```
import nltk
from nltk.tokenize import word_tokenize,sent_tokenize
block = "India is a land of vibrant cultures and ancient traditions,
known for its diversity in languages, festivals, and cuisines. From
the snow-capped peaks of the Himalayas in the north to the tropical
beaches of Kerala in the south, the country offers a wide range of
landscapes. India is also home to many historical landmarks, such as
the Taj Mahal, Qutub Minar, and the forts of Rajasthan, which reflect
its rich architectural heritage."
print("This is word wise tokenization-:",'\n',
nltk.word_tokenize(block), '\n')
print("-----------------------------------------------------------------
--------------'\n'")
print("This is sentence wise tokenization-:",'\n',
nltk.sent_tokenize(block))
```

```
This is word wise tokenization-:
 ['India', 'is', 'a', 'land', 'of', 'vibrant', 'cultures', 'and',
'ancient', 'traditions', ',', 'known', 'for', 'its', 'diversity',
'in', 'languages', ',', 'festivals', ',', 'and', 'cuisines', '.',
'From', 'the', 'snow-capped', 'peaks', 'of', 'the', 'Himalayas', 'in',
'the', 'north', 'to', 'the', 'tropical', 'beaches', 'of', 'Kerala',
'in', 'the', 'south', ',', 'the', 'country', 'offers', 'a', 'wide',
'range', 'of', 'landscapes', '.', 'India', 'is', 'also', 'home', 'to',
'many', 'historical', 'landmarks', ',', 'such', 'as', 'the', 'Taj',
'Mahal', ',', 'Qutub', 'Minar', ',', 'and', 'the', 'forts', 'of',
'Rajasthan', ',', 'which', 'reflect', 'its', 'rich', 'architectural',
'heritage', '.']

----------------------------------------------------------------------
-------'
'
This is sentence wise tokenization-:
 ['India is a land of vibrant cultures and ancient traditions, known
for its diversity in languages, festivals, and cuisines.', 'From the
snow-capped peaks of the Himalayas in the north to the tropical
beaches of Kerala in the south, the country offers a wide range of
landscapes.', 'India is also home to many historical landmarks, such
as the Taj Mahal, Qutub Minar, and the forts of Rajasthan, which
reflect its rich architectural heritage.']
```

```
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = stopwords.words('english')
print(stop_words)
token = nltk.word_tokenize(block)
cleaned_token = []
for word in token:
    if word not in stop_words:
        cleaned_token.append(word)
```

```
print("This is the unclean version-:",'\n', token, '\n')
print("------------------------------------------------------------
-------'\n'")
print("This is the cleaned version-:",'\n', cleaned_token)
```

```
['a', 'about', 'above', 'after', 'again', 'against', 'ain', 'all',
'am', 'an', 'and', 'any', 'are', 'aren', "aren't", 'as', 'at', 'be',
'because', 'been', 'before', 'being', 'below', 'between', 'both',
'but', 'by', 'can', 'couldn', "couldn't", 'd', 'did', 'didn',
"didn't", 'do', 'does', 'doesn', "doesn't", 'doing', 'don', "don't",
'down', 'during', 'each', 'few', 'for', 'from', 'further', 'had',
'hadn', "hadn't", 'has', 'hasn', "hasn't", 'have', 'haven', "haven't",
'having', 'he', "he'd", "he'll", 'her', 'here', 'hers', 'herself',
"he's", 'him', 'himself', 'his', 'how', 'i', "i'd", 'if', "i'll",
"i'm", 'in', 'into', 'is', 'isn', "isn't", 'it', "it'd", "it'll",
"it's", 'its', 'itself', "i've", 'just', 'll', 'm', 'ma', 'me',
'mightn', "mightn't", 'more', 'most', 'mustn', "mustn't", 'my',
'myself', 'needn', "needn't", 'no', 'nor', 'not', 'now', 'o', 'of',
'off', 'on', 'once', 'only', 'or', 'other', 'our', 'ours',
'ourselves', 'out', 'over', 'own', 're', 's', 'same', 'shan',
"shan't", 'she', "she'd", "she'll", "she's", 'should', 'shouldn',
"shouldn't", "should've", 'so', 'some', 'such', 't', 'than', 'that',
"that'll", 'the', 'their', 'theirs', 'them', 'themselves', 'then',
'there', 'these', 'they', "they'd", "they'll", "they're", "they've",
'this', 'those', 'through', 'to', 'too', 'under', 'until', 'up', 've',
'very', 'was', 'wasn', "wasn't", 'we', "we'd", "we'll", "we're",
'were', 'weren', "weren't", "we've", 'what', 'when', 'where', 'which',
'while', 'who', 'whom', 'why', 'will', 'with', 'won', "won't",
'wouldn', "wouldn't", 'y', 'you', "you'd", "you'll", 'your', "you're",
'yours', 'yourself', 'yourselves', "you've"]
This is the unclean version-:
 ['India', 'is', 'a', 'land', 'of', 'vibrant', 'cultures', 'and',
'ancient', 'traditions', ',', 'known', 'for', 'its', 'diversity',
'in', 'languages', ',', 'festivals', ',', 'and', 'cuisines', '.',
'From', 'the', 'snow-capped', 'peaks', 'of', 'the', 'Himalayas', 'in',
'the', 'north', 'to', 'the', 'tropical', 'beaches', 'of', 'Kerala',
'in', 'the', 'south', ',', 'the', 'country', 'offers', 'a', 'wide',
'range', 'of', 'landscapes', '.', 'India', 'is', 'also', 'home', 'to',
'many', 'historical', 'landmarks', ',', 'such', 'as', 'the', 'Taj',
'Mahal', ',', 'Qutub', 'Minar', ',', 'and', 'the', 'forts', 'of',
'Rajasthan', ',', 'which', 'reflect', 'its', 'rich', 'architectural',
'heritage', '.']

----------------------------------------------------------------------
'
'
This is the cleaned version-:
 ['India', 'land', 'vibrant', 'cultures', 'ancient', 'traditions',
',', 'known', 'diversity', 'languages', ',', 'festivals', ',',
'cuisines', '.', 'From', 'snow-capped', 'peaks', 'Himalayas', 'north',
```

```
'tropical', 'beaches', 'Kerala', 'south', ',', 'country', 'offers',
'wide', 'range', 'landscapes', '.', 'India', 'also', 'home', 'many',
'historical', 'landmarks', ',', 'Taj', 'Mahal', ',', 'Qutub', 'Minar',
',', 'forts', 'Rajasthan', ',', 'reflect', 'rich', 'architectural',
'heritage', '.']
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Jayditya\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
from nltk.stem import PorterStemmer
stemmer = nltk.PorterStemmer()
words = ['rain', 'rained', 'raining', 'rains']
stemmed = [stemmer.stem(word) for word in words]
print(stemmed)
```

```
['rain', 'rain', 'rain', 'rain']
```

```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')#data dependencies
nltk.download('omw-1.4')
lemmatizer = nltk.WordNetLemmatizer()
lemmatized = [lemmatizer.lemmatize(word) for word in cleaned_token]
print(lemmatized)
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\Jayditya\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     C:\Users\Jayditya\AppData\Roaming\nltk_data...
```

```
['India', 'land', 'vibrant', 'culture', 'ancient', 'tradition', ',',
'known', 'diversity', 'language', ',', 'festival', ',', 'cuisine',
'.', 'From', 'snow-capped', 'peak', 'Himalayas', 'north', 'tropical',
'beach', 'Kerala', 'south', ',', 'country', 'offer', 'wide', 'range',
'landscape', '.', 'India', 'also', 'home', 'many', 'historical',
'landmark', ',', 'Taj', 'Mahal', ',', 'Qutub', 'Minar', ',', 'fort',
'Rajasthan', ',', 'reflect', 'rich', 'architectural', 'heritage', '.']
```

```
import nltk
from nltk import pos_tag


nltk.download('averaged_perceptron_tagger_eng')  # NEW line to fix
your error

tagged = pos_tag(cleaned_token)
print("POS Tagged Tokens:", tagged)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     C:\Users\Jayditya\AppData\Roaming\nltk_data...
```

```
POS Tagged Tokens: [('India', 'NNP'), ('land', 'NN'), ('vibrant',
'NN'), ('cultures', 'VBZ'), ('ancient', 'JJ'), ('traditions', 'NNS'),
(',', ','), ('known', 'VBN'), ('diversity', 'NN'), ('languages',
'NNS'), (',', ','), ('festivals', 'NNS'), (',', ','), ('cuisines',
'NNS'), ('.', '.'), ('From', 'IN'), ('snow-capped', 'JJ'), ('peaks',
'NNS'), ('Himalayas', 'NNP'), ('north', 'RB'), ('tropical', 'JJ'),
('beaches', 'NNS'), ('Kerala', 'NNP'), ('south', 'RB'), (',', ','),
('country', 'NN'), ('offers', 'NNS'), ('wide', 'JJ'), ('range', 'NN'),
('landscapes', 'NNS'), ('.', '.'), ('India', 'NNP'), ('also', 'RB'),
('home', 'VBD'), ('many', 'JJ'), ('historical', 'JJ'), ('landmarks',
'NNS'), (',', ','), ('Taj', 'NNP'), ('Mahal', 'NNP'), (',', ','),
('Qutub', 'NNP'), ('Minar', 'NNP'), (',', ','), ('forts', 'VBZ'),
('Rajasthan', 'NNP'), (',', ','), ('reflect', 'VBP'), ('rich', 'JJ'),
('architectural', 'JJ'), ('heritage', 'NN'), ('.', '.')]

[nltk_data]   Unzipping taggers\averaged_perceptron_tagger_eng.zip.
```

```python
import pandas as pd
import sklearn as sk
import math

block_1 = "India is known for its rich cultural heritage, diverse
traditions, and unity in diversity. People from various regions,
religions, and languages come together to celebrate festivals, support
each other, and build a strong nation."
block_2 = "India has made remarkable progress in fields like science,
technology, space research, and education. The government and citizens
work together to uplift society by promoting innovation, digital
growth, and inclusive development."

# Split so each word has its own string
first_block = block_1.split(" ")
second_block = block_2.split(" ")

# Join them to remove common duplicate words
total = set(first_block).union(set(second_block))
print(total)
```

```
{'work', 'research,', 'each', 'technology,', 'promoting', 'unity',
'development.', 'rich', 'fields', 'cultural', 'various', 'like', 'to',
'religions,', 'society', 'other,', 'People', 'nation.', 'science,',
'innovation,', 'India', 'diverse', 'languages', 'diversity.',
'progress', 'growth,', 'its', 'uplift', 'come', 'from', 'education.',
'for', 'celebrate', 'strong', 'build', 'known', 'made', 'festivals,',
'a', 'The', 'inclusive', 'and', 'government', 'is', 'space',
'digital', 'traditions,', 'together', 'support', 'by', 'has', 'in',
'heritage,', 'regions,', 'remarkable', 'citizens'}
```

```python
wordDictA = dict.fromkeys(total, 0)
wordDictB = dict.fromkeys(total, 0)
for word in first_block:
```

```
    wordDictA[word]+=1
for word in second_block:
    wordDictB[word]+=1
pd.DataFrame([wordDictA, wordDictB])
```

```
   work  research,  each  technology,  promoting  unity  development.
rich  \
0     0          0     1            0          0      1              0
1
1     1          1     0            1          1      0              1
0

   fields  cultural  ...  traditions,  together  support  by  has  in
\
0       0         1  ...            1         1        1   0    0   1

1       1         0  ...            0         1        0   1    1   1


   heritage,  regions,  remarkable  citizens
0          1         1           0         0
1          0         0           1         1

[2 rows x 56 columns]
```

```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
filtered_sentence = [w for w in wordDictA if not w in stop_words]
print(filtered_sentence)
```

```
['work', 'research,', 'technology,', 'promoting', 'unity',
'development.', 'rich', 'fields', 'cultural', 'various', 'like',
'religions,', 'society', 'other,', 'People', 'nation.', 'science,',
'innovation,', 'India', 'diverse', 'languages', 'diversity.',
'progress', 'growth,', 'uplift', 'come', 'education.', 'celebrate',
'strong', 'build', 'known', 'made', 'festivals,', 'The', 'inclusive',
'government', 'space', 'digital', 'traditions,', 'together',
'support', 'heritage,', 'regions,', 'remarkable', 'citizens']

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Jayditya\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
def computeTF(wordDict, doc):
    tfDict = {}
    corpusCount = len(doc)
    for word, count in wordDict.items():tfDict[word] =
count/float(corpusCount)
    return(tfDict)
```

```
#running our sentences through the tf function:
tfFirst = computeTF(wordDictA, first_block)
tfSecond = computeTF(wordDictB, second_block)
tf = pd.DataFrame([tfFirst, tfSecond])
print(tf)
```

```
       work  research,      each  technology,  promoting     unity  \
0  0.000000   0.000000  0.029412     0.000000   0.000000  0.029412
1  0.032258   0.032258  0.000000     0.032258   0.032258  0.000000

   development.      rich    fields  cultural  ...  traditions,
together  \
0      0.000000  0.029412  0.000000  0.029412  ...     0.029412
0.029412
1      0.032258  0.000000  0.032258  0.000000  ...     0.000000
0.032258

    support        by       has        in  heritage,  regions,
remarkable  \
0  0.029412  0.000000  0.000000  0.029412   0.029412  0.029412
0.000000
1  0.000000  0.032258  0.032258  0.032258   0.000000  0.000000
0.032258

   citizens
0  0.000000
1  0.032258

[2 rows x 56 columns]
```

```
def computeIDF(docList):
    idfDict = {}
    N = len(docList)
    idfDict = dict.fromkeys(docList[0].keys(), 0)
    for word, val in idfDict.items(): idfDict[word] = math.log10(N
/(float(val) + 1))
    return(idfDict)

idfs = computeIDF([wordDictA, wordDictB])
idfs1 = pd.DataFrame([wordDictA, wordDictB])
print(idfs1)
```

```
   work  research,  each  technology,  promoting  unity  development.
rich  \
0     0          0     1            0          0      1              0
1
1     1          1     0            1          1      0              1
0

   fields  cultural  ...  traditions,  together  support  by  has  in
\
```

```
0       0         1  ...             1         1        1   0    0   1

1       1         0  ...             0         1        0   1    1   1


   heritage,  regions,  remarkable  citizens
0          1         1           0         0
1          0         0           1         1

[2 rows x 56 columns]
```

```
def computeTFIDF(tfBow, idfs):
    tfidf = {}
    for word, val in tfBow.items(): tfidf[word] = val*idfs[word]
    return(tfidf)
#running our two sentences through the IDF:
idfFirst = computeTFIDF(tfFirst, idfs)
idfSecond = computeTFIDF(tfSecond, idfs)
#putting it in a dataframe
idf= pd.DataFrame([idfFirst, idfSecond])
print(idf)
```

```
       work  research,      each  technology,  promoting     unity  \
0  0.000000   0.000000  0.008854     0.000000   0.000000  0.008854
1  0.009711   0.009711  0.000000     0.009711   0.009711  0.000000

   development.      rich    fields  cultural  ...  traditions,
together  \
0      0.000000  0.008854  0.000000  0.008854  ...     0.008854
0.008854
1      0.009711  0.000000  0.009711  0.000000  ...     0.000000
0.009711

    support        by       has        in  heritage,  regions,
remarkable  \
0  0.008854  0.000000  0.000000  0.008854   0.008854  0.008854
0.000000
1  0.000000  0.009711  0.009711  0.009711   0.000000  0.000000
0.009711

   citizens
0  0.000000
1  0.009711

[2 rows x 56 columns]
```