In [240]:
```
import os
import pandas as pd
import numpy as np
import re
import string
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
nltk.download('wordnet')
from autocorrect import Speller
spell = Speller(lang='en')
import warnings
warnings.filterwarnings("ignore")
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Embedding
from scipy.spatial import distance
from sklearn.metrics.pairwise import cosine_similarity
import spacy
import numpy as np
from numpy import dot
from numpy.linalg import norm
```

```
[nltk_data] Error loading stopwords: <urlopen error [Errno 8] nodename
[nltk_data]     nor servname provided, or not known>
[nltk_data] Error loading wordnet: <urlopen error [Errno 8] nodename
[nltk_data]     nor servname provided, or not known>
```

In [241]:
```
company_desc=pd.read_excel('/Users/sandeeppydi/Desktop/nlp/Company Descriptions.xlsx')
```

In [242]:
```
Ind_segments=pd.read_excel('/Users/sandeeppydi/Desktop/nlp/Industry Segments.xlsx')
```

In [243]:
```
Ind_segments=Ind_segments.iloc[:27,:]
```

In [244]:
```
#Data Preprocessing of company_desc
#following standard steps for preprocessing
#Lower casing
#Removal of Punctuations
#Removal of Numbers
#Removal of Stopwords
#Lemmatization
#Removal of URLs
#Removal of HTML tags
#removing white spaces
#stripping
#sorting sentence
```

In [245]:
```
#going to work on short desription
data=company_desc.company_short_description
```

In [246]:
```
#Lower casing
data=data.str.lower()
```

In [247]:
```
#Removal of Punctuations
def punct(s):
    return s.translate(str.maketrans('','',string.punctuation))
```

In [248]:
```
data=data.apply(punct)
```

In [249]:
```
#Special case for double quote characters not removed from above functi
on
def punct_(s):
    return re.sub('[“”]','',s)
```

In [250]:
```
data=data.apply(punct_)
```

In [251]:
```
#Removal of Numbers
```

```
def numbers_remove(s):
    return re.sub('\d+','',s)
```

In [252]:
```
data=data.apply(numbers_remove)
```

In [253]:
```
#Removal of Stopwords
#remove stop words
stop=list(stopwords.words('english'))
def remove_stopwords(x):
    l=x.split(' ')
    m=[]
    for i in l:
        if i not in stop:
            m.append(i)
    return ' '.join(m)
```

In [254]:
```
data=data.apply(remove_stopwords)
```

In [255]:
```
#lemitization
w=WordNetLemmatizer()
def lemitization(x):
    m=[]
    l=re.split(' ',x)
    for i in l:
        m.append(w.lemmatize(i))
    return ' '.join(m)
```

In [256]:
```
data=data.apply(lemitization)
```

In [257]:
```
#remove urls
def remove_url(x):
    return re.sub('https?\S*\s*','',x)
```

In [258]:
```
data=data.apply(remove_url)
```

In [259]:
```
#remove html tags
```

```
def remove_html(x):
    return re.sub('<.*?>','',x)
```

In [260]:
```
data=data.apply(remove_html)
```

In [261]:
```
##removing white spaces
def remove_white_spaces(x):
    s=re.split('\s+',x)
    return ' '.join(s)
```

In [262]:
```
data=data.apply(remove_white_spaces)
```

In [263]:
```
#strip
def strip(s):
    return s.strip()
```

In [264]:
```
data=data.apply(strip)
```

In [265]:
```
tokenizer = Tokenizer(num_words=27258,oov_token='<UNK>')
tokenizer.fit_on_texts(data)
seqences = tokenizer.texts_to_sequences(data)
word_index = tokenizer.word_index
```

In [266]:
```
#convert seqences into list
out1=[]
for i in seqences:
    out1.append(list(i))
```

In [267]:
```
embeddings_index = {}
f = open(os.path.join('/Users/sandeeppydi/Desktop/glove','glove.6B.300
d.txt'))
for line in f:
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
```

```
    embeddings_index[word] = coefs
f.close()
```

In [268]:
```
embedding_matrix = np.zeros((27258,300))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
       # words not found in embedding index will be all-zeros.
        embedding_matrix[i] = embedding_vector
```

In [269]:
```
for i in range(len(out1)):
    for j in range(len(out1[i])):
        out1[i][j]=embedding_matrix[out1[i][j]]
```

In [270]:
```
#average word embeddings to convert sentense to vector
sent_vector1=[]
for i in out1:
    l=[]
    for j in i:
        l.append(j)
    sent_vector1.append(np.sum((l),axis=0))
```

In [271]:
```
for i in range(len(sent_vector1)):
    sent_vector1[i]=sent_vector1[i].tolist()
```

In [272]:
```
#data preprocessing for Ind_segments
data_tags=Ind_segments.Tags
```

In [273]:
```
#following standard steps for preprocessing
#Lower casing
#Removal of Punctuations
#Removal of Numbers
#Removal of Stopwords
#Lemmatization
#Removal of URLs
#Removal of HTML tags
```

```
#removing white spaces
#stripping
```

In [274]:
```
data_tags=data_tags.str.lower()
```

In [275]:
```
#Removal of Punctuations
data_tags=data_tags.apply(punct)
data_tags=data_tags.apply(punct_)
```

In [276]:
```
data_tags=data_tags.apply(numbers_remove)
data_tags=data_tags.apply(remove_stopwords)
#data_tags=data_tags.apply(stemming)
data_tags=data_tags.apply(lemitization)
data_tags=data_tags.apply(remove_url)
data_tags=data_tags.apply(remove_html)
data_tags=data_tags.apply(remove_white_spaces)
data_tags=data_tags.apply(strip)
```

In [277]:
```
#Vector Creation
tokenizer1 = Tokenizer(num_words=120,oov_token='<UNK>')
tokenizer1.fit_on_texts(data_tags)
seqences1 = tokenizer1.texts_to_sequences(data_tags)
word_index1 = tokenizer1.word_index
```

In [278]:
```
#convert seqences into list
out2=[]
for i in seqences1:
    out2.append(list(i))
```

In [279]:
```
embedding_matrix1 = np.zeros((120,300))
for word, i in word_index1.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        # words not found in embedding index will be all-zeros.
        embedding_matrix1[i] = embedding_vector
```

In [280]:
```
for i in range(len(out2)):
    for j in range(len(out2[i])):
        out2[i][j]=embedding_matrix[out2[i][j]]
```

In [281]:
```
#average word embeddings to convert sentense to vector
sent_vector2=[]
for i in out2:
    l=[]
    for j in i:
        l.append(j)
    sent_vector2.append(np.sum((l),axis=0))
```

In [282]:
```
for i in range(len(sent_vector2)):
    sent_vector2[i]=sent_vector2[i].tolist()
```

In [283]:
```
def cos_similarity(a,b):
    return 1-distance.cosine(i,j)
```

In [284]:
```
#clasification based on cosine similarity
final_list=[]
for i in sent_vector1:
    val=-3
    for j in sent_vector2:
        if ((cos_similarity(i,j))>val):
            val=cos_similarity(i,j)
            k=j
    final_list.append(Ind_segments["Industry segment"][sent_vector2.ind
ex(k)])
```

In [285]:
```
company_desc["classification"]=pd.Series(final_list)
```

In [286]:
```
company_desc.to_excel("prog_3_a.xlsx")
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: