```
import pandas as pd
import numpy as np
from google.colab import files
```

```
uploaded = files.upload()
```

选择文件 FullData.csv

- **FullData.csv**(application/vnd.ms-excel) - 10845337 bytes, last modified: 2021/3/29 - 100% done

```
Saving FullData.csv to FullData.csv
```

```
import warnings
warnings.filterwarnings("ignore")
import io
```

Load Data

```
df=pd.read_csv(io.BytesIO(uploaded['FullData.csv']))
df['patterns'] = df.txt
#df['patterns'] = df.MsgBody
```

```
df.Celebrity.unique()
```

```
array(['Kerwin Frost', 'Beyonce', 'Zoe Saldana', 'Karlie Kloss',
       'Yara Shahidi', 'Pharrell Williams', 'Adriene Mishler',
       'Ninjas Hyper', 'Bad Bunny', 'Jerry Lorenzo', 'Chinae Alexander',
       'Ally Love', 'BlackPink', 'Naeun Son', 'Seolhyun', 'Solar',
       'GFriend', 'iZone', 'BTS', 'NCT'], dtype=object)
```

```
df.Celebrity.value_counts()
```

```
NCT            6348
iZone          4911
GFriend        4492
BlackPink      3433
```

```
BTS                  3135
Seolhyun             2777
Solar                2484
Naeun Son            2439
Bad Bunny            2183
Ninjas Hyper         1890
Karlie Kloss         1850
Yara Shahidi         1563
Kerwin Frost         1330
Beyonce              1279
Zoe Saldana          1278
Jerry Lorenzo        1161
Pharrell Williams     966
Chinae Alexander      955
Ally Love             876
Adriene Mishler       152
Name: Celebrity, dtype: int64
```

```
df.isnull().sum()
```

```
Celebrity       0
id              0
author          0
subreddit       0
Date            0
Score           0
num_comments    0
txt             0
patterns        0
dtype: int64
```

Clean data

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
nltk.download('wordnet')
import string
from textblob import Word
import re
```

```
import re
```

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

```
df=df.groupby(["Celebrity","txt"]).size().reset_index(name="freq")
df['patterns'] = df.txt
```

```
stop = stopwords.words('english')
df['patterns'] = df['patterns'].apply(lambda x:' '.join(x.lower() for x in x.split()))
df['patterns'] = df['patterns'].apply(lambda x:' '.join(x for x in x.split() if x not in string.punctuation)) #remove pun
df['patterns'] = df['patterns'].str.replace('https*\S+','')   #remove url
df['patterns'] = df['patterns'].str.replace('\'\w+','')          #remove ticks
df['patterns'] = df['patterns'].str.replace('[^\w\s]','')
df['patterns'] = df['patterns'].str.replace('@\S+','')             #remove email
df['patterns'] = df['patterns'].str.encode('ascii', 'ignore').str.decode("utf-8")   #remove unicode
df['patterns'] = df['patterns'].str.replace('\w*\d+\w*','')   #remove digits
df['patterns'] = df['patterns'].str.replace('#\S+','')   #remove hashtag
df['patterns'] = df['patterns'].str.replace('_','')   #remove underscore
df['patterns'] = df['patterns'].apply(lambda x: ' '.join(x for x in x.split() if  not x.isdigit()))
df['patterns'] = df['patterns'].apply(lambda x:' '.join(x for x in x.split() if not x in stop)) #remove stop words
df['patterns'] = df['patterns'].apply(lambda x: " ".join([Word(word).lemmatize() for word in x.split()]))
```

Define Cosine Similarity

```
def most_similar(doc_id,similarity_matrix,matrix):
    print (f'Document: {documents_df.iloc[doc_id]}')
    print ('\n')
    print ('Similar Documents:')
    if matrix=='Cosine Similarity':
        similar_ix=np.argsort(similarity_matrix[doc_id])[::-1]
    elif matrix=='Euclidean Distance':
        similar_ix=np.argsort(similarity_matrix[doc_id])
    for ix in similar_ix:
        if ix==doc_id:
```

```
            if ix==doc_id:
                continue
            print (f'Celebrity: {documents_df.index[ix]}',":",similarity_matrix[doc_id][ix])
            #print (f'{matrix} : {similarity_matrix[doc_id][ix]}')
```

Tfidf with cosine similarity

https://towardsdatascience.com/calculating-document-similarities-using-bert-and-other-models-b2c1a29c9630

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.metrics.pairwise import euclidean_distances
```

```
col=["Celebrity","patterns"]
df_merge = df[col].groupby('Celebrity')['patterns'].apply(lambda x:x.str.cat(sep=" "))
df_merge
```

```
Celebrity
Adriene Mishler      hii im vicky let workout buddy stay motivated ...
Ally Love            hi guy anyone know meaning elusive playgirlfox...
BTS                  jimin shoe ever lol kim jongkook focused game ...
Bad Bunny            guy leave immediately run reddit post vod grav...
Beyonce              fake hair hey bestie beyonc diffewent account ...
BlackPink            cute someone fix neck hey thinker great post t...
Chinae Alexander     japan must losing naval war unlock kamikaze sl...
GFriend               stair north snapshot post archiveorg megalodo...
Jerry Lorenzo        detail pls contact always glad help detail pls...
Karlie Kloss         pretty foot zen android original submission rk...
Kerwin Frost         idea bro tryna found stuff telethon shake cons...
NCT                  taeyongblack ten black taeil black haechan bla...
Naeun Son            dont play vr cannot really participative think...
Ninjas Hyper         anything valheim willing trade yakuza kiwami d...
Pharrell Williams    enjoy day like shoe want know detail shoe cont...
Seolhyun             seolhyun boy jonghyun shinee seungyoon winner ...
Solar                youtube video link solars youtube channel yout...
Yara Shahidi         please remember representation andrew yang par...
Zoe Saldana          fake hair hey bestie beyonc diffewent account ...
```

```
iZone                begin pgp message end pgp message begin pgp me...
Name: patterns, dtype: object
```

```
documents_df=pd.DataFrame(df_merge.values,columns=['documents'])
documents_df.index = df_merge.index
```

```
# removing special characters and stop words from the text
#stop_words_l=stopwords.words('english')
max_feature = 5000
tfidfvectoriser=TfidfVectorizer(max_features=max_feature)
tfidfvectoriser.fit(documents_df.documents)
tfidf_vectors=tfidfvectoriser.transform(documents_df.documents)
```

```
tfidf_vectors.shape
```

```
(20, 5000)
```

```
tfidf_vectors=tfidf_vectors.toarray()
```

```
pairwise_similarities1=np.dot(tfidf_vectors,tfidf_vectors.T)
pairwise_differences1=euclidean_distances(tfidf_vectors)
```

```
pairwise_similarities1.shape
```

```
(20, 20)
```

```
for i in range(len(df_merge)):
    print(i,":",df_merge.index[i])
```

```
0 : Adriene Mishler
1 : Ally Love
2 : BTS
3 : Bad Bunny
4 : Beyonce
```

```
5 : BlackPink
6 : Chinae Alexander
7 : GFriend
8 : Jerry Lorenzo
9 : Karlie Kloss
10 : Kerwin Frost
11 : NCT
12 : Naeun Son
13 : Ninjas Hyper
14 : Pharrell Williams
15 : Seolhyun
16 : Solar
17 : Yara Shahidi
18 : Zoe Saldana
19 : iZone
```

```
most_similar(5,pairwise_similarities1,'Cosine  Similarity')
```

```
Document: documents    cute someone fix neck hey thinker great post t...
Name: BlackPink, dtype: object


Similar Documents:
Celebrity: iZone : 0.790563777396686
Celebrity: BTS : 0.7699359218506847
Celebrity: NCT : 0.7521877379939635
Celebrity: Solar : 0.6850817702225174
Celebrity: GFriend : 0.67294263845l0462
Celebrity: Seolhyun : 0.6495518696531659
Celebrity: Naeun Son : 0.6228490352239701
Celebrity: Ally Love : 0.6074284308858239
Celebrity: Jerry Lorenzo : 0.5802851619840322
Celebrity: Yara Shahidi : 0.5334916031933165
Celebrity: Bad Bunny : 0.5136057315978038
Celebrity: Zoe Saldana : 0.5107611024997509
Celebrity: Beyonce : 0.5088112159341345
Celebrity: Pharrell Williams : 0.4657860352246673
Celebrity: Karlie Kloss : 0.459268598094079l6
Celebrity: Ninjas Hyper : 0.4382743282605043
Celebrity: Kerwin Frost : 0.4242254661001755
Celebrity: Chinae Alexander : 0.38793475784583714
Celebrity: Adriene Mishler : 0.19016591343946898
```

```
most_similar(5,pairwise_differences1,'Euclidean  Distance')
```

```
Document: documents    cute someone fix neck hey thinker great post t...
Name: BlackPink, dtype: object


Similar Documents:
Celebrity: iZone : 0.6472035577827382
Celebrity: BTS : 0.6783274698098493
Celebrity: NCT : 0.704006053959828
Celebrity: Solar : 0.793622365835906
Celebrity: GFriend : 0.808773591988458
Celebrity: Seolhyun : 0.8371954734073023
Celebrity: Naeun Son : 0.8685055725509592
Celebrity: Ally Love : 0.886083031226962
Celebrity: Jerry Lorenzo : 0.9162039489283735
Celebrity: Yara Shahidi : 0.9659279443174712
Celebrity: Bad Bunny : 0.9863004292832914
Celebrity: Zoe Saldana : 0.9891803652522159
Celebrity: Beyonce : 0.9911496194479115
Celebrity: Pharrell Williams : 1.0336478750283737
Celebrity: Karlie Kloss : 1.0399340382023528
Celebrity: Ninjas Hyper : 1.0599298766800556
Celebrity: Kerwin Frost : 1.0731025430030707
Celebrity: Chinae Alexander : 1.1064043041801386
Celebrity: Adriene Mishler : 1.2726618455509187
```

Word2vec + TFIDF with cosine similarity

```
from  keras.preprocessing.text  import  Tokenizer
from  keras.preprocessing.sequence  import  pad_sequences
import  gensim
from  gensim.models  import  Word2Vec
```

```
#  tokenize  and  pad  every  document  to  make  them  of  the  same  size
tokenizer=Tokenizer()
tokenizer.fit_on_texts(documents_df.documents)
```

```
tokenizer.fit_on_texts(documents_df.documents)
tokenized_documents=tokenizer.texts_to_sequences(documents_df.documents)
tokenized_paded_documents=pad_sequences(tokenized_documents,maxlen=max_feature,padding=’post’)
vocab_size=len(tokenizer.word_index)+1
```

```
#### train word2vec
data=[]
for i in df[’patterns’]:
    li = list(i.split(″ ″))
    data.append(li)
```

This Google Developers blog post says:

Well, the following "formula" provides a general rule of thumb about the number of embedding dimensions:

embedding_dimensions = number_of_categories**0.25

That is, the embedding vector dimension should be the 4th root of the number of categories.

Interestingly, the Word2vec Wikipedia article says (emphasis mine):

Nevertheless, for skip-gram models trained in medium size corpora, with 50 dimensions, a window size of 15 and 10 negative samples seems to be a good parameter setting.

Assuming a standard-ish sized vocabulary of 1.5 million words, this rule of thumb comes surprisingly close:

50 == 1.5e6 ** 0.2751

Parameters: https://radimrehurek.com/gensim/models/word2vec.html

```
### calculate the vector_size
def flatten(data):
    return ″ ″.join([str(item) for var in data for item in var])

num_words = 0
for item in flatten(data):
    num_words += len(item)
num_words
```

```
1481617
```

```
import math
dim_size = num_words**0.25
vector_size = math.ceil(float(dim_size))
vector_size
```

```
35
```

```
# Create skip-gram model
model1 = gensim.models.Word2Vec(data,
                                min_count = 5,
                                size = vector_size,
                                window = 15,
                                negative= 10,
                                sg=1)
#model1.build_vocab(data)
model1.train(data, total_examples=model1.corpus_count,epochs=10)
```

```
(1878326, 2267140)
```

```
model_w2v = model1
```

```
# creating embedding matrix, every row is a vector representation from the vocabulary indexed by the tokenizer index.
embedding_matrix=np.zeros((vocab_size,vector_size))
for word,i in tokenizer.word_index.items():
    if word in model_w2v:
        embedding_matrix[i]=model_w2v[word]
# creating document-word embeddings
document_word_embeddings=np.zeros((len(tokenized_paded_documents),max_feature,vector_size))
for i in range(len(tokenized_paded_documents)):
    for j in range(len(tokenized_paded_documents[0])):
        document_word_embeddings[i][j]=embedding_matrix[tokenized_paded_documents[i][j]]
document_word_embeddings.shape
```

```
(20, 5000, 35)
```

```
embedding_matrix[tokenizer.word_index['kpop']]
```

```
array([-4.03511412e-02,  1.91909989e-04,  1.15708068e-01,  3.29520404e-01,
       -6.93451092e-02, -5.02134979e-01, -3.64862755e-02,  5.68540812e-01,
        4.98470634e-01,  7.39543200e-01, -4.18413967e-01,  6.89668357e-01,
        6.90077320e-02,  1.39977574e-01,  7.28365257e-02, -2.37692624e-01,
       -3.95094395e-01,  1.23837322e-01, -6.43764377e-01, -5.66305757e-01,
        5.65280914e-01,  1.04139125e+00,  7.86042333e-01,  2.36562695e-02,
        1.40126979e+00,  1.27799496e-01,  4.61241663e-01, -4.13074195e-02,
       -7.23924160e-01,  1.02353084e+00, -1.60487086e-01,  6.44798517e-01,
       -1.44350186e-01, -1.53855324e-01,  8.31936821e-02])
```

```
# tf-idf vectors do not keep the original sequence of words, converting them into actual word sequences from the docu
document_embeddings=np.zeros((len(tokenized_paded_documents),vector_size))
words=tfidfvectoriser.get_feature_names()


for i in range(len(document_word_embeddings)):
    for j in range(len(words)):
        document_embeddings[i]+=embedding_matrix[tokenizer.word_index[words[j]]]*tfidf_vectors[i][j]

document_embeddings=document_embeddings/np.sum(tfidf_vectors,axis=1).reshape(-1,1)
```

```
pairwise_similarities2=cosine_similarity(document_embeddings)
pairwise_differences2=euclidean_distances(document_embeddings)
```

```
for i in range(len(df_merge)):
      print(i,":",df_merge.index[i])
```

```
0 : Adriene Mishler
1 : Ally Love
2 : BTS
3 : Bad Bunny
4 : Beyonce
5 : BlackPink
6 : Chinae Alexander
7 : GFriend
```

```
8 : Jerry Lorenzo
9 : Karlie Kloss
10 : Kerwin Frost
11 : NCT
12 : Naeun Son
13 : Ninjas Hyper
14 : Pharrell Williams
15 : Seolhyun
16 : Solar
17 : Yara Shahidi
18 : Zoe Saldana
19 : iZone
```

```
most_similar(5,pairwise_similarities2,'Cosine  Similarity')
```

```
Document: documents    cute someone fix neck hey thinker great post t...
Name: BlackPink, dtype: object


Similar Documents:
Celebrity: Solar : 0.9930507776518557
Celebrity: BTS : 0.9923860788046267
Celebrity: iZone : 0.9923011165462109
Celebrity: NCT : 0.9875918941250481
Celebrity: Seolhyun : 0.9852364710361722
Celebrity: GFriend : 0.9822094707061249
Celebrity: Naeun Son : 0.9792640514450628
Celebrity: Yara Shahidi : 0.9761049982249927
Celebrity: Ally Love : 0.9750803201398447
Celebrity: Kerwin Frost : 0.9740044222621533
Celebrity: Karlie Kloss : 0.966556629728006
Celebrity: Bad Bunny : 0.9617148541316574
Celebrity: Adriene Mishler : 0.9575133794506746
Celebrity: Jerry Lorenzo : 0.957504855625673
Celebrity: Zoe Saldana : 0.9551448512733475
Celebrity: Beyonce : 0.9548660804143762
Celebrity: Ninjas Hyper : 0.9400557579012934
Celebrity: Pharrell Williams : 0.9267610651694266
Celebrity: Chinae Alexander : 0.8972835589448773
```

```
most_similar(5,pairwise_differences2,'Euclidean  Distance')
```

```
Document: documents    cute someone fix neck hey thinker great post t...
Name: BlackPink, dtype: object


Similar Documents:
Celebrity: Solar : 0.189492618734733 86
Celebrity: BTS : 0.1969673458858676
Celebrity: iZone : 0.20100381767445974
Celebrity: NCT : 0.2545834549940307
Celebrity: Seolhyun : 0.2867532140265033
Celebrity: GFriend : 0.3137458134017913
Celebrity: Yara Shahidi : 0.3450308649886061
Celebrity: Naeun Son : 0.3458382539705487
Celebrity: Ally Love : 0.35203342999471965
Celebrity: Kerwin Frost : 0.3591151097370604
Celebrity: Karlie Kloss : 0.41898103557893546
Celebrity: Bad Bunny : 0.45082163197606034
Celebrity: Jerry Lorenzo : 0.463886420640746 75
Celebrity: Adriene Mishler : 0.46593455230914305
Celebrity: Zoe Saldana : 0.5068051316467111
Celebrity: Beyonce : 0.5086659852489535
Celebrity: Ninjas Hyper : 0.5512348689358116
Celebrity: Pharrell Williams : 0.6755445546489346
Celebrity: Chinae Alexander : 0.7722872405273942
```

Tfidf + GloVe with cosine similarity

Get the pre-trained GloVe model from Stanford, at: https://figshare.com/articles/dataset/Twitter_pre-trained_word_vectors/11640300

```
# reading Glove word embeddings into a dictionary with "word" as key and values as word vectors

embeddings_index = dict()

with open("glove.twitter.27B.25d.txt") as file:
    for line in file:
```

```
            values = line.split()
            word = values[0]
            coefs = np.asarray(values[1:], dtype='float32')
            embeddings_index[word] = coefs
```

```
len(embeddings_index)
```

    1193515

```
# creating embedding matrix, every row is a vector representation from the vocabulary indexed by the tokenizer index.

embedding_matrix=np.zeros((vocab_size,len(embeddings_index["key"])))

for word,i in tokenizer.word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

```
# tf-idf vectors do not keep the original sequence of words, converting them into actual word sequences from the docu

document_embeddings=np.zeros((len(tokenized_paded_documents),len(embeddings_index["key"])))
words=tfidfvectoriser.get_feature_names()

for i in range(documents_df.shape[0]):
    for j in range(len(words)):
        document_embeddings[i]+=embedding_matrix[tokenizer.word_index[words[j]]]*tfidf_vectors[i][j]

document_embeddings=document_embeddings/np.sum(tfidf_vectors,axis=1).reshape(-1,1)
```

```
document_embeddings.shape
```

    (20, 25)

```
pairwise_similarities3=cosine_similarity(document_embeddings)
pairwise_differences3=euclidean_distances(document_embeddings)
```

```
most_similar(5,pairwise_similarities3,'Cosine  Similarity')
```

```
Document: documents    cute someone fix neck hey thinker great post t...
Name: BlackPink, dtype: object


Similar Documents:
Celebrity: BTS : 0.9992144061559247
Celebrity: iZone : 0.999162118516264
Celebrity: GFriend : 0.998842325061602
Celebrity: Solar : 0.9986440276534759
Celebrity: NCT : 0.9985136762136441
Celebrity: Seolhyun : 0.9981831196191092
Celebrity: Yara Shahidi : 0.9970586743633253
Celebrity: Ally Love : 0.9965183439201347
Celebrity: Naeun Son : 0.9963755342975111
Celebrity: Karlie Kloss : 0.9962310736609273
Celebrity: Zoe Saldana : 0.9958797862946573
Celebrity: Beyonce : 0.9958603485781699
Celebrity: Kerwin Frost : 0.9948020906821515
Celebrity: Pharrell Williams : 0.9939547824785617
Celebrity: Bad Bunny : 0.9935617491317197
Celebrity: Jerry Lorenzo : 0.9922703052469238
Celebrity: Ninjas Hyper : 0.9907680906132905
Celebrity: Adriene Mishler : 0.9866160811824223
Celebrity: Chinae Alexander : 0.9760087902895899
```

```
most_similar(5,pairwise_differences3,'Euclidean  Distance')
```

```
Document: documents    cute someone fix neck hey thinker great post t...
Name: BlackPink, dtype: object


Similar Documents:
Celebrity: BTS : 0.142910848643079З2
Celebrity: iZone : 0.2002425622602317
Celebrity: Solar : 0.2036309064959097
Celebrity: Seolhyun : 0.231878440042З2283
Celebrity: GFriend : 0.2849569934165974
Celebrity: NCT : 0.3063777061323183
Celebrity: Karlie Kloss : 0.31574941252349015
```

```
Celebrity: Naeun Son : 0.32808482199613304
Celebrity: Zoe Saldana : 0.329095613985837 86
Celebrity: Beyonce : 0.32977087445681447
Celebrity: Yara Shahidi : 0.3395009687275297
Celebrity: Pharrell Williams : 0.397184139494182 63
Celebrity: Bad Bunny : 0.4104657571682252
Celebrity: Kerwin Frost : 0.45248519233343 48
Celebrity: Ally Love : 0.4591948121131957
Celebrity: Jerry Lorenzo : 0.4646894216830501
Celebrity: Ninjas Hyper : 0.49237817307786336
Celebrity: Chinae Alexander : 0.7930915432612369
Celebrity: Adriene Mishler : 0.8375021246891716
```

Doc2vec with cosine similarity

```
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
from nltk.tokenize import word_tokenize
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
tagged_data = [TaggedDocument(words=word_tokenize(doc), tags=[i]) for i, doc in enumerate(documents_df.documents)]
```

```
model_d2v = Doc2Vec(vector_size=100,alpha=0.025, min_count=1)

model_d2v.build_vocab(tagged_data)

for epoch in range(100):
    model_d2v.train(tagged_data,
                    total_examples=model_d2v.corpus_count,
                    epochs=model_d2v.epochs)


document_embeddings=np.zeros((documents_df.shape[0],100))
```

```
for  i  in  range(len(document_embeddings)):
        document_embeddings[i]=model_d2v.docvecs[i]


pairwise_similarities4=cosine_similarity(document_embeddings)
pairwise_differences4=euclidean_distances(document_embeddings)


most_similar(5,pairwise_similarities4,'Cosine  Similarity')
```

```
Document: documents    cute someone fix neck hey thinker great post t...
Name: BlackPink, dtype: object


Similar Documents:
Celebrity: Adriene Mishler : 0.8376071862592145
Celebrity: BTS : 0.8256581802851971
Celebrity: Naeun Son : 0.8236491138937424
Celebrity: Beyonce : 0.8222327839117646
Celebrity: Zoe Saldana : 0.82131610607439
Celebrity: iZone : 0.8165864674589224
Celebrity: GFriend : 0.8105729245720346
Celebrity: Bad Bunny : 0.8105287756809676
Celebrity: Seolhyun : 0.8071866233165885
Celebrity: Ally Love : 0.8051993891879464
Celebrity: NCT : 0.8050990936836503
Celebrity: Solar : 0.7999428197780517
Celebrity: Jerry Lorenzo : 0.7973557128423785
Celebrity: Chinae Alexander : 0.7953199655656029
Celebrity: Ninjas Hyper : 0.793487738385525
Celebrity: Karlie Kloss : 0.7823220022834693
Celebrity: Pharrell Williams : 0.7686005025203629
Celebrity: Kerwin Frost : 0.7556117487774464
Celebrity: Yara Shahidi : 0.723379568443592
```

```
most_similar(5,pairwise_differences4,'Euclidean  Distance')
```

```
Document: documents    cute someone fix neck hey thinker great post t...
Name: BlackPink, dtype: object
```

```
Similar Documents:
Celebrity: BTS : 21.39257340710193
Celebrity: Naeun Son : 21.447466454159642
Celebrity: Adriene Mishler : 21.603179601893878
Celebrity: iZone : 21.672985311247892
Celebrity: GFriend : 22.003049112428
Celebrity: Solar : 22.935978706395385
Celebrity: NCT : 23.155770148868058
Celebrity: Seolhyun : 23.608177903324254
Celebrity: Bad Bunny : 23.74946985491297
Celebrity: Ally Love : 25.32843335948801
Celebrity: Zoe Saldana : 25.840228137967166
Celebrity: Beyonce : 25.869150976746248
Celebrity: Jerry Lorenzo : 26.48962565900195
Celebrity: Karlie Kloss : 27.63562020871692
Celebrity: Pharrell Williams : 27.71267874221189
Celebrity: Chinae Alexander : 28.228563854666252
Celebrity: Ninjas Hyper : 29.572249980144917
Celebrity: Kerwin Frost : 30.981244947591783
Celebrity: Yara Shahidi : 34.289751742673715
```

BERT model for sentence embedding

```
from sentence_transformers import SentenceTransformer
```

Pre-trained model: https://www.sbert.net/docs/pretrained_models.html

Paraphrase Identification

The following models are recommended for various applications, as they were trained on Millions of paraphrase examples. They create extremely good results for various similarity and retrieval tasks. They are currently under development, better versions and more details will be released in future. But they many tasks they work better than the NLI / STSb models.

paraphrase-distilroberta-base-v1 - Trained on large scale paraphrase data.

paraphrase-xlm-r-multilingual-v1 - Multilingual version of

paraphrase-distilroberta-base-v1, trained on parallel data for 50+ languages.

(Teacher: paraphrase-distilroberta-base-v1, Student: xlm-r-base)

Semantic Textual Similarity

The following models were optimized for Semantic Textual Similarity (STS). They were trained on SNLI+MultiNLI and then fine-tuned on the STS benchmark train set.

The best available models for STS are:

stsb-roberta-large - STSb performance: 86.39

stsb-roberta-base - STSb performance: 85.44

stsb-bert-large - STSb performance: 85.29

stsb-distilbert-base - STSb performance: 85.16

» Full List of STS Models

Duplicate Questions Detection

The following models were trained for duplicate questions mining and duplicate questions retrieval. You can use them to detect duplicate questions in a large corpus (see paraphrase mining) or to search for similar questions (see semantic search).

Available models:

quora-distilbert-base - Model first tuned on NLI+STSb data, then fine-tune* for Quora Duplicate Questions detection retrieval.

quora-distilbert-multilingual - Multilingual version of quora-distilbert-base. Fine-tuned with parallel data for 50+ languages.

Question-Answer Retrieval - MSMARCO

The following models were trained on MSMARCO Passage Ranking, a dataset with 500k real queries from Bing search. Given a search query, find the relevant passages.

msmarco-distilbert-base-v3: MRR@10: 33.13 on MS MARCO dev set

msmarco-roberta-base-ance-fristp: MRR@10: 33.03 on MS MARCO dev set

```
sbert_model = SentenceTransformer('stsb-roberta-large')
```

100% 1.31G/1.31G [01:04<00:00, 20.5MB/s]

```
document_embeddings = sbert_model.encode(documents_df['documents'])
```

```
document_embeddings
```

```
array([[-0.53672266,  0.54417115, -1.4026839 , ...,  1.3140677 ,
        -1.4579101 ,  0.34982672],
       [ 0.62302977,  0.47568327,  0.34358895, ...,  0.43715477,
        -2.7842007 ,  0.6610758 ],
       [-0.6929906 ,  0.11186998, -1.0118369 , ...,  0.83492076,
        -2.1865032 ,  1.015724  ],
       ...,
       [-0.14839004, -0.03186914,  0.3927361 , ...,  0.12975009,
        -1.9353865 ,  0.27613652],
       [-0.33167017,  0.51998717,  0.18267697, ...,  0.71633   ,
        -0.4115541 ,  0.59306455],
       [ 0.37954757,  1.4980035 ,  0.32582128, ...,  0.45599562,
        -1.8093145 ,  0.52114165]], dtype=float32)
```

```
pairwise_similarities5=cosine_similarity(document_embeddings)
pairwise_differences5=euclidean_distances(document_embeddings)
```

```
most_similar(5,pairwise_similarities5,'Cosine Similarity')
```

```
Document: documents    cute someone fix neck hey thinker great post t...
Name: BlackPink, dtype: object


Similar Documents:
Celebrity: Pharrell Williams : 0.7617722
Celebrity: Karlie Kloss : 0.6797999
Celebrity: Solar : 0.647331
Celebrity: NCT : 0.6314053
Celebrity: GFriend : 0.6263839
```

```
Celebrity: Ally Love : 0.61949587
Celebrity: Zoe Saldana : 0.6169745
Celebrity: Beyonce : 0.6169745
Celebrity: Bad Bunny : 0.61651134
Celebrity: Yara Shahidi : 0.60371923
Celebrity: Ninjas Hyper : 0.5982954
Celebrity: Chinae Alexander : 0.5609665
Celebrity: Kerwin Frost : 0.54631805
Celebrity: BTS : 0.5378001
Celebrity: Jerry Lorenzo : 0.5205925
Celebrity: iZone : 0.48225877
Celebrity: Adriene Mishler : 0.46840653
Celebrity: Naeun Son : 0.4391661
Celebrity: Seolhyun : 0.42663768
```

```
most_similar(5,pairwise_differences5,'Euclidean  Distance')
```

```
Document: documents    cute someone fix neck hey thinker great post t...
Name: BlackPink, dtype: object


Similar Documents:
Celebrity: Beyonce : 7.298213
Celebrity: Zoe Saldana : 7.298213
Celebrity: Pharrell Williams : 8.108789
Celebrity: Naeun Son : 8.260986
Celebrity: NCT : 8.378961
Celebrity: Jerry Lorenzo : 8.41909
Celebrity: Bad Bunny : 8.596975
Celebrity: Yara Shahidi : 8.866427
Celebrity: Chinae Alexander : 8.925954
Celebrity: Ally Love : 9.07926
Celebrity: Solar : 9.235973
Celebrity: iZone : 9.410283
Celebrity: Karlie Kloss : 9.502069
Celebrity: BTS : 9.900484
Celebrity: Adriene Mishler : 9.963263
Celebrity: GFriend : 10.007881
Celebrity: Seolhyun : 10.230555
Celebrity: Kerwin Frost : 10.2994995
Celebrity: Ninjas Hyper : 10.696829
```

✓ 0s completed at 9:27 PM