**import** nltk

**from** nltk **import** word\_tokenize, sent\_tokenize

**from** nltk **import** pos\_tag

**from** nltk.corpus **import** stopwords

**from** nltk.stem **import** PorterStemmer

**from** nltk.stem **import** WordNetLemmatizer

**from** sklearn.feature\_extraction.text **import** TfidfVectorizer

In [77]:

Out[77]:

True

Tokenization example

In [84]:

text **=** "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce commodo mauris id justo condimentum dignissim. Nullam placerat semper dapibus. Pellentesque ac risus nulla. Phasellus ut dapibus nunc, id aliquam dolor."

In [65]:

print(word\_tokenize(text))

['Lorem', 'ipsum', 'dolor', 'sit', 'amet', ',', 'consectetur', 'adipiscing', 'elit', '.', 'Fusce', 'commodo', 'mauris', 'id', 'justo', 'condimentum', 'dignissim', '.', 'Nullam', 'placerat', 'semper', 'dapibus', '.', 'Pellentesque', 'ac', 'risus', 'nulla', '.', 'Phasellus', 'ut', 'dapibus', 'nunc', ',', 'id', 'aliquam', 'dolor', '.']

In [66]:

print(sent\_tokenize(text))

['Lorem ipsum dolor sit amet, consectetur adipiscing elit.', 'Fusce commodo mauris id justo condimentum dignissim.', 'Nullam placerat semper dapibus.', 'Pellentesque ac risus nulla.', 'Phasellus ut dapibus nunc, id aliquam dolor.']

POS Tagging

In [67]:

to\_tag **=** word\_tokenize(text)

In [68]:

print(pos\_tag(to\_tag))

[('Lorem', 'NNP'), ('ipsum', 'NN'), ('dolor', 'NN'), ('sit', 'NN'), ('amet', 'NN'), (',', ','), ('consectetur', 'NN'), ('adipiscing', 'VBG'), ('elit', 'NN'), ('.', '.'), ('Fusce', 'NNP'), ('commodo', 'JJ'), ('mauris', 'NN'), ('id', 'NN'), ('justo', 'NN'), ('condimentum', 'NN'), ('dignissim', 'NN'), ('.', '.'), ('Nullam', 'NNP'), ('placerat', 'VBZ'), ('semper', 'JJR'), ('dapibus', 'NN'), ('.', '.'), ('Pellentesque', 'NNP'), ('ac', 'JJ'), ('risus', 'NN'), ('nulla', 'NN'), ('.', '.'), ('Phasellus', 'CC'), ('ut', 'JJ'), ('dapibus', 'NN'), ('nunc', 'NN'), (',', ','), ('id', 'JJ'), ('aliquam', 'NN'), ('dolor', 'NN'), ('.', '.')]

Stopwords

In [69]:

stop\_words **=** set(stopwords**.**words("english"))

print(stop\_words)

{'again', 'ours', 'there', 'a', 'any', 'at', 'wouldn', 'this', 'does', 'doing', 'by', "mightn't", "weren't", "that'll", 'who', 'where', 'been', 'mustn', 'that', 'then', 'haven', 'on', 'did', 'll', 't', 'below', "you're", 'being', 'shan', 'we', 'against', 'don', 'it', 'they', "haven't", 'hasn', 'while', 'himself', 'won', 'than', 'more', 'are', 'or', 'your', 'couldn', 'into', "needn't", "shouldn't", 'you', 'not', 'up', 'm', "should've", 'no', 'hers', 'some', 'once', 'if', "mustn't", 'here', 'both', 're', 'shouldn', 'needn', 'didn', 'through', 'between', 'i', 'very', "hasn't", 'was', 'after', 'he', 'just', 'off', 'so', "it's", 'when', 'her', 'hadn', "don't", 'myself', 'each', 'down', 's', 'were', 'ourselves', 'can', 'will', 'its', 'am', 'because', 'all', 'isn', 'but', "you've", 'and', 'before', 'my', 'wasn', 'weren', 'above', 'is', 'him', 'aren', 'now', 'yourself', "didn't", 'has', 'mightn', 'do', 'theirs', 'too', 'which', 'she', 'yours', 'in', 've', 'doesn', "wasn't", 'how', "aren't", 'd', 'itself', 'ma', 'ain', 'y', "won't", 'having', 'as', 'those', "wouldn't", 'to', 'for', "isn't", 'his', 'our', 'same', 'them', 'should', 'these', 'own', 'under', "you'd", 'an', 'with', 'few', 'only', 'had', 'such', 'most', 'be', 'o', 'during', 'out', 'further', "doesn't", 'other', 'why', 'themselves', 'yourselves', 'over', 'have', "she's", 'the', "couldn't", 'herself', 'what', "hadn't", 'about', 'their', 'of', 'nor', 'me', 'whom', "you'll", 'until', "shan't", 'from'}

In [70]:

to\_clean **=** word\_tokenize(text)

In [71]:

no\_stopwords\_text **=** []

**for** token **in** to\_clean:

**if**(token **not** **in** stop\_words):

no\_stopwords\_text**.**append(token)

print(no\_stopwords\_text)

['Lorem', 'ipsum', 'dolor', 'sit', 'amet', ',', 'consectetur', 'adipiscing', 'elit', '.', 'Fusce', 'commodo', 'mauris', 'id', 'justo', 'condimentum', 'dignissim', '.', 'Nullam', 'placerat', 'semper', 'dapibus', '.', 'Pellentesque', 'ac', 'risus', 'nulla', '.', 'Phasellus', 'ut', 'dapibus', 'nunc', ',', 'id', 'aliquam', 'dolor', '.']

Stemming

In [72]:

stemmer **=** PorterStemmer()

In [73]:

stemmed\_words **=** []

**for** token **in** no\_stopwords\_text:

stemmed\_word **=** stemmer**.**stem(token)

stemmed\_words**.**append(stemmed\_word)

In [74]:

print(stemmed\_words)

['lorem', 'ipsum', 'dolor', 'sit', 'amet', ',', 'consectetur', 'adipisc', 'elit', '.', 'fusc', 'commodo', 'mauri', 'id', 'justo', 'condimentum', 'dignissim', '.', 'nullam', 'placerat', 'semper', 'dapibu', '.', 'pellentesqu', 'ac', 'risu', 'nulla', '.', 'phasellu', 'ut', 'dapibu', 'nunc', ',', 'id', 'aliquam', 'dolor', '.']

Lemmatization

In [75]:

lemmatizer **=** WordNetLemmatizer()

In [78]:

lemmatized\_words **=** []

**for** token **in** no\_stopwords\_text:

lemmatized **=** lemmatizer**.**lemmatize(token) *# Assuming you want to lemmatize verbs (you can change the 'pos' argument as needed)*

lemmatized\_words**.**append(lemmatized)

In [79]:

print(lemmatized\_words)

['Lorem', 'ipsum', 'dolor', 'sit', 'amet', ',', 'consectetur', 'adipiscing', 'elit', '.', 'Fusce', 'commodo', 'mauris', 'id', 'justo', 'condimentum', 'dignissim', '.', 'Nullam', 'placerat', 'semper', 'dapibus', '.', 'Pellentesque', 'ac', 'risus', 'nulla', '.', 'Phasellus', 'ut', 'dapibus', 'nunc', ',', 'id', 'aliquam', 'dolor', '.']

TF-IDF Vectorization

In [82]:

vectorizer **=** TfidfVectorizer()

In [88]:

corpus **=** [

"I love to eat pizza",

"Pizza is my favorite food",

"I enjoy eating pizza with friends",

"I like to have pizza for dinner",

"Pizza toppings include cheese, pepperoni, and mushrooms"

]

In [89]:

vectorizer **=** TfidfVectorizer()

In [91]:

tfidf\_matrix **=** vectorizer**.**fit\_transform(corpus)

feature\_names **=** vectorizer**.**get\_feature\_names\_out()

In [92]:

print(tfidf\_matrix**.**toarray())

print(feature\_names)

[[0. 0. 0. 0.58946308 0. 0.

0. 0. 0. 0. 0. 0.

0. 0. 0.58946308 0. 0. 0.

0.28088232 0.4755751 0. 0. ]

[0. 0. 0. 0. 0. 0.

0.48638585 0.48638585 0. 0. 0. 0.

0.48638585 0. 0. 0. 0.48638585 0.

0.23176546 0. 0. 0. ]

[0. 0. 0. 0. 0.48638585 0.48638585

0. 0. 0. 0.48638585 0. 0.

0. 0. 0. 0. 0. 0.

0.23176546 0. 0. 0.48638585]

[0. 0. 0.45277275 0. 0. 0.

0. 0. 0.45277275 0. 0.45277275 0.

0. 0.45277275 0. 0. 0. 0.

0.21574864 0.36529421 0. 0. ]

[0.40073619 0.40073619 0. 0. 0. 0.

0. 0. 0. 0. 0. 0.40073619

0. 0. 0. 0.40073619 0. 0.40073619

0.19095294 0. 0.40073619 0. ]]

['and' 'cheese' 'dinner' 'eat' 'eating' 'enjoy' 'favorite' 'food' 'for'

'friends' 'have' 'include' 'is' 'like' 'love' 'mushrooms' 'my'

'pepperoni' 'pizza' 'to' 'toppings' 'with']