In [22]:
```
#Load the libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import nltk
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelBinarizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize,sent_tokenize
import re,string,unicodedata
from nltk.tokenize.toktok import ToktokTokenizer
from nltk.stem import LancasterStemmer,WordNetLemmatizer
from sklearn.linear_model import LogisticRegression,SGDClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
import math
import warnings
warnings.filterwarnings('ignore')
import gensim
from gensim.utils import simple_preprocess
import nltk
from nltk.corpus import stopwords
import pandas as pd
import re
from wordcloud import WordCloud
import gensim.corpora as corpora
import pyLDAvis.gensim_models
import pickle
import pyLDAvis
import os
import random
```

In [23]:
```
import os

def read_texts(path):
    texts = []
    for filename in os.listdir(path):
        if filename.endswith(".en"):
            with open(os.path.join(path, filename), "r", encoding="utf-8") as f:
                texts.append(f.read())
    return " ".join(texts)
```

In [24]:
```
import nltk
from nltk import word_tokenize
from collections import defaultdict

nltk.download("punkt") # download the tokenizer
# nltk.download("europarl_raw") # download the dataset

text = read_texts('europarl_raw/english')
tokens = word_tokenize(text) # tokenize the text

def ngrams(tokens, n):
    ngrams = []
    for i in range(len(tokens) - n + 1):
        ngrams.append(tuple(tokens[i:i+n]))
    return ngrams

def train_ngram(tokens, n):
    ngram_counts = defaultdict(int)
    for ngram in ngrams(tokens, n):
        ngram_counts[ngram] += 1
    return ngram_counts

def perplexity(model, tokens, n):
    ngram_counts = model
    total_count = sum(ngram_counts.values())
    log_prob = 0
    num_ngrams = 0
    for i in range(len(tokens) - n + 1):
        ngram = tuple(tokens[i:i+n])
        count = ngram_counts[ngram] if ngram in ngram_counts else 0
        prob = count / total_count
        log_prob -= np.log2(prob)
        num_ngrams += 1
    return 2**(log_prob/num_ngrams)

def generate_sentence(model, n, seed_text, max_length):
    ngram_counts = model
    generated_text = seed_text
    while len(generated_text.split()) < max_length:
        ngrams_in_text = ngrams(word_tokenize(generated_text), n)
        try:
            last_ngram = ngrams_in_text[-1]
        except:
            continue
        next_word_probs = defaultdict(int)
        for ngram, count in ngram_counts.items():
            if ngram[:-1] == last_ngram:
                next_word_probs[ngram[-1]] = count
        if sum(next_word_probs.values()) == 0:
            break
        next_word = random.choices(list(next_word_probs.keys()), weights=list(next_word_probs.values()), k=1)[0]
        generated_text += " " + next_word
    return generated_text

# Train the 2-gram language model
bigram_model = train_ngram(tokens, 2)

# Evaluation of the model
perplexity = perplexity(bigram_model, tokens, 2) # calculate the perplexity
print(f"Perplexity for 2-gram language model: {perplexity:.2f}")
for n in [2, 3, 4, 5]:
    print(f"Generated sentence using {n}-gram model:")
    print(generate_sentence(bigram_model, n, "The European", 20))
    print()
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/farjad.ahmed/nltk_data...
[nltk_data]   Package punkt is already up-to-date!

Perplexity for 2-gram language model: 29784.34
Generated sentence using 2-gram model:
The European

Generated sentence using 3-gram model:
```

In [ ]:
```
# Train the 3-gram language model
trigram_model = train_ngram(tokens, 3)

# Evaluation of the model
perplexity = perplexity(trigram_model, tokens, 2) # calculate the perplexity
print(f"Perplexity for 2-gram language model: {perplexity:.2f}")
for n in [2, 3, 4, 5]:
    print(f"Generated sentence using {n}-gram model:")
    print(generate_sentence(trigram_model, n, "The European", 20))
    print()
```

In [ ]:
```
# def generate_sentence(model, n, seed_text, max_length):
#     ngram_counts = model
#     generated_text = seed_text
#     while len(generated_text.split()) < max_length:
#         ngrams_in_text = ngrams(word_tokenize(generated_text), n)
#         last_ngram = ngrams_in_text[-1]
#         next_word_probs = defaultdict(int)
#         for ngram, count in ngram_counts.items():
#             if ngram[:-1] == last_ngram:
#                 next_word_probs[ngram[-1]] = count
#         if next_word_probs:
#             next_word = max(next_word_probs, key=next_word_probs.get)
#         else:
#             break
#         generated_text += " " + next_word
#     return generated_text

# for n in [2, 3, 4, 5]:
#     print(f"Generated sentence using {n}-gram model:")
#     print(generate_sentence(ngram_model, n, "The European", 20))
#     print()
```

```
Generated sentence using 2-gram model:
The European

Generated sentence using 3-gram model:

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
/var/folders/t6/rk7lq7211555v4bhr2_wl3x00000gp/T/ipykernel_23599/319860404.py in <module>
     18 for n in [2, 3, 4, 5]:
     19     print(f"Generated sentence using {n}-gram model:")
---> 20     print(generate_sentence(ngram_model, n, "The European", 20))
     21     print()

/var/folders/t6/rk7lq7211555v4bhr2_wl3x00000gp/T/ipykernel_23599/319860404.py in generate_sentence(model, n, seed_te
xt, max_length)
      4     while len(generated_text.split()) < max_length:
      5         ngrams_in_text = ngrams(word_tokenize(generated_text), n)
----> 6         last_ngram = ngrams_in_text[-1]
      7         next_word_probs = defaultdict(int)
      8         for ngram, count in ngram_counts.items():

IndexError: list index out of range
```