In [32]:
```
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] Downloading package stopwords to
[nltk_data]     /Users/sandeeppydi/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/sandeeppydi/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

In [33]:
```
company_desc=pd.read_excel('/Users/sandeeppydi/Desktop/nlp/Company Desc
riptions.xlsx')
```

```
In [34]: Ind_segments=pd.read_excel('/Users/sandeeppydi/Desktop/nlp/Industry Seg
         ments.xlsx')
```

```
In [35]: Ind_segments=Ind_segments.iloc[:27,:]
```

```
In [36]: #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 [37]: #going to work on short desription
         data=company_desc.company_short_description
```

```
In [38]: #Lower casing
         data=data.str.lower()
```

```
In [39]: #Removal of Punctuations
         def punct(s):
             return s.translate(str.maketrans('','',string.punctuation))
```

```
In [40]: data=data.apply(punct)
```

```
In [41]: #Special case for double quote characters not removed from above functi
         on
         def punct_(s):
             return re.sub('[“”]','',s)
```

```
In [42]: data=data.apply(punct_)
```

In [43]:
```
#Removal of Numbers
def numbers_remove(s):
    return re.sub('\d+','',s)
```

In [44]:
```
data=data.apply(numbers_remove)
```

In [45]:
```
#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 [46]:
```
data=data.apply(remove_stopwords)
```

In [47]:
```
#lemitization
w=WordNetLemmatizer()
def lemitization(x):
    m=[]
    l=re.split(' ',x)
    for i in l:
        m.append(w.lemmatize(i))
    return ' '.join(m)
```

In [48]:
```
data=data.apply(lemitization)
```

In [49]:
```
#remove urls
def remove_url(x):
    return re.sub('https?\S*\s*','',x)
```

In [50]:
```
data=data.apply(remove_url)
```

In [51]:
```
#remove html tags
def remove_html(x):
    return re.sub('<.*?>','',x)
```

In [52]:
```
data=data.apply(remove_html)
```

In [53]:
```
##removing white spaces
def remove_white_spaces(x):
    s=re.split('\s+',x)
    return ' '.join(s)
```

In [54]:
```
data=data.apply(remove_white_spaces)
```

In [55]:
```
#strip
def strip(s):
    return s.strip()
```

In [56]:
```
data=data.apply(strip)
```

In [57]:
```
for i in range(len(data)):
    data[i]=list(data[i].split(' '))
```

In [58]:
```
#data preprocessing for Ind_segments
data_tags=Ind_segments.Tags
```

In [59]:
```
#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 [60]:
```
data_tags=data_tags.str.lower()
```

In [61]:
```
#Removal of Punctuations
data_tags=data_tags.apply(punct)
data_tags=data_tags.apply(punct_)
```

In [62]:
```
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 [63]:
```
for i in range(len(data_tags)):
    data_tags[i]=list(data_tags[i].split(' '))
```

In [80]:
```
out1=[]
for i in data:
    out1.append(list(i))
```

In [82]:
```
out2=[]
for i in data_tags:
    out2.append(list(i))
```

In [84]:
```
def jacc_similar(list1, list2):
    s1 = set(list1)
    s2 = set(list2)
    return len(s1.intersection(s2)) / len(s1.union(s2))
```

In [85]:
```
#clasification based on JACCARD SIMILARITY similarity
final_list=[]
```

```
for i in out1:
    val=-3333
    for j in out2:
        if ((jacc_similar(i,j))>val):
            val=jacc_similar(i,j)
            k=j
    final_list.append(Ind_segments["Industry segment"][out2.index(k)])
```

In [86]:
```
company_desc["classification"]=pd.Series(final_list)
```

In [87]:
```
company_desc.to_excel("prog_3.xlsx")
```

In [ ]: