```
from sklearn.feature_extraction.text import CountVectorizer
 corpus = [ 'jazz music has a swing rhythm', 'swing is hard to explain', 'swing rhythm is a natural rhythm']
vectorizer = CountVectorizer(binary=False)
X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names_out())
 print(X.toarray())


from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer(norm=None)
 X = vectorizer.fit_transform(corpus) print(vectorizer.get_feature_names_out()) print(X) absolute TF-IDF

from sklearn.feature_extraction.text import TfidfVectorizer
 vectorizer = TfidfVectorizer(norm='l1')
 X = vectorizer.fit_transform(corpus)
 print(vectorizer.get_feature_names_out())
print(X) normalized TF-IDF


from sklearn.feature_extraction.text import CountVectorizer corpus = [ 'the hotel has one bad room', 'the room of the hotel is bad', 'one bathroom is bad, the other bathroom is good' ]
vectorizer = CountVectorizer(binary=True)
 X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names_out())
 print(X.toarray())

 Let p word = Number of documents containing the term "word" /total number of documents
IDF word = log2 1 p(word)

from nltk import word_tokenize
 from nltk.corpus import stopwords
 from nltk.stem import PorterStemmer
 tokens = word_tokenize(text)
# Remove the punctuations and numbers
 tokens = [word for word in tokens if word.isalpha()]
 Lower the tokens
tokens = [word.lower() for word in tokens]
 Remove stopword
 tokens = [word for word in tokens if not word in stopwords.words("english")]
 Stem the tokens
 ps = PorterStemmer() tokens = [ps.stem(w) for w in tokens]
 text_cleaned = " ".join(tokens)
```