In [1]:

```
import pandas as pd
```

In [2]:

```
df = pd.read_csv('resume.csv')
```

In [3]:

```
df.head()
```

Out[3]:

| | ID | Resume_str | Resume_html | Category | Unnamed: 4 | Unnamed: 5 | Unnamed: 6 | Unn |
|---|---|---|---|---|---|---|---|---|
| **0** | 16852973 | HR ADMINISTRATOR/MARKETING ASSOCIATE\... | <div class="fontsize fontface vmargins hmargin... | HR | NaN | NaN | NaN | |
| **1** | 22323967 | HR SPECIALIST, US HR OPERATIONS ... | <div class="fontsize fontface vmargins hmargin... | HR | NaN | NaN | NaN | |
| **2** | 33176873 | HR DIRECTOR Summary Over 2... | <div class="fontsize fontface vmargins hmargin... | HR | NaN | NaN | NaN | |
| **3** | 27018550 | HR SPECIALIST Summary Dedica... | <div class="fontsize fontface vmargins hmargin... | HR | NaN | NaN | NaN | |
| **4** | 17812897 | HR MANAGER Skill Highlights ... | <div class="fontsize fontface vmargins hmargin... | HR | NaN | NaN | NaN | |

5 rows × 169 columns

In [4]:

```
df.tail()
```

Out[4]:

| | ID | Resume_str | Resume_html | Category | Unnamed: 4 | Unnamed: 5 | Unnamed: 6 | Unnamed: 7 |
|---|---|---|---|---|---|---|---|---|
| **3441** | 99416532 | RANK: SGT/E-5 NON-COMMISSIONED OFFIC... | <div class="fontsize fontface vmargins hmargin... | AVIATION | NaN | NaN | NaN | NaN |
| **3442** | 24589765 | GOVERNMENT RELATIONS, COMMUNICATIONS ... | <div class="fontsize fontface vmargins hmargin... | AVIATION | NaN | NaN | NaN | NaN |
| **3443** | 31605080 | GEEK SQUAD AGENT Professional... | <div class="fontsize fontface vmargins hmargin... | AVIATION | NaN | NaN | NaN | NaN |
| **3444** | 21190805 | PROGRAM DIRECTOR / OFFICE MANAGER ... | <div class="fontsize fontface vmargins hmargin... | AVIATION | NaN | NaN | NaN | NaN |
| **3445** | 37473139 | STOREKEEPER II Professional Sum... | <div class="fontsize fontface vmargins hmargin... | AVIATION | NaN | NaN | NaN | NaN |

5 rows × 169 columns

In [5]:

```
df.shape
```

Out[5]:

```
(3446, 169)
```

In [6]:

```
df.columns
```

Out[6]:

```
Index(['ID', 'Resume_str', 'Resume_html', 'Category', 'Unnamed: 4',
       'Unnamed: 5', 'Unnamed: 6', 'Unnamed: 7', 'Unnamed: 8', 'Unnamed: 9',
       ...
       'Unnamed: 159', 'Unnamed: 160', 'Unnamed: 161', 'Unnamed: 162',
       'Unnamed: 163', 'Unnamed: 164', 'Unnamed: 165', 'Unnamed: 166',
       'Unnamed: 167', 'Unnamed: 168'],
      dtype='object', length=169)
```

In [7]:

```
df = df[['Resume_str', 'Category']]
```

In [8]:

```
df.duplicated().sum()
```

Out[8]:

```
846
```

In [9]:

```
df = df.drop_duplicates()
```

In [10]:

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

Out[10]:

```
Resume_str     1
Category      92
dtype: int64
```

In [11]:

```
df = df.dropna()
```

In [12]:

```
df.shape
```

Out[12]:

```
(2508, 2)
```

In [13]:

```
df.columns
```

Out[13]:

```
Index(['Resume_str', 'Category'], dtype='object')
```

In [14]:

```
df
```

Out[14]:

| | Resume_str | Category |
|---|---|---|
| **0** | HR ADMINISTRATOR/MARKETING ASSOCIATE\... | HR |
| **1** | HR SPECIALIST, US HR OPERATIONS ... | HR |
| **2** | HR DIRECTOR Summary Over 2... | HR |
| **3** | HR SPECIALIST Summary Dedica... | HR |
| **4** | HR MANAGER Skill Highlights ... | HR |
| **...** | ... | ... |
| **3440** | ADVANCED LEVEL WHEELED VEHICLE MECHAN... | AVIATION |
| **3441** | RANK: SGT/E-5 NON- COMMISSIONED OFFIC... | AVIATION |
| **3442** | GOVERNMENT RELATIONS, COMMUNICATIONS ... | AVIATION |
| **3443** | GEEK SQUAD AGENT Professional... | AVIATION |
| **3444** | PROGRAM DIRECTOR / OFFICE MANAGER ... | AVIATION |

2508 rows × 2 columns

In [15]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2508 entries, 0 to 3444
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Resume_str  2508 non-null   object
 1   Category    2508 non-null   object
dtypes: object(2)
memory usage: 58.8+ KB
```

In [16]:

```
df['Category'].unique()
```

```
520JCIT4"" itemprop=""addressLocality""> City</span> <span class=""joblocation"">
',
       ' Idaho', ' Billings',
       ' General Cleanup. </li> <li>  1990- 1992. </li> <li>  Reading Plans',
       " and equipment for the products we were designing.</li> <li> Customer Focu
s:  Mission was to exceed the customers' expectations within capabilities.</li> <li
> Made sure that the extra steps were taken to insure our customers that Thermasys
was their best choice.</li> <li> Regularly visited with the Technical staff at the
customer locations in an effort to insure viability of the design.</li> <li> Afterm
arket:  Designed",
       ' Microsoft Project</li> <li> Field Operations / Inspections</li> <li> Troub
leshooting/ Problem Solver</li> <li> Quality Control / Inspection</li> </ul> <ul> <
li> Team Building / Leadership</li> <li> Project Coordination / Development</li> <l
i> OSHA Regulation / Project Safety</li> <li> Cost / Profit Analysis</li> <li> Civi
l Engineering</li> <li> Project Planning and Development</li> <li> Project supervis
ion</li> </ul> </div> </div> </div> <div class=""section"" id=""SECTION_WRKH92a9e8e
6-e2d1-4f49-b957-456f264ee0f1"" style=""padding-top:0px;""> <div class=""heading"">
<div class=""sectiontitle"" id=""SECTNAME_WRKH92a9e8e6-e2d1-4f49-b957-456f264ee0f
1""> Work History</div> </div> <div class=""paragraph firstparagraph"" id=""PARAGRA
PH_92a9e8e6-e2d1-4f49-b957-456f264ee0f1_1_592bf69f-1fa4-4042-9598-aee0d9494dd4"" it
```

In [17]:

```
df['Category'].value_counts()
```

Out[17]:

```
BUSINESS-DEVELOPMENT
119
INFORMATION-TECHNOLOGY
119
CHEF
117
FITNESS
117
ACCOUNTANT
116

...
 Business Development
1
 0); line-height: 12pt; font-family: ""Calibri""
1
sans-serif; font-size: 12pt; font-style: normal; font-weight: normal;'> <p style='col
or: rgb(0
1
 ethical and independent decision-making ability consistent with medical protocols. <
br/> <span class=""""> </span> </span> </span> </span> </span> </span> </div> <div> <
span class=""""> <span class=""""> <span class=""""> <span class=""""> <span class
=""""> <span class=""""> </span> </span> </span> </span> </span> </span> </div> <div>
<span class=""""> <span class=""""> <span class=""""> <span class=""""> <span class
=""""> <span class=""""> Disciplined      1
 and evaluate
1
Name: Category, Length: 75, dtype: int64
```

In [18]:

```
category_counts = df['Category'].value_counts()
```

In [19]:

```
if any(category_counts.between(1, 10)):
    values_to_drop = category_counts[category_counts.between(1, 10)].index.tolist()
    df = df[~df['Category'].isin(values_to_drop)]
```

In [21]:

```
df['Category'].value_counts()
```

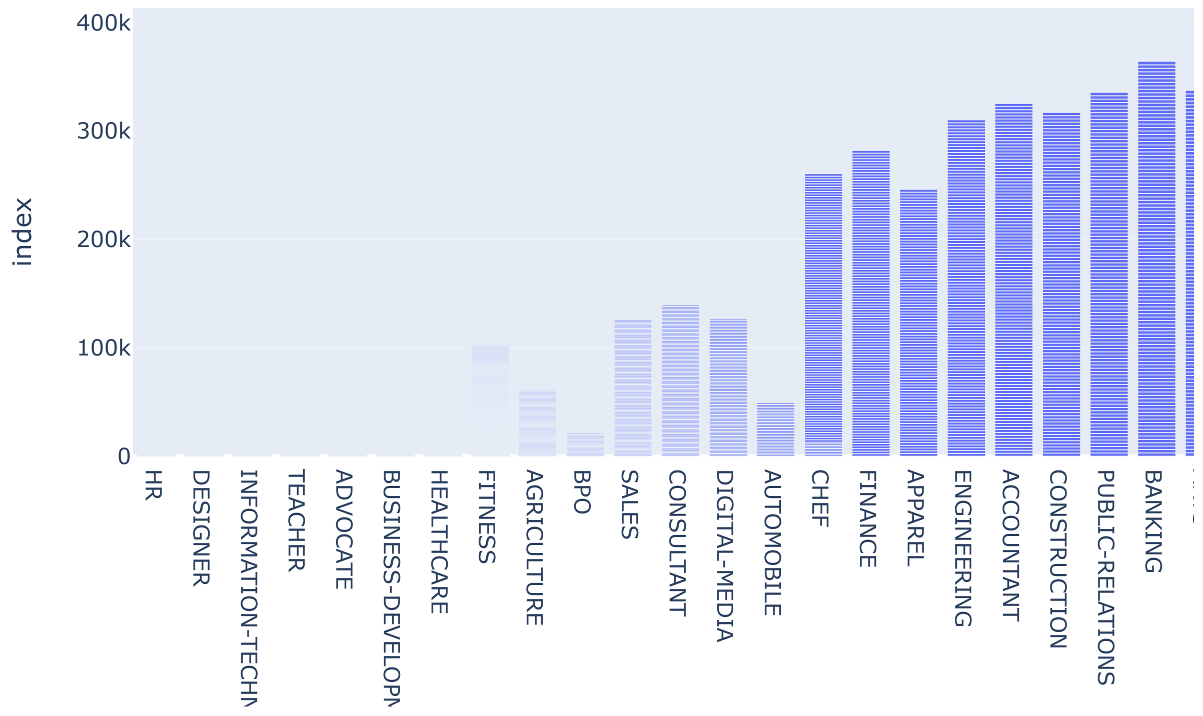
Out[21]:

```
INFORMATION-TECHNOLOGY    119
BUSINESS-DEVELOPMENT      119
CHEF                      117
FITNESS                   117
AVIATION                  116
ACCOUNTANT                116
ENGINEERING               116
BANKING                   115
ADVOCATE                  115
SALES                     115
CONSULTANT                115
FINANCE                   115
HEALTHCARE                113
PUBLIC-RELATIONS          110
HR                        109
CONSTRUCTION              108
DESIGNER                  106
ARTS                      103
TEACHER                   102
APPAREL                    96
DIGITAL-MEDIA              95
AGRICULTURE                63
AUTOMOBILE                 35
BPO                        21
Name: Category, dtype: int64
```

In [22]:

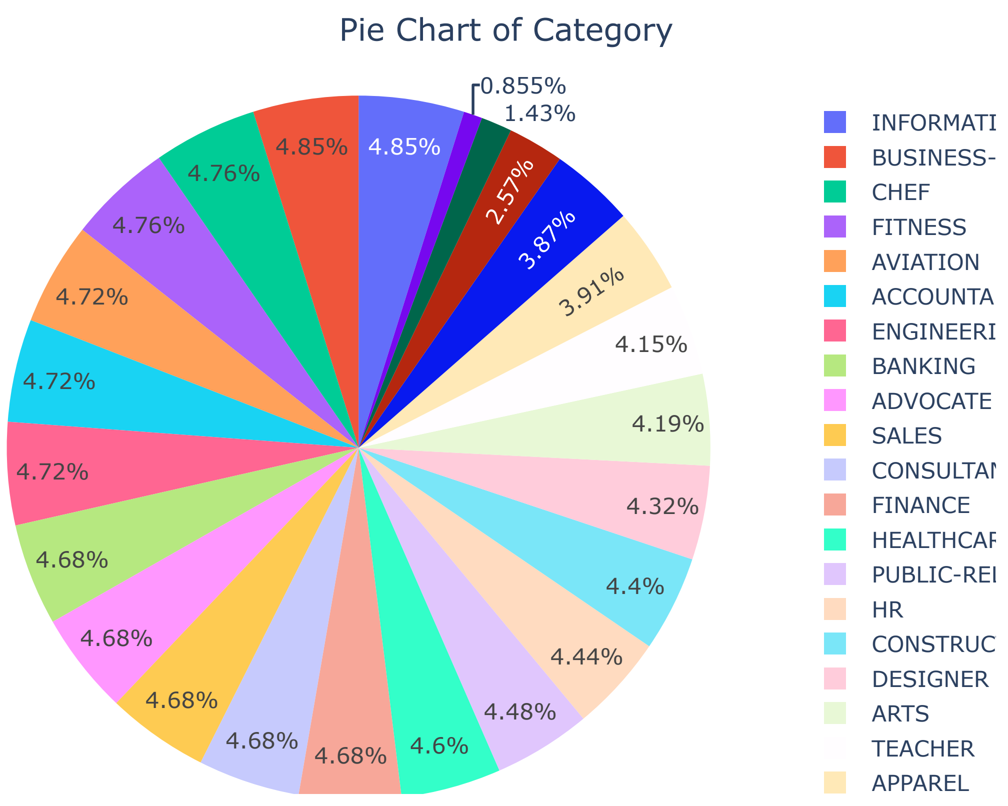
```
import plotly.express as px

fig = px.bar(df, x="Category", y= df.index)
fig.show()
```



In [23]:

```
value_counts = df['Category'].value_counts()
fig = px.pie(names=value_counts.index, values=value_counts.values)
fig.update_layout(
    title='Pie Chart of Category',
    title_x=0.5
)
fig.show()
```



In [24]:

```
df = df.rename(columns={'Resume_str': 'Resume'})
```

In [25]:

```
df_new = df.copy()
```

In [26]:

```
def clean_text(text):
    text = text.lower()
    return text.strip()
```

In [27]:

```
df_new['Resume'] = df_new['Resume'].apply(lambda x: clean_text(x))
```

In [28]:

```
def strip_text(text):
    text = text.strip()
    return text
```

In [29]:

```
df_new['Resume'] = df_new['Resume'].apply(lambda x: strip_text(x))
```

In [30]:

```
import string
string.punctuation
```

Out[30]:

```
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

In [31]:

```
def remove_punctuation(text):
    punctuationfree="".join([i for i in text if i not in string.punctuation])
    return punctuationfree
```

In [32]:

```
df_new['Resume']= df_new['Resume'].apply(lambda x:remove_punctuation(x))
```

In [33]:

```
import re
```

In [34]:

```
def tokenization(text):
    tokens = re.split('W+',text)
    return tokens
```

In [35]:

```
df_new['Resume']= df_new['Resume'].apply(lambda x: tokenization(x))
```

In [36]:

```
import nltk
stopwords = nltk.corpus.stopwords.words('english')
```

In [37]:

```
def remove_stopwords(text):
    output= " ".join(i for i in text if i not in stopwords)
    return output
```

In [38]:

```
df_new['Resume']= df_new['Resume'].apply(lambda x:remove_stopwords(x))
```

In [39]:

```
from nltk.stem import WordNetLemmatizer
wordnet_lemmatizer = WordNetLemmatizer()
```

In [40]:

```
def lemmatizer(text):
    lemm_text = "".join([wordnet_lemmatizer.lemmatize(word) for word in text])
    return lemm_text
```

In [41]:

```
df_new['Resume']=df_new['Resume'].apply(lambda x:lemmatizer(x))
```

In [42]:

```
def clean_text(text):
    text = re.sub('\[.*\]','', text).strip()
    text = re.sub('\S*\d\S*\s*','', text).strip()
    return text.strip()
```

In [44]:

```
df_new['Resume'] = df_new['Resume'].apply(lambda x: clean_text(x))
```

In [45]:

```
def remove_urls(vTEXT):
    vTEXT = re.sub(r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b', '', vTEXT, flags=re.MULTILINE)
    return(vTEXT)
```

In [46]:

```
df_new['Resume'] = df_new['Resume'].apply(lambda x: remove_urls(x))
```

In [47]:

```
def remove_digits(text):
    clean_text = re.sub(r"\b[0-9]+\b\s*", "", text)
    return(text)
```

In [48]:

```
df_new['Resume'] = df_new['Resume'].apply(lambda x: remove_digits(x))
```

In [49]:

```
def remove_emojis(data):
    emoji_pattern = re.compile("["
                               u"\U0001F600-\U0001F64F"  # emoticons
                               u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                               u"\U0001F680-\U0001F6FF"  # transport & map symbols
                               u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                               "]+", flags=re.UNICODE)
    return re.sub(emoji_pattern, '', data)
```

In [50]:

```
df_new['Resume'] = df_new['Resume'].apply(lambda x: remove_emojis(x))
```

In [54]:

```
df_new['Resume'] = df_new['Resume'].replace(r'\s+', ' ', regex=True)
```

In [55]:

```
df_new
```

Out[55]:

| | Resume | Category |
|---|---|---|
| **0** | hr administratormarketing associate hr adminis... | HR |
| **1** | hr specialist us hr operations summary versati... | HR |
| **2** | hr director summary over years experience in r... | HR |
| **3** | hr specialist summary dedicated driven and dyn... | HR |
| **4** | hr manager skill highlights hr skills hr depar... | HR |
| **...** | ... | ... |
| **3440** | advanced level wheeled vehicle mechanic career... | AVIATION |
| **3441** | rank non commissioned officer in charge brigad... | AVIATION |
| **3442** | government relations communications and organi... | AVIATION |
| **3443** | geek squad agent professional profile it suppo... | AVIATION |
| **3444** | program director office manager summary highly... | AVIATION |

2456 rows × 2 columns

In [57]:

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
le.fit(df_new['Category'])

df_new['category_encoded'] = le.transform(df_new['Category'])
df_new.head()
```

Out[57]:

| | Resume | Category | category_encoded |
|---|---|---|---|
| **0** | hr administratormarketing associate hr adminis... | HR | 19 |
| **1** | hr specialist us hr operations summary versati... | HR | 19 |
| **2** | hr director summary over years experience in r... | HR | 19 |
| **3** | hr specialist summary dedicated driven and dyn... | HR | 19 |
| **4** | hr manager skill highlights hr skills hr depar... | HR | 19 |

In [58]:

```
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

In [59]:

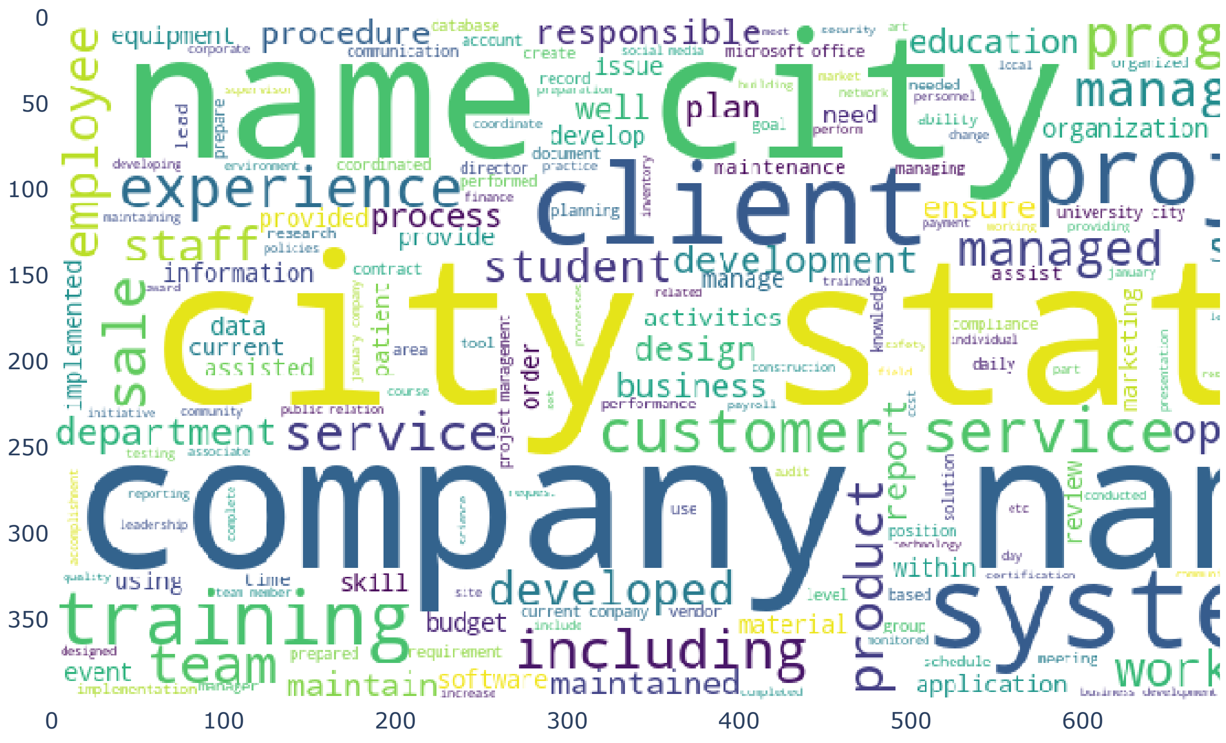
```
import numpy as np
```

In [62]:

```
import plotly.graph_objs as go
```

In [63]:

```
text = ' '.join(df_new['Resume'].astype(str))
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)
fig = go.Figure(go.Image(z=wordcloud.to_array()))
fig.update_layout(width=800, height=400, margin=dict(l=0, r=0, t=0, b=0))
fig.show()
```



In [64]:

```
import locale
locale.setlocale(locale.LC_ALL, 'en_US.utf-8')
```

Out[64]:

```
'en_US.utf-8'
```

In [65]:

```
import locale
print(locale.getpreferredencoding())
```

```
cp1252
```

In [66]:

```
import locale
def getpreferredencoding(do_setlocale = True):
    return "UTF-8"
locale.getpreferredencoding = getpreferredencoding
```

In [67]:

```
import transformers
import torch
import torch.nn as nn
import torch.optim as optim
```

In [69]:

```
x = df_new['Resume']
y = df_new['category_encoded']

print(len(x), len(y))
```

```
2456 2456
```

In [84]:

```
rn.model_selection import train_test_split

_test, y_train, y_test = train_test_split(x, y, shuffle = True, stratify = df_new['category_encoded'
                                          random_state=42)
x_train), len(y_train))
x_test), len(y_test))
```

```
1842 1842
614 614
```

In [85]:

```
from sklearn.feature_extraction.text import CountVectorizer

vect = CountVectorizer()
vect.fit(x_train)
```

Out[85]:

```
▾ CountVectorizer
CountVectorizer()
```

In [86]:

```
x_train_dtm = vect.transform(x_train)
x_test_dtm = vect.transform(x_test)
```

In [87]:

```
vect_tunned = CountVectorizer(stop_words='english', ngram_range=(1,2), min_df=0.1, max_df=0.7, max_f
```

In [88]:

```
from sklearn.feature_extraction.text import TfidfTransformer

tfidf_transformer = TfidfTransformer()

tfidf_transformer.fit(x_train_dtm)
x_train_tfidf = tfidf_transformer.transform(x_train_dtm)

x_train_tfidf
```

Out[88]:

```
<1842x40980 sparse matrix of type '<class 'numpy.float64'>'
	with 630532 stored elements in Compressed Sparse Row format>
```

In [89]:

```
texts = df_new['Resume']
target = df_new['category_encoded']
```

In [90]:

```
from keras.preprocessing.text import Tokenizer
```

In [91]:

```
word_tokenizer = Tokenizer()
word_tokenizer.fit_on_texts(texts)

vocab_length = len(word_tokenizer.word_index) + 1
vocab_length
```

Out[91]:

```
48881
```

In [92]:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.sequence import pad_sequences
from nltk.tokenize import word_tokenize
```

In [93]:

```
def embed(corpus):
    return word_tokenizer.texts_to_sequences(corpus)

longest_train = max(texts, key=lambda sentence: len(word_tokenize(sentence)))
length_long_sentence = len(word_tokenize(longest_train))

train_padded_sentences = pad_sequences(
    embed(texts),
    length_long_sentence,
    padding='post'
)

train_padded_sentences
```

Out[93]:

```
array([[  188, 18203,   211, ...,     0,     0,     0],
       [  188,   224,   287, ...,     0,     0,     0],
       [  188,   170,   107, ...,     0,     0,     0],
       ...,
       [  445,   112,   221, ...,     0,     0,     0],
       [10298,  5139,  1366, ...,     0,     0,     0],
       [   62,   170,    39, ...,     0,     0,     0]])
```

In [94]:

```
embeddings_dictionary = dict()
embedding_dim = 100

# Load GloVe 100D embeddings
with open('glove.6B.100d.txt', encoding="utf8") as fp:
    for line in fp.readlines():
        records = line.split()
        word = records[0]
        vector_dimensions = np.asarray(records[1:], dtype='float32')
        embeddings_dictionary [word] = vector_dimensions
```

In [95]:

```
from sklearn.naive_bayes import MultinomialNB
nb = MultinomialNB()
nb.fit(x_train_dtm, y_train)
```

Out[95]:

```
▾ MultinomialNB
MultinomialNB()
```

In [96]:

```
y_pred_class = nb.predict(x_test_dtm)
y_pred_prob = nb.predict_proba(x_test_dtm)[:, 1]
```

In [97]:

```
from sklearn import metrics
print(metrics.accuracy_score(y_test, y_pred_class))
```

```
0.5570032573289903
```

In [99]:

```
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.pipeline import Pipeline

pipe = Pipeline([('bow', CountVectorizer()),
                 ('tfid', TfidfTransformer()),
                 ('model', MultinomialNB())])
```

In [100]:

```
pipe.fit(x_train, y_train)

y_pred_class = pipe.predict(x_test)

print(metrics.accuracy_score(y_test, y_pred_class))
```

```
0.5390879478827362
```

In [101]:

```
from sklearn.preprocessing import LabelEncoder
```

In [102]:

```
le = LabelEncoder()
y_encoded = le.fit_transform(y)
```

In [103]:

```
X_train, X_test, y_train, y_test = train_test_split(x, y_encoded, test_size=0.2, shuffle = True, str
                                                    random_state=42)
```

In [104]:

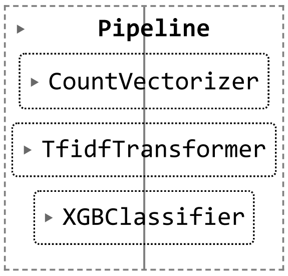
```
import xgboost as xgb

pipe = Pipeline([
    ('bow', CountVectorizer()),
    ('tfid', TfidfTransformer()),
    ('model', xgb.XGBClassifier(
        learning_rate=0.1,
        max_depth=7,
        n_estimators=80,
        use_label_encoder=False,
        eval_metric='auc',
    ))
])
```

In [105]:

```
pipe.fit(X_train, y_train)
```

Out[105]:



In [106]:

```
y_pred = pipe.predict(X_test)
```

In [108]:

```
from sklearn.metrics import accuracy_score
```

In [109]:

```
acc = accuracy_score(y_test, y_pred)
print('Test accuracy:', acc)
```

```
Test accuracy: 0.75
```