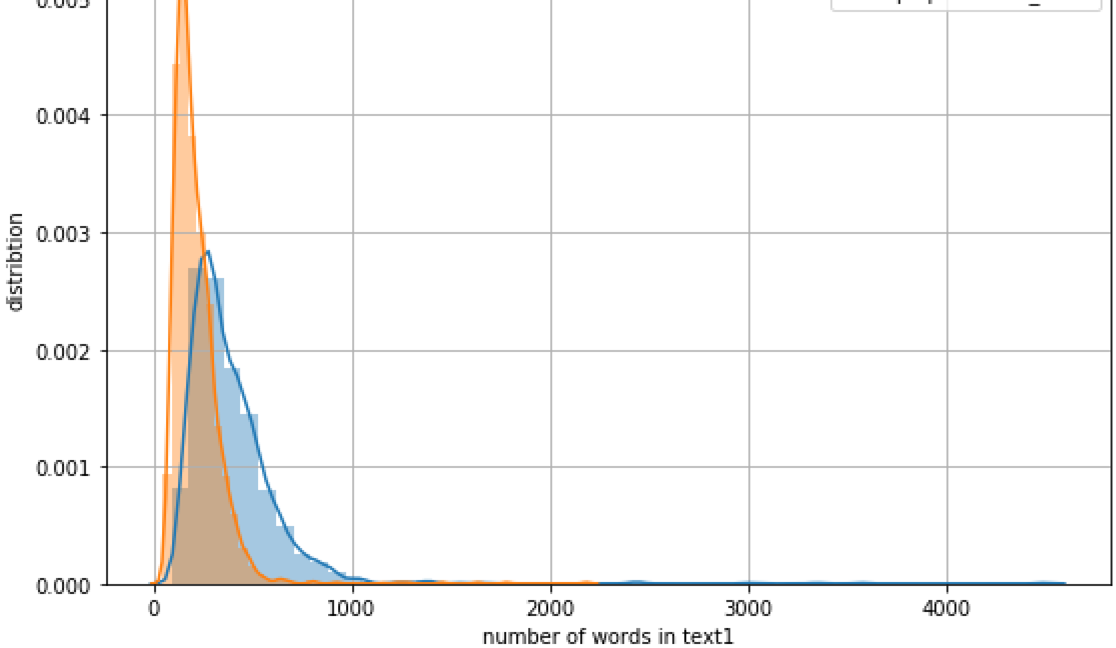


5.1.2. Distribution of number of words of text1 text before v/s after preprocessing

```
text_features_column = ['text1']

for idx,column in enumerate(text_features_column):

    # Calculating the length of text before and after preprocessing
    len_after_cleaning = data[f'clean_{column}_for_embedding'].apply(lambda x : 
len(x.split()))
    len_before_cleaning = data[f'{column}'].apply(lambda x : len(x.split()))

    # ploting
    print( f"{idx+1}: Plot for {column}")
    plt.figure(figsize=(9,6))
    sns.distplot(len_before_cleaning, label=f'{column}')
    sns.distplot(len_after_cleaning, label=f'preprocessed_{column}')
    plt.title(f" Distribution of number of words of {column}  before v/s after
 preprocessing\n",fontsize=15)
    plt.ylabel("distribtion")
    plt.xlabel(f"number of words in {column}")
    plt.legend()
    plt.grid()
    plt.show()
```

```
1: Plot for text1
```





Observation

- Preproceesing has reduced the large amount of number of words making distribution more symmetrical but it is still higly skewed towards right.
- We can see from above disribution that most of the text has less than 500 number of words. (we can use 500 tokens as max_len in BERT Embeddings)

5.2. Preprocessing: text2

```
clean_text_2_for_embedding = clean_data_for_embeding(data['text2'])
data['clean_text2_for_embedding'] = clean_text_2_for_embedding
```

```
100%|██████████████████████████████████████████████████████████████████| 4023/4023
[00:02<00:00, 1357.43it/s]
```

Sample

```
# Sample preproceesing
i=15

print(f"Before preproceesing: \n{'-'*20}\n{data['text2'][i]}")
print(f"\nAfter preproceesing: \n{'-'*20}\n{data['clean_text2_for_embedding']
[i]}")
```

```
Before preproceesing:
--------------------
mobile multimedia slow to catch on there is no doubt that mobile phones sporting cameras and
colour screens are hugely popular. consumers swapping old phones for slinkier  dinkier versi
ons are thought to be responsible for a 26% increase in the number of phones sold during the
third quarter of 2004  according to analysts gartner more than 167 million handsets were sol
d between july and september 2004  a period that  according to gartner analyst carolina mila
nesi is  seldom strong . but although consumers have mobiles that can take and send snaps  s
ounds and video clips few  so far  are taking the chance to do so.  in fact  the numbers of
people not taking and sending pictures  audio and video is growing. figures gathered by cont
inental research shows that 36% of british camera phone users have never sent a multimedia m
essage (mms)  up from 7% in 2003. this is despite the fact that  during the same period  the
numbers of camera phones in the uk more than doubled to 7.5 million. getting mobile phone us
```

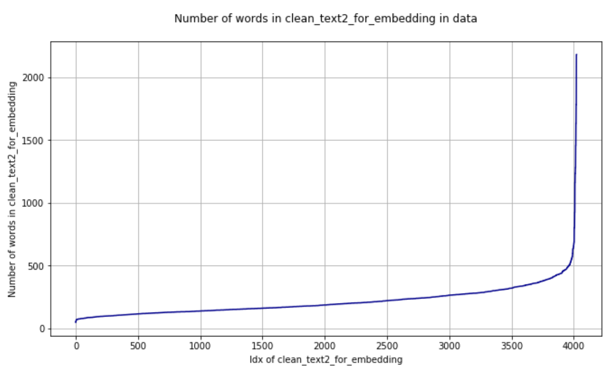
ers to send multimedia messages is really important for operators keen to squeeze more cash out of their customers and offset the cost of subsidising the handsets people are buying. th e problem they face said shailendra jain head of mms firm adamind is educating people in how to send the multimedia messages using their funky handsets.   also   he said   they have to simplify the interface so its not rocket science in terms of someone understanding it. r esearch bears out the suspicion that people are not sending multimedia messages because they do not know how to. according to continental research 29% of the people it questioned said they were technophobes that tended to shy away from innovation. only 11% regarded themselves as technically savvy enough to send a picture or video message. the fact that multimedia ser vices are not interoperable across networks and phones only adds to people s reluctance to s tart sending them said mr jain. they ask themselves: if i m streaming video from one hand set to another will it work   he said. there s a lot of user apprehension about that.   th ere are other deeper technical reasons why multimedia messages are not being pushed as stron gly as they might. andrew bud executive chairman of messaging firm mblox said mobile phone operators cap the number of messages that can be circulating at any one time for fear of ove rwhelming the system. the rate we can send mms into the mobile network is fairly constant he said. the reason for this is that there are finite capacities for data traffic on the sec ond generation networks that currently have the most users. no-one wants to take the risk o f swamping these relatively narrow channels so the number of mms messages is capped said mr bud. this has led to operators finding other technologies particularly one known as wap-pus h to get multimedia to their customers. but when networks do find a good way to get multime dia to their customers the results can be dramatic. israeli technology firm celltick has fo und a way to broadcast data across phone networks in a way that does not overwhelm existing bandwidth. one of the first firms to use the celltick service is hutch india the largest mo bile firm in the country. the broadcast system gets multimedia to customers via a rolling me nu far faster than would be possible with other systems. while not multimedia messaging suc h a system gets people used to seeing their phones as a device that can handle all different types of content. as a result 40% of the subscribers to the hutch alive which uses celltick s broadcast technology regularly click for more pictures sounds and images from the operat or. operators really need to start utilising this tool to reach their customers   said yaro n toren spokesman for celltick. until then multimedia will be a message that is not gettin g through.

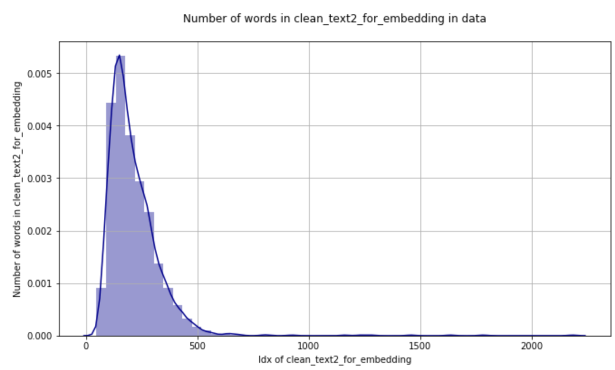
After preproceesing:
----
mobile multimedia slow catch doubt mobile phones sporting cameras colour screens hugely popu lar consumers swapping old phones slinkier dinkier versions thought responsible increase num ber phones sold third quarter according analysts gartner million handsets sold july septembe r period according gartner analyst carolina milanesi seldom strong although consumers mobile s take send snaps sounds video clips far taking chance fact numbers people not taking sendin g pictures audio video growing figures gathered continental research shows british camera ph one users never sent multimedia message mms despite fact period numbers camera phones double d million getting mobile phone users send multimedia messages really important operators kee n squeeze cash customers offset cost subsidising handsets people buying problem face said sh ailendra jain head mms firm adamind educating people send multimedia messages using funky ha ndsets also said simplify interface not rocket science terms someone understanding research bears suspicion people not sending multimedia messages not know according continental resear ch people questioned said technophobes tended shy away innovation regarded technically savvy enough send picture video message fact multimedia services not interoperable across networks phones adds people reluctance start sending said jain ask streaming video one handset anothe r work said lot user apprehension deeper technical reasons multimedia messages not pushed st rongly might andrew bud executive chairman messaging firm mblox said mobile phone operators cap number messages circulating one time fear overwhelming system rate send mms mobile netwo rk fairly constant said reason finite capacities data traffic second generation networks cur rently users one wants take risk swamping relatively narrow channels number mms messages cap ped said bud led operators finding technologies particularly one known wap push get multimed ia customers networks find good way get multimedia customers results dramatic israeli techno logy firm celltick found way broadcast data across phone networks way not overwhelm existing bandwidth one first firms use celltick service hutch india largest mobile firm country broad cast system gets multimedia customers via rolling menu far faster would possible systems not multimedia messaging system gets people used seeing phones device handle different types con tent result subscribers hutch alive uses celltick broadcast technology regularly click pictu

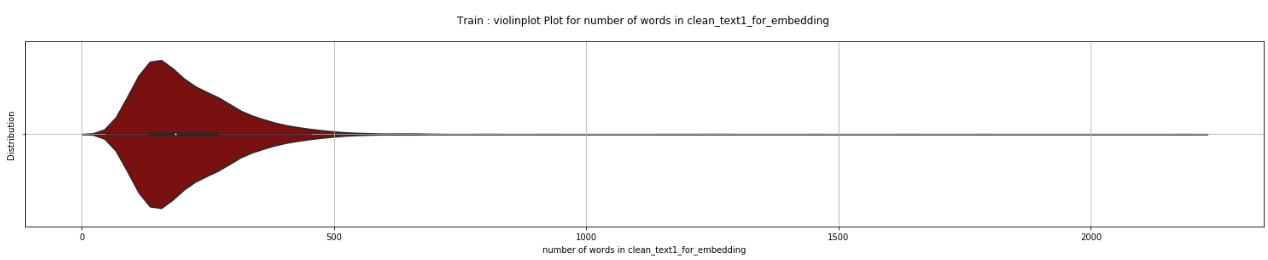
res sounds images operator operators really need start utilising tool reach customers said y
aron toren spokesman celltick multimedia message not getting

5.2.1. Number of words in clean_text2_for_embedding

```
# Number of words of text1
data['n_word_clean_text2_for_embedding'] = data['clean_text2_for_embedding'].ap
ply(lambda x: len(x.split()))

# plot
plot_sns(sorted(data['n_word_clean_text2_for_embedding']),"clean_text2_for_embe
dding",color='darkblue',title='number',subtitle='data')

# Box plot of Length of question_title in train and test
box_plot(sorted(data['n_word_clean_text2_for_embedding']), "number of words in
 clean_text1_for_embedding")
```







Observation

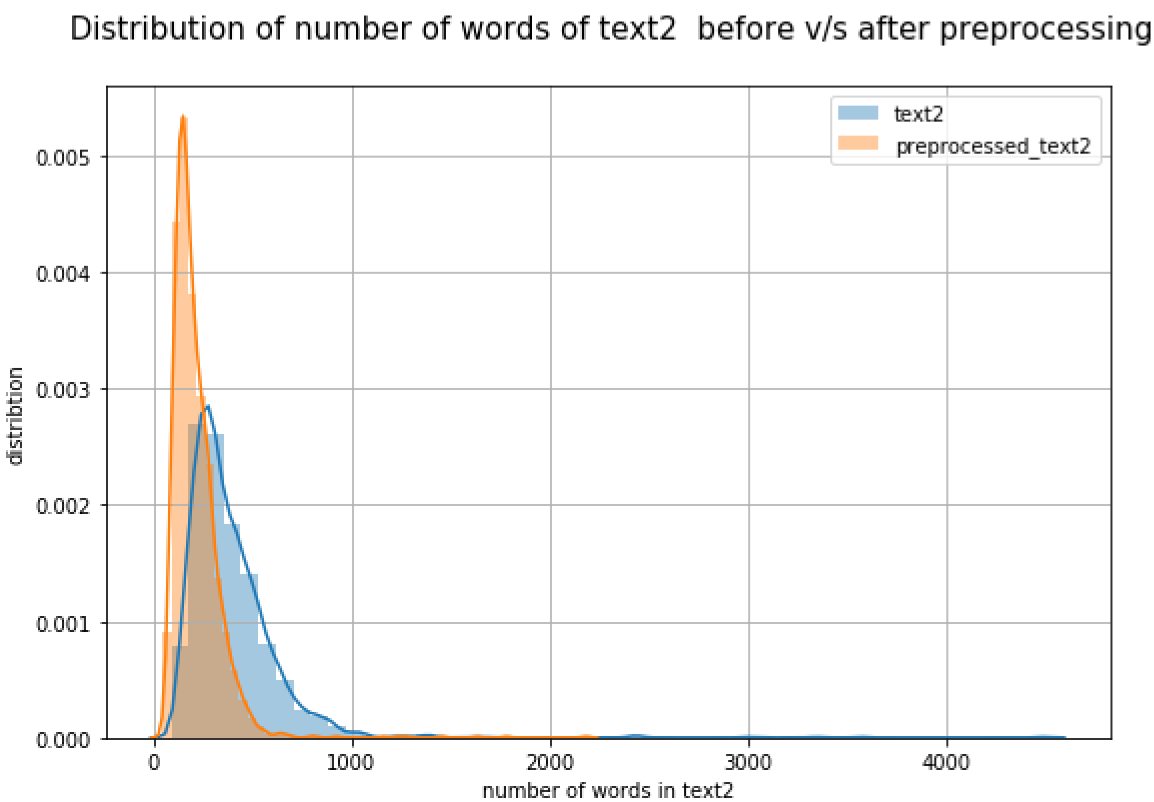
- Preproceesing has reduced the large amount of number of words making distribution more symmetrical but it is still higly skewed towards right.
- We can see from above disribution that most of the text has less than 500 number of words. (we can use 500 tokens as max_len in BERT Embeddings)

5.2.2. Distribution of number of words of text1 text before v/s after preprocessing

```
text_features_column = ['text2']

for idx,column in enumerate(text_features_column):

    # Calculating the length of text before and after preprocessing
    len_after_cleaning = data[f'clean_{column}_for_embedding'].apply(lambda x :
len(x.split()))
    len_before_cleaning = data[f'{column}'].apply(lambda x : len(x.split()))
```

```
    # ploting
    print( f"{idx+1}: Plot for {column}")
    plt.figure(figsize=(9,6))
    sns.distplot(len_before_cleaning, label=f'{column}')
    sns.distplot(len_after_cleaning, label=f'preprocessed_{column}')
    plt.title(f" Distribution of number of words of {column}  before v/s after preprocessing\n",fontsize=15)
    plt.ylabel("distribtion")
    plt.xlabel(f"number of words in {column}")
    plt.legend()
    plt.grid()
    plt.show()
```

```
1: Plot for text2
```



6. Feature Engineering and Modeling (Finding Similarity)

Vector Representation of text

Using count vectoriser and tfidf vectoriser directly make vocab size too large and might not good as well. Therefore I am skipping it.

6.1. W2V Representation

6.2. Tf-idf W2V Representation

6.3. Using Bert Embedding

- Note: For word Embedding we have to preprocess the text again to remove stemming as it impact the sentiment of word in Embedding badly.

6.1. Approach 1: W2V Representation

Refer: https://www.kaggle.com/phoenix9032/quest-preprocessing-data-for-embedding

```
from gensim.models import KeyedVectors

news_path = 'crawl-300d-2M.vec'
embeddings_index = KeyedVectors.load_word2vec_format(news_path, binary=False)
```

```
## Building vocubulary from our Quest Data
def build_vocab(sentences, verbose =  True):
    """
    :param sentences: list of list of words
    :return: dictionary of words and their count
    """
    vocab = {}
    for sentence in tqdm(sentences, disable = (not verbose)):
        for word in sentence:
            try:
                vocab[word] += 1
            except KeyError:
                vocab[word] = 1
    return vocab

#===============================================================================
===========================
import operator
## This is a common function to check coverage between our quest data and the w
ord embedding
def check_coverage(vocab,embeddings_index):
    a = {}
    oov = {}
    k = 0
    i = 0
    for word in tqdm(vocab):
        try:
            a[word] = embeddings_index[word]
            k += vocab[word]
        except:

            oov[word] = vocab[word]
            i += vocab[word]
            pass

    print('Found embeddings for {:.2%} of vocab'.format(len(a) / len(vocab)))
    print('Found embeddings for  {:.2%} of all text'.format(k / (k + i)))
    sorted_x = sorted(oov.items(), key=operator.itemgetter(1))[::-1]

    return sorted_x
```

6.1.1. Check Coverage for clean_text1_for_embedding

```
##Apply the vocab function to get the words and the corresponding counts
```

```
sentences = data["clean_text1_for_embedding"].apply(lambda x: x.split()).values
vocab = build_vocab(sentences)

print(f"\nFor clean_text1_for_embedding: \n{'-'*40}")
oov = check_coverage(vocab,embeddings_index)

## List 10 out of vocabulary word
print(f"\nTop 10 out of vocabulary word: \n{'-'*30}")
oov[:10]
```

```
100%|████████████████████████████████████████████████████████████████| 4023/4023
[00:00<00:00, 31648.56it/s]


For clean_text1_for_embedding:
----------------------------------------


100%|██████████████████████████████████████████████████████████████| 27541/27541
[00:00<00:00, 451080.21it/s]


Found embeddings for 89.70% of vocab
Found embeddings for  98.27% of all text

Top 10 out of vocabulary word:
------------------------------


  [('yukos', 323),
   ('blunkett', 189),
   ('gazprom', 123),
   ('kenteris', 114),
   ('iaaf', 111),
   ('boerse', 110),
   ('rosneft', 103),
   ('ebbers', 101),
   ('yugansk', 101),
   ('thanou', 99)]
```

6.1.2. Check Coverage for clean_text2_for_embedding

```
##Apply the vocab function to get the words and the corresponding counts
sentences = data["clean_text2_for_embedding"].apply(lambda x: x.split()).values
vocab = build_vocab(sentences)

print(f"\nFor clean_text2_for_embedding: \n{'-'*40}")
oov = check_coverage(vocab,embeddings_index)

## List 10 out of vocabulary word
print(f"\nTop 10 out of vocabulary word: \n{'-'*30}")
oov[:10]
```

```
100%|████████████████████████████████████████████████████████████████| 4023/4023
[00:00<00:00, 31157.84it/s]


For clean_text2_for_embedding:
----------------------------------------
```

```
100%|██████████████████████████████████████████████████████████████| 27530/27530
[00:00<00:00, 483533.24it/s]


Found embeddings for 89.71% of vocab
Found embeddings for  98.25% of all text

Top 10 out of vocabulary word:
------------------------------


  [('yukos', 337),
   ('blunkett', 198),
   ('boerse', 128),
   ('gazprom', 117),
   ('iaaf', 108),
   ('rosneft', 108),
   ('kenteris', 107),
   ('ebbers', 106),
   ('yugansk', 100),
   ('thanou', 93)]
```

6.1.3. Creating AVG W2V array

```
# Function to convert text into avg W2V
def avg_w2v_of_text(text):

    avg_w2v_sent = np.zeros(300)
    for word in text.split():

        try:
            avg_w2v_sent += embeddings_index.word_vec(word)
        except:
            pass

    avg_w2v_sent = avg_w2v_sent/len(text.split())
    return avg_w2v_sent
```

```
# For text1
cleaned_text1_avg_w2v =  data['clean_text1_for_embedding'].apply(lambda x: avg_
w2v_of_text(x))

# For text2
cleaned_text2_avg_w2v =  data['clean_text2_for_embedding'].apply(lambda x: avg_
w2v_of_text(x))
```

6.1.4 Finding Similarity using W2V Representation

```
# Finding Similarity using pair wise distance
similarity_score_w2v=[]
for i in range(data.shape[0]):
       similarity_score_w2v.append(cosine_similarity([cleaned_text1_avg_w2v[i
]],[cleaned_text2_avg_w2v[i]])[0][0])
```

```
# Scaling the similarity between [0-1]
similarity_score_w2v = minmax_scale(similarity_score_w2v, feature_range=(0, 1))

# Round off upto 2 decimal
similarity_score_w2v = np.round(similarity_score_w2v,decimals=3)
```

```
# Checking some similarity text
print(f"Index where similarity bestween text1 and text2 is greater than 0.9: {n
p.where(similarity_score_w2v>0.99)[0]}")

print(f"Similarity Score of those index: {similarity_score_w2v[np.where(similar
ity_score_w2v > 0.99)]}")
```

```
Index where similarity bestween text1 and text2 is greater than 0.9: [3403]
Similarity Score of those index: [1.]
```

```
# Sample of text where similarity score is greater than 0.99
i = 3403
print("Sample of text where similarity score is greater than 0.9\n\n")
print(f" text1: \n{'-'*7}\n{data.iloc[i].text1}\n")
print(f" text2: \n{'-'*7}\n{data.iloc[i].text2}\n")
```

```
Sample of text where similarity score is greater than 0.9


 text1:
-------
holmes starts 2005 with gb events kelly holmes will start 2005 with a series of races in bri
tain.  holmes will make her first track appearance on home soil since winning double olympic
gold in january s norwich union international in glasgow. she will also run in the grand pri
x in birmingham in february and may defend her indoor aaa 800m title in sheffield earlier th
at month.  i am still competitive and still want to win   she said.  i m an athlete and i ca
n t wait to get back on the track.  she added:  these events are also a great opportunity to
thank the british public for the enormous levels of support they have given me from the mome
nt i stepped off that plane from greece.  the glasgow meeting will see holmes compete over 1
500m in a five-way match against sweden  france  russia and italy.

 text2:
-------
holmes starts 2005 with gb events kelly holmes will start 2005 with a series of races in bri
tain.  holmes will make her first track appearance on home soil since winning double olympic
gold in january s norwich union international in glasgow. she will also run in the grand pri
x in birmingham in february and may defend her indoor aaa 800m title in sheffield earlier th
at month.  i am still competitive and still want to win   she said.  i m an athlete and i ca
n t wait to get back on the track.  she added:  these events are also a great opportunity to
thank the british public for the enormous levels of support they have given me from the mome
nt i stepped off that plane from greece.  the glasgow meeting will see holmes compete over 1
500m in a five-way match against sweden  france  russia and italy.
```

6.2. Approach 2: Tf-idf W2V Representation

```
def tfidf_w2v_of_dataframe(dataframe_text1, dataframe_text2):

    model = TfidfVectorizer()
    model.fit(dataframe_text1 + dataframe_text2)
```

```
    # we are converting a dictionary with word as a key, and the idf as a value
    dictionary = dict(zip(model.get_feature_names(), list(model.idf_)))

    # avg tfidf w2v conversion for text1
    avg_tfidf_w2v_text1 = []
    for i in tqdm(range(dataframe_text1.shape[0])):

        text = dataframe_text1.iloc[i]
        avg_tfidf_w2v_sent = np.zeros(300)
        len_of_text = len(text.split())
        weighted_sum =0

        for word in text.split():

            try:
                # dictionary[word] = idf value of word in whole courpus
                # text.count(word)/len_of_text = tf valeus of word in this revi
ew

                idf_word = dictionary[word]
                tf_word = text.count(word)/len_of_text

                tf_idf_word =  tf_word*idf_word
                weighted_sum += tf_idf_word

                avg_tfidf_w2v_sent += embeddings_index.word_vec(word) * tf_idf_
word

            except:
                pass

        avg_tfidf_w2v_sent = avg_tfidf_w2v_sent/weighted_sum
        avg_tfidf_w2v_text1.append(avg_tfidf_w2v_sent)

#===============================================================================
=======

    # avg tfidf w2v conversion for text2
    avg_tfidf_w2v_text2 = []
    for i in tqdm(range(dataframe_text2.shape[0])):

        text = dataframe_text2.iloc[i]
        avg_tfidf_w2v_sent = np.zeros(300)
        len_of_text = len(text.split())
        weighted_sum =0

        for word in text.split():

            try:
                # dictionary[word] = idf value of word in whole courpus
                # text.count(word)/len_of_text = tf valeus of word in this revi
ew

                idf_word = dictionary[word]
                tf_word = text.count(word)/len_of_text
```

```
                tf_idf_word =  tf_word*idf_word
                weighted_sum += tf_idf_word

                avg_tfidf_w2v_sent += embeddings_index.word_vec(word) * tf_idf_
word

            except:
                pass

        avg_tfidf_w2v_sent = avg_tfidf_w2v_sent/weighted_sum
        avg_tfidf_w2v_text2.append(avg_tfidf_w2v_sent)

    return avg_tfidf_w2v_text1 ,avg_tfidf_w2v_text2
```

```
cleaned_text1_tfidf_avg_w2v, cleaned_text2_tfidf_avg_w2v = tfidf_w2v_of_datafra
me(data['clean_text1_for_embedding'], data['clean_text2_for_embedding'])
```

```
100%|████████████████████████████████████████| 4023/402
3 [00:06<00:00, 575.76it/s]
100%|████████████████████████████████████████| 4023/402
3 [00:07<00:00, 569.07it/s]
```

6.2.1 Finding Similarity using TFIDF-W2V Representation

```
# Finding Similarity using pair wise distance
similarity_score_tfidf_w2v=[]
for i in range(data.shape[0]):
       similarity_score_tfidf_w2v.append(cosine_similarity([cleaned_text1_tfidf
_avg_w2v[i]],[cleaned_text2_tfidf_avg_w2v[i]])[0][0])


# Scaling the similarity between [0-1]
similarity_score_tfidf_w2v = minmax_scale(similarity_score_tfidf_w2v, feature_r
ange=(0, 1))

# Round off upto 2 decimal
similarity_score_tfidf_w2v = np.round(similarity_score_tfidf_w2v, decimals=3)
```

```
# Checking some similarity text
print(f"Index where similarity bestween text1 and text2 is greater than 0.9: {n
p.where(similarity_score_tfidf_w2v>0.9)[0]}")

print(f"Similarity Score of those index: {similarity_score_tfidf_w2v[np.where(s
imilarity_score_tfidf_w2v>0.9)]}")
```

```
Index where similarity bestween text1 and text2 is greater than 0.9: [2284 3056 3403 3859]
Similarity Score of those index: [0.998 0.959 1.    0.931]
```

```
# Sample of text where similarity score is greater than 0.9
i = 2284

print("Sample of text where similarity score is greater than 0.9\n\n")
```

```
print(f" text1: \n{'-'*7}\n{data.iloc[i].text1}\n")
print(f" text2: \n{'-'*7}\n{data.iloc[i].text2}\n")
```

```
Sample of text where similarity score is greater than 0.9


 text1:
-------
dvd copy protection strengthened dvds will be harder to copy thanks to new anti-piracy measu
res devised by copy protection firm macrovision.  the pirated dvd market is enormous because
current copy protection was hacked more than five years ago. macrovision says its new ripgua
rd technology will thwart most  but not all  of the current dvd ripping (copying) programs u
sed to pirate dvds.  ripguard is designed to... reduce dvd ripping and the resulting supply
of illegal peer to peer   said the firm. macrovision said the new technology will work in  n
early all  current dvd players when applied to the discs  but it did not specify how many ma
chines could have a problem with ripguard. some bbc news website users have expressed concer
ns that the new technology will mean that dvds will not work on pcs running the operating sy
stem linux. the new technology will be welcomed by hollywood film studios which are increasi
ngly relying on revenue from dvd sales.  the film industry has stepped up efforts to fight d
vd piracy in the last 12 months  taking legal action against websites which offer pirated co
pies of dvd movies for download.   ultimately  we see ripguard dvd... evolving beyond anti-p
iracy  and towards enablement of legitimate online transactions  interoperability in tomorro
w s digital home  and the upcoming high-definition formats   said steve weinstein  executive
vice president and general manager of macrovision s entertainment technologies group. macrov
ision said ripguard would also prevent against  rent  rip and return  - where people would r
ent a dvd  copy it and then return the original. ripguard is expected to be rolled out on dv
ds from the middle of 2005  the company said. the new system works specifically to block mos
t ripping programs - if used  those programs will now most likely crash  the company said. m
acrovision has said that rip guard can be updated if hackers find a way around the new anti-
copying measures.

 text2:
-------
dvd copy protection strengthened dvds will be harder to copy thanks to new anti-piracy measu
res devised by copy protection firm macrovision.  the pirated dvd market is enormous because
current copy protection was hacked more than five years ago. macrovision says its new ripgua
rd technology will thwart most  but not all  of the current dvd ripping (copying) programs u
sed to pirate dvds.  ripguard is designed to... reduce dvd ripping and the resulting supply
of illegal peer to peer   said the firm. macrovision said the new technology will work in  n
early all  current dvd players when applied to the discs  but it did not specify how many ma
chines could have a problem with ripguard. the new technology will be welcomed by hollywood
film studios which are increasingly relying on revenue from dvd sales.  the film industry ha
s stepped up efforts to fight dvd piracy in the last 12 months  taking legal action against
websites which offer pirated copies of dvd movies for download.   ultimately  we see ripguar
d dvd... evolving beyond anti-piracy  and towards enablement of legitimate online transactio
ns  interoperability in tomorrow s digital home  and the upcoming high-definition formats
said steve weinstein  executive vice president and general manager of macrovision s entertai
nment technologies group. macrovision said ripguard was designed to plug the  digital hole
that was created by so-called decss ripper software. it circumvents content scrambling syste
m measures placed on dvds and let people make perfect digital copies of copyrighted dvds in
minutes. those copies could then be burned onto a blank dvd or uploaded for exchange to a pe
er-to-peer network. macrovision said ripguard would also prevent against  rent  rip and retu
rn  - where people would rent a dvd  copy it and then return the original. ripguard is expec
ted to be rolled out on dvds from the middle of 2005  the company said. the new system works
specifically to block most ripping programs - if used  those programs will now most likely c
rash  the company said. macrovision has said that rip guard can be updated if hackers find a
way around the new anti-copying measures.
```

6.3. Approach 3: Finding Similarity Using Spacy api

Refer : https://stackoverflow.com/a/44102463

```
import spacy
import en_core_web_sm

nlp =spacy.load('en_core_web_sm')

similarity_score_spacy = []
for i in range(data.shape[0]):

    doc1 = nlp(data['clean_text1_for_embedding'].iloc[i])
    doc2 = nlp(data['clean_text2_for_embedding'].iloc[i])
    similarity_score_spacy.append(doc1.similarity(doc2))

# Scaling the similarity between [0-1]
similarity_score_spacy = minmax_scale(similarity_score_spacy, feature_range=(0,
1))

# Round off upto 2 decimal
similarity_score_spacy = np.round(similarity_score_spacy, decimals=3)
```

```
# Checking some similarity text
print(f"Index where similarity bestween text1 and text2 is greater than 0.9: {n
p.where(similarity_score_spacy>0.95)[0]}")

print(f"Similarity Score of those index: {similarity_score_spacy[np.where(simil
arity_score_spacy>0.95)]}")
```

```
Index where similarity bestween text1 and text2 is greater than 0.9: [ 462  684 1483 2270 22
84 3403 3859]
Similarity Score of those index: [0.953 0.951 0.951 0.953 0.988 1.    0.951]
```

```
# Sample of text where similarity score is greater than 0.99
i = 2488

print("Sample of text where similarity score is greater than 0.9\n\n")
print(f" text1: \n{'-'*7}\n{data.iloc[i].text1}\n")
print(f" text2: \n{'-'*7}\n{data.iloc[i].text2}\n")
```

```
Sample of text where similarity score is greater than 0.9


 text1:
-------
gadgets galore on show at fair the 2005 consumer electronics show in las vegas is a geek s p
aradise with more than 50 000 new gadgets and technologies launched during the four-day even
t.  top gadgets at the show are highlighted in the innovations showcase  which recognises so
me of the hottest developments in consumer electronics. the bbc news website took an early p
re-show look at some of those technologies that will be making their debut in 2005.      one
of the key issues for keen gadget users is how to store all their digital images  audio and
video files. the 2.5gb and 5gb circular pocket hard drive from seagate might help. the exter
nal usb drive won a ces best innovations design and engineering award and is small enough to
slip into a pocket.  it is the kind of storage that appeals to people who want their pcs to
look cool   said seagate.  it is all about style but it also has lots of functionality.   it
is the first time you can say a hard drive is sexy   it said. in the centre of the device is
a blue light that flashes while data is being written to ensure users do not unplug it when
```

it is busy saving those precious pictures.   universal electronics nevosl is a universal controller that lets people use one device to get at their multimedia content such as photo s no matter where it is in their house. it can also act as a remote for home theatre and st ereo systems. working with home broadband networks and pcs the gadget has built-in wireless and a colourful simple interface. paul arling uei chief said consumers face real problems when trying to get at all the files they own that are typically spread across several differ ent devices. he said the nevo gave people a simple single way to regain some control over d igital media in the home. the nevo won two awards at ces one as a girl s best friend award and another for innovation design and engineering. the gadget is expected to go on sale bef ore the summer and will cost about $799 (£425).   hotseat is targeting keen gamers with m oney to spend with its solo chassis gaming chair. the specially-designed chair lets gamers p lay in surround-sound while stretching out in their own space . it is compatible with all t he major games consoles dvd players and pcs. we found that kids love playing in surround s ound   said jay leboff from hotseat. we are looking at offering different types of seats d epending on the market success of this one. the chair also lets people experience surround sound while watching videos with wireless control for six surround sound speakers. and a dr inks holder. the chair which looks like a car seat on a skeletal frame should go on sale i n april and is expected to cost $399 (£211).   satellite radio is big business in the us. in the uk the digital radio technology is known as dab and works on slightly different tech nology. eton corporation s porsche designed p7131 digital radio set will be launched both as a dab radio in the uk as well as a satellite radio set in the us. dab sets have been slow to take-off in the uk but this one concentrates on sleek looks as much as technology. it is f or the risqué consumer   said an eton spokesperson. we are proud of it because it has the sound quality for the audiophile and the looks for the design-conscious consumer. the p orsche radio is set to go on sale at the end of january in the us and in the first quarter o f 2005 in the uk. in the us is it expected to cost $250 (£133).   the average person has a library of 600 digital images estimates the consumer electronics association the organisa tion behind ces. this is expected to grow to a massive 3 420 images - or 7.2gb - in five yea rs time. one gadget that might help swell that collection is sanyo s tiny handheld vpc-c4 c amcorder which is another innovation in design and engineering award winner. it combines hig h quality video and stills in a very small device. it takes mpeg4 video quality at 30 frames a second and has a four megapixel still camera. images and video are stored on sd cards whi ch have come down in price in recent months. a 512mb card will store about 30 minutes of vid eo and 420 stills. the device is so tiny it can be controlled with one thumb. because images and video are stored on sd memory it is portable to other devices and means other data like audio can be stored on the card too.   wearable technology has always promised much but f ailed to deliver because of lack of storage capability and poor design. mpio s tiny digital usb music players come in an array of fashionable colours taking a leaf out of the apple ip od mini book of design and reflecting the desire for gadgets that look good. slung on a cord the player would not look too geeky dangling discreetly from the neck. although the pendant design was launched three months ago the device emphasises large storage as well as good lo oks for fashion-conscious gadget fiends. an even dinkier model the fy500 comes out in may and will store about 256mb of music. the range of players recently won an international foru m design award 2005.

text2:

----

rivals of the £400 apple... the mac mini is the cheapest apple computer ever. but though it is cheap for a mac how does it compare to pcs that cost about the same amount dot.life trie s to find out if you can you get more for your money if you stick with the beige box.   a n extremely small computer that is designed to bring the macintosh to the masses. apple off er a less powerful mac mini for £339 but the £399 models has a 1.4ghz power pc chip 80 giga byte hard drive combined cd burner/dvd player. it comes equipped with usb and firewire port s for peripheral connections ethernet port for broadband a port for standard video output and an audio/headphone jack.the machine comes with mac os x the apple operating system the software suite ilife which includes itunes iphoto imovie idvd and garageband. a monitor keyboard or mouse. there is also no built-in support for wireless technology or any speaker s. the lack of a dvd burner is an omission in the age of backing-up important software. wire less and a dvd burner can be added at extra cost. apple are targeting people who already ha ve a main computer and want to upgrade - especially pc users who have used an apple ipod. c ompact and stylish the mac mini would not look out of place in any home. apple computers ar e famously user friendly and offer much better network security which means fewer viruses.

the package of software that comes with the machine is the best money can buy. the mac mini is just a box. if you don t already have a monitor etc adding them to the package sees the value for money begin to dwindle. macs don t offer the upgrade flexibility of a pc and the machine s specifications lack the horse power for tasks such as high-end video editing or games. the mac mini puts the macintosh within the reach of everyone an apple spokesman said. it will bring more customers to the platform especially pc users and owners. an entry-level machine designed for basic home use. a 2.6ghz intel celeron chip 40 gigabyte hard drive 256mb combined cd burner/dvd player. it comes equipped with a 17 inch monitor keyboard and mouse. the machine has 6 usb ports and an ethernet port for broadband connection. there s also a port for standard video output. the machine comes with windows xp home edition. it provides basic home tools such as a media player and word processor. a dvd burner or any wireless components built in. wireless and a dvd burner can be added at extra cost. homes and small offices including those looking to add a low cost second computer. cost is the clear advantage. the dell provides enough power and software for basic gaming and internet surfing. it s easily upgradeable so a bigger hard drive better sound and graphics cards can be added. the dell is hardly stylish and the hard drive is on the small size for anyone wanting to store photos or a decent sized digital music collection. this machine is for small businesses and for people who want a second computer for basic home use perhaps in a kids bedroom a spokesman for dell said. i think we offer better value once you realise all the extras needed for the mac mini. a desktop computer that pc pro magazine dubbed best performer in a group test of machines that cost only £399 (£469 including vat). a good basic pc that according to pc pro has superb upgrade potential . for your money you get a 1.8ghz amd sempron processor 512mb of ram 120gb hard drive dvd writer 16-inch monitor mouse keyboard and windows xp2 much more than the basics. it cannot handle 3d graphics and has no firewire slots. those on a limited budget who want a machine they can add to and improve as their cash allows. it s cheap and has plenty of room to improve but that could end up making it expensive in the long run. it s a good basic workhorse. it s not pretty and has a monitor rather than a flat-panel display. some of the upgrades offered by jal to the basic model are pricey. you might find that you want to chop and change quite quickly. nick ross deputy labs editor at pc pro said the important point about buying a cheap and cheerful pc is the upgrade path. interest has switched from processor power to graphics and sound cards as that s what makes the difference in games. even manufacturers are not going to be marketing machines as faster he said they ll emphasise the different features. a computer built from bits you buy and put together yourself. a surprisingly good pc sporting an amd athlon xp 2500 processor 512 megabytes ram a graphics card with 128 ram on board plus tv out a 40 gb hard drive cd-writer and dvd player windows xp home. anything else. you re building it so you have to buy all the software you want to install and do your own trouble-shooting and tech support. building your own machine is easier than it used to be but you need to read specifications carefully to make sure all parts work together. experienced and keen pc users. building your own pc or upgrading the one you have is a great way to improve your understanding of how it all works. it s cheap you can specify exactly what you want and you get the thrill of putting it together yourself. and a bigger thrill if everything works as it should. once it s built you won t be able to do much with it until you start buying software for it. if it starts to go wrong it might take a lot of fixing. as gavin cox of the excellent buildyourown.org.uk website put it: it will be tough to obtain/build a pc to ever be as compact and charming as the mac mini. performance-wise it s not cutting edge and is barely entry-level by today s market but up against the mac mini i believe it will hold its own and even pull a few more tricks says gavin cox. the good news is that the machine is eminently expandable. by contrast says mr cox the mac mini is almost disposable.

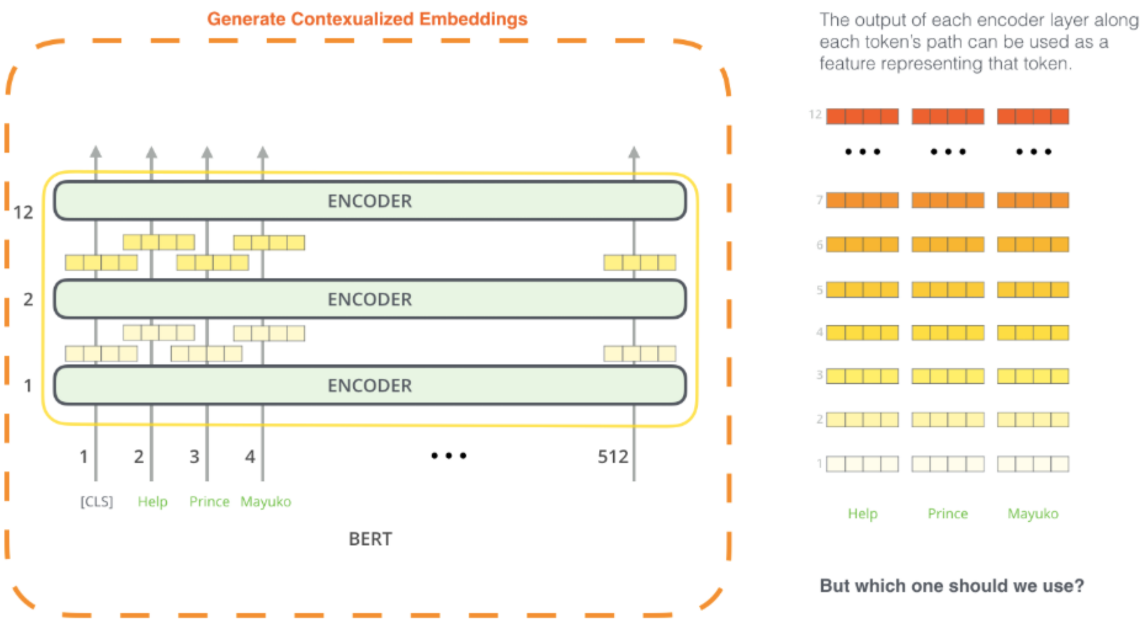
6.4. Approach 4: Using AVG BERT Embedding to find Similarity

Refer: https://jalammar.github.io/illustrated-bert/

```python
from IPython.display import Image
Image(filename='bert fine tuning.png')
```

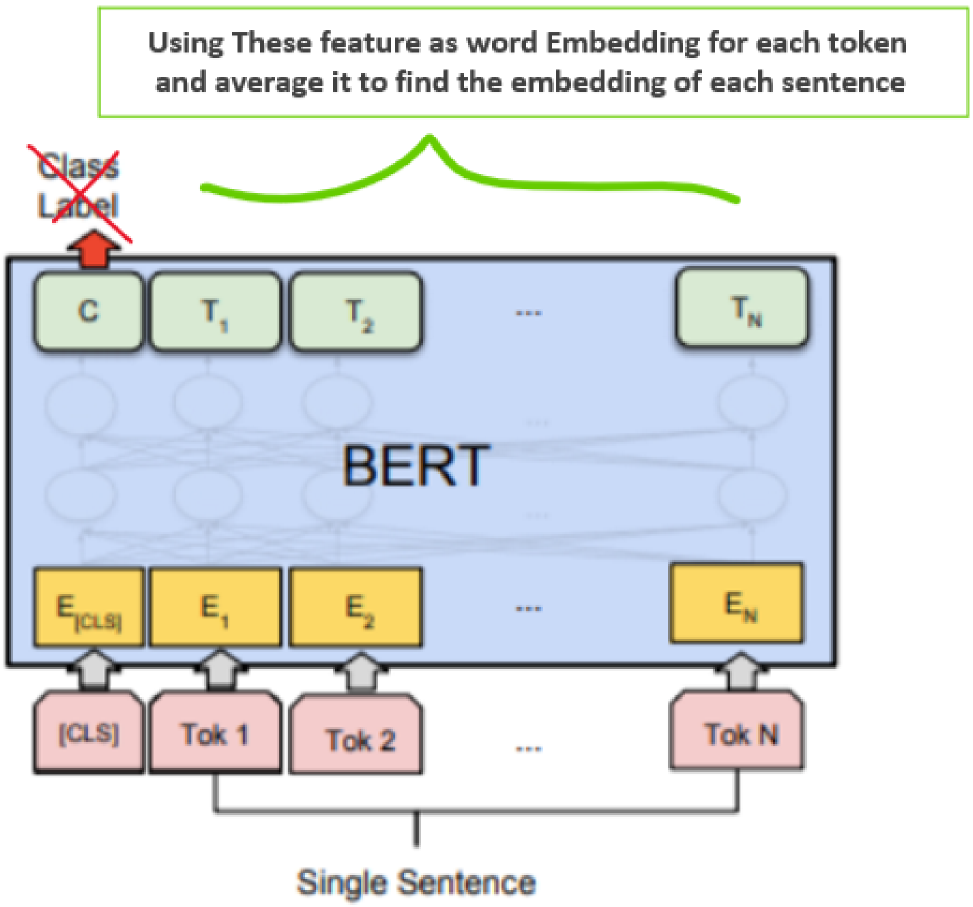
BERT for feature extraction

The fine-tuning approach isn't the only way to use BERT. Just like ELMo, you can use the pre-trained BERT to create contextualized word embeddings. Then you can feed these embeddings to your existing model – a process the paper shows yield results not far behind fine-tuning BERT on a task such as named-entity recognition.



Which vector works best as a contextualized embedding? I would think it depends on the task. The paper examines six choices (Compared to the fine-tuned model which achieved a score of 96.4):

```
from IPython.display import Image
Image(filename='bert.png')
```



6.4.1. Finding AVG BERT Embedding

```
# Loading pretrained Model and Tokenizer

bert_model = BertModel.from_pretrained('bert-base-uncased')
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

bert_model.cuda()
print("Model loaded on gpu.." )
```

```
Model loaded on gpu..
```

```
# function to find embeding of test using Bert
def bert_embedding_of_text(text,max_len = 128):

    # Step 1: Tokenize
    tokens = tokenizer.tokenize(text)

    # Step 2: Add [CLS] and [SEP]
    tokens = ['[CLS]'] + tokens + ['[SEP]']

    # Step 3.1: Pad tokens
    if len(tokens) < max_len:
        padded_tokens = tokens + ['[PAD]' for _ in range(max_len - len(tokens
))]
    else:
```

```
        padded_tokens = tokens[:max_len-1] + ['[SEP]']

    # Step 3.2: atttention mask
    attn_mask = [1 if token != '[PAD]' else 0 for token in padded_tokens]

    # Step 4: Segment ids
    seg_ids = [0 for _ in range(len(padded_tokens))] #Optional!

    # Step 5: Get BERT vocabulary index for each token
    token_ids = tokenizer.convert_tokens_to_ids(padded_tokens)


    ## Bert Embeddings of words except [cls] token
    # Convert to pytorch tensors
    token_ids = torch.tensor(token_ids).unsqueeze(0).cuda()
    attn_mask = torch.tensor(attn_mask).unsqueeze(0).cuda()
    seg_ids = torch.tensor(seg_ids).unsqueeze(0).cuda()

    # Feed them to bert
    hidden_reps, cls_head = bert_model(token_ids, attention_mask = attn_mask,\
                                    token_type_ids = seg_ids)

    # Covert torch tensor to numpy array
    hidden_reps = hidden_reps.detach()

    return hidden_reps.squeeze()

#==============================================================

# Function to find similarity using BERT avg Embedding using only 512 as max_le
n
def similarity_using_bert(text1, text2,max_length = 512):

    temp1 = bert_embedding_of_text(text1, max_length)
    temp2 = bert_embedding_of_text(text2, max_length)

    cos_sim = torch.nn.functional.cosine_similarity(temp1.mean(dim=1).unsqueeze
(0) ,temp2.mean(dim=1).unsqueeze(0))

    return cos_sim.cpu().detach().numpy()[0]
```

6.4.2. Finding Similarity

```
# Finding similarity

similarity_score_bert = []
for i in tqdm(range(data.shape[0])):

    sentence1 = data['text1'].iloc[i]
    sentence2 = data['text2'].iloc[i]
    similarity_score_bert.append(similarity_using_bert(sentence1, sentence2, ma
x_length = 512))

# Scaling the similarity between [0-1]
similarity_score_bert = minmax_scale(similarity_score_bert, feature_range=(0, 1
```

```
))

# Round off upto 2 decimal
similarity_score_bert = np.round(similarity_score_bert, decimals=3)
```

```
100%|████████████████████████████████████████████████████████████████| 4023/40
23 [04:58<00:00, 13.50it/s]
```

```
# Checking some similarity text
print(f"Index where similarity bestween text1 and text2 is greater than 0.9: {n
p.where(similarity_score_bert>0.8)[0]}")

print(f"Similarity Score of those index: {similarity_score_bert[np.where(simila
rity_score_bert>0.8)]}")
```

```
Index where similarity bestween text1 and text2 is greater than 0.9: [3403]
Similarity Score of those index: [1.]
```

```
# Sample of text where similarity score is greater than 0.90
i = 3403

print("Sample of text where similarity score is greater than 0.9\n\n")
print(f"text1: \n{'-'*7}\n{data.iloc[i].text1}\n")
print(f"text2: \n{'-'*7}\n{data.iloc[i].text2}\n")
```

```
Sample of text where similarity score is greater than 0.9


text1:
-------
holmes starts 2005 with gb events kelly holmes will start 2005 with a series of races in bri
tain.  holmes will make her first track appearance on home soil since winning double olympic
gold in january s norwich union international in glasgow. she will also run in the grand pri
x in birmingham in february and may defend her indoor aaa 800m title in sheffield earlier th
at month.  i am still competitive and still want to win   she said.  i m an athlete and i ca
n t wait to get back on the track.  she added:  these events are also a great opportunity to
thank the british public for the enormous levels of support they have given me from the mome
nt i stepped off that plane from greece.  the glasgow meeting will see holmes compete over 1
500m in a five-way match against sweden  france  russia and italy.

text2:
-------
holmes starts 2005 with gb events kelly holmes will start 2005 with a series of races in bri
tain.  holmes will make her first track appearance on home soil since winning double olympic
gold in january s norwich union international in glasgow. she will also run in the grand pri
x in birmingham in february and may defend her indoor aaa 800m title in sheffield earlier th
at month.  i am still competitive and still want to win   she said.  i m an athlete and i ca
n t wait to get back on the track.  she added:  these events are also a great opportunity to
thank the british public for the enormous levels of support they have given me from the mome
nt i stepped off that plane from greece.  the glasgow meeting will see holmes compete over 1
500m in a five-way match against sweden  france  russia and italy.
```

7. Result And Conclusion

7.1. Saving Result

```
# Saving submission File
submission_w2v = pd.DataFrame({'Unique_ID':data.Unique_ID, 'similarity_score_w2v':similarity_score_w2v})
submission_w2v.to_csv(path_or_buf= 'submission_w2v.csv', sep=',',)

# Saving submission File
submission_tfidf_w2v = pd.DataFrame({'Unique_ID':data.Unique_ID, 'similarity_score_tfidf_w2v':similarity_score_tfidf_w2v})
submission_tfidf_w2v.to_csv(path_or_buf= 'submission_tfidf_w2v.csv', sep=',',)

# Saving submission File
submission_spacy = pd.DataFrame({'Unique_ID':data.Unique_ID, 'similarity_score_spacy':similarity_score_spacy})
submission_spacy.to_csv(path_or_buf= 'submission_spacy.csv', sep=',',)

# Saving submission File
submission_bert = pd.DataFrame({'Unique_ID':data.Unique_ID, 'similarity_score_bert':similarity_score_spacy})
submission_bert.to_csv(path_or_buf= 'submission_bert.csv', sep=',',)
```

Conclusion

- These approaches are just some first cut solution to the problem.
- We can see using mentioned 4 approaches we are getting quite decent results; some approaches are obviously better than otheres because of better embeddings.
- TfIDF avg W2V has done really well to find the similarity scores.
- Results could be definitely improved using some addition techines like dimension reduction techniques and matrix factorisation techniques.
- Need to explore more "how to tackle unsupervised problem ? "

END :)