```
import gensim
import pandas as pd
from google.colab import files
uploaded = files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Music_Lyrics_cleaned.csv to Music_Lyrics_cleaned.csv
```

```
import pandas as pd
import numpy as np
import gensim
from gensim.models import Word2Vec
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import string
import nltk
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import os
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder

# Download NLTK data (stopwords and WordNet)
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
```

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

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

```
# Fil

print(df.head())
```

```
                                              lyrics       topic
0  hold time feel break feel untrue convince spea...     sadness
1  believe drop rain fall grow believe darkest ni...  world/life
2  sweetheart send letter goodbye secret feel bet...       music
3  kiss lips want stroll charm mambo chacha merin...    romantic
4  till darling till matter know till dream live ...    romantic
```

```
# running time 6 ms
import nltk
nltk.download('omw-1.4')

def preprocess_text(text):
    # Tokenization and lowercase
    tokens = nltk.word_tokenize(text.lower())

    # Remove stopwords and special characters
    stopwords_set = set(stopwords.words('english'))
    tokens = [word for word in tokens if word not in stopwords_set and word not in string.punctuation]

    # Lemmatization
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(word) for word in tokens]

    return tokens

# Apply text preprocessing to the lyrics column
df['lyrics_tokens'] = df['lyrics'].apply(preprocess_text)
df.to_csv('en_token.csv',index=False)
df.head()
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

	lyrics	topic	lyrics_tokens
0	hold time feel break feel untrue convince spea...	sadness	[hold, time, feel, break, feel, untrue, convin...
1	believe drop rain fall grow believe darkest ni...	world/life	[believe, drop, rain, fall, grow, believe, dar...
2	sweetheart send letter goodbye secret feel bet...	music	[sweetheart, send, letter, goodbye, secret, fe...
3	kiss lips want stroll charm mambo chacha merin...	romantic	[kiss, lip, want, stroll, charm, mambo, chacha...
4	till darling till matter know till dream live	romantic	[till, darling, till, matter, know, till, drea...

```
tokenized_lyrics = df['lyrics_tokens'].tolist()
print(tokenized_lyrics)
#  # Train Word2Vec model
model = Word2Vec(sentences=tokenized_lyrics, vector_size=100, window=5, min_count=5, sg=0)

# Save the trained Word2Vec model
model.save("custom_lyrics_word2vec.model")
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=1000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

```
model.wv.most_similar("love")
```

```
[('humdinger', 0.5926448106765747),
 ('sweetest', 0.5726975202560425),
 ('forevermore', 0.5386531352996826),
 ('absolutely', 0.532386302947998),
 ('naturally', 0.511110782623291),
 ('depend', 0.4923951327800751),
 ('sweeter', 0.48558276891708374),
 ('yeees', 0.48530063033103943),
 ('darling', 0.4811593294143677),
 ('mambo', 0.4796101450920105)]
```

```
from google.colab import files

# Download the model file to your local machine
files.download('custom_lyrics_word2vec.model')
```

```
# Load your sentiment dataset
data = pd.read_csv('Music_Lyrics_cleaned.csv')

# Preprocess the text data
data['lyrics'] = data['lyrics'].str.lower()
data['lyrics'] = data['lyrics'].str.replace('[^a-zA-Z\s]', '')
w2v_model=model
data.dropna(subset=['lyrics'], inplace=True)  # Remove rows with missing content
```

```
<ipython-input-32-7b7a3fd3e941>:6: FutureWarning: The default value of regex will change from True to False in a future version.
  data['lyrics'] = data['lyrics'].str.replace('[^a-zA-Z\s]', '')
```

```
# Encode sentiment labels to numeric values
label_encoder = LabelEncoder()
data['topic_encoded'] = label_encoder.fit_transform(data['topic'])

# Convert text data to document vectors using custom word embeddings
sentence_vectors = []
for sentence in data['lyrics']:
    words = sentence.split()
    vectors = [w2v_model.wv[word] for word in words if word in w2v_model.wv]
    if vectors:
        sentence_vector = sum(vectors) / len(vectors)
        sentence_vectors.append(sentence_vector)
    else:
        # Handle cases where there are no valid words in the sentence
        sentence_vectors.append([0] * w2v_model.vector_size)
```

```
X = sentence_vectors
y = data['topic_encoded']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)



# Train a machine learning model (Logistic Regression in this example)
model = LogisticRegression()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Provide sentiment analysis for new input using custom word embeddings
new_lyrics = ["Love me like you do"]
new_lyrics_words = new_lyrics[0].split()
new_lyrics_vectors = [w2v_model.wv[word] for word in new_lyrics_words if word in w2v_model.wv]
if new_lyrics_vectors:
    new_lyrics_vector = sum(new_lyrics_vectors) / len(new_lyrics_vectors)
else:
    new_lyrics_vector = [0] * w2v_model.vector_size

predicted_sentiment_encoded = model.predict([new_lyrics_vector])
predicted_sentiment = label_encoder.inverse_transform(predicted_sentiment_encoded)
print(f"Predicted Sentiment: {predicted_sentiment[0]}")
```

```
Accuracy: 0.9053744493392071
Predicted Sentiment: obscene
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

```
df.columns
```

```
Index(['ALink', 'SName', 'SLink', 'Lyric', 'language', 'lyrics_tokens'], dtype='object')
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.