```
import nltk
import re
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,WordNetLemmatizer
```

```
text = "I am a student.hello!! there is a session going onn."
```

```
def preprocess_text(text):
    if text:
        text = text.lower()
        text = re.sub(r'[^\w\s]', " ", text)
    return text
```

```
preprocessed_text = preprocess_text(text)
preprocessed_text
```

```
'i am a student hello   there is a session going onn '
```

```
def tokenize(text):
    tokens = word_tokenize(text)
    return tokens
```

```
import nltk
import re
from nltk.tokenize import word_tokenize
from nltk import pos_tag
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,WordNetLemmatizer
# Download the 'punkt_tab' resource
nltk.download('punkt_tab')
# Download the 'punkt' resource (already downloaded in the user's code, but including it for completeness)
nltk.download('punkt')

text = "I am a student.hello!! there is a session going onn."

def preprocess_text(text):
    if text:
        text = text.lower()
        text = re.sub(r'[^\w\s]', " ", text)
    return text

preprocessed_text = preprocess_text(text)
preprocessed_text

def tokenize(text):
    tokens = word_tokenize(text)
    return tokens

tokens = tokenize(preprocessed_text)
tokens
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
['i',
 'am',
 'a',
 'student',
 'hello',
 'there',
 'is',
 'a',
 'session',
 'going',
 'onn']
```

```
def pos_tagging(tokens):
    pos_tags = pos_tag(tokens)
    return pos_tags
```

```
import nltk
import re
from nltk.tokenize import word_tokenize
from nltk import pos_tag
```

```
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer,WordNetLemmatizer

# Download the 'punkt' resource (already downloaded in the user's code, but including it for completeness)
nltk.download('punkt')
# Download the resource needed for pos_tag in English
nltk.download('averaged_perceptron_tagger_eng')


text = "I am a student.hello!! there is a session going onn."

def preprocess_text(text):
    if text:
        text = text.lower()
        text = re.sub(r'[^\w\s]', " ", text)
    return text

preprocessed_text = preprocess_text(text)

def tokenize(text):
    tokens = word_tokenize(text)
    return tokens

tokens = tokenize(preprocessed_text)

def pos_tagging(tokens):
    pos_tags = pos_tag(tokens) # This line was causing the error
    return pos_tags

pos_tags = pos_tagging(tokens)
pos_tags
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
[('i', 'NN'),
 ('am', 'VBP'),
 ('a', 'DT'),
 ('student', 'NN'),
 ('hello', 'NN'),
 ('there', 'EX'),
 ('is', 'VBZ'),
 ('a', 'DT'),
 ('session', 'NN'),
 ('going', 'VBG'),
 ('onn', 'NN')]
```

```
nltk.download('stopwords')
def remove_stop_words(tokens):
    stop_words = stopwords.words('english')
    filtered_tokens = [words for words in tokens  if words not in stop_words]
    return filtered_tokens
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
filtered_tokens = remove_stop_words(tokens)
filtered_tokens
```

```
['student', 'hello', 'session', 'going', 'onn']
```

```
def stem_tokens(tokens):
    stemming = PorterStemmer()
    stemmed_tokens = [stemming.stem(word) for word in tokens]
    return stemmed_tokens
```

```
stemmed_tokens = stem_tokens(filtered_tokens)
stemmed_tokens
```

```
['student', 'hello', 'session', 'go', 'onn']
```

```
def lemmatization(tokens):
    lemma = WordNetLemmatizer()
    lemmatized_tokens = [lemma.lemmatize(word) for word in tokens]
    return lemmatized_tokens
```

```
nltk.download('wordnet')
lemmatized_tokens = lemmatization(filtered_tokens)
```

```
lemmatized_tokens
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
['student', 'hello', 'session', 'going', 'onn']
```

```
def calculate_term_frequency(tokens):
    word_counts = {}
    for word in tokens:
        word_counts[word] = word_counts.get(word, 0) + 1
    total_words = sum(word_counts.values())
    term_frequencies = {word: count / total_words for word, count in word_counts.items()}
    return term_frequencies
```

```
calculate_term_frequency(filtered_tokens)
```

```
{'student': 0.2, 'hello': 0.2, 'session': 0.2, 'going': 0.2, 'onn': 0.2}
```

```
import math
def calculate_document_frequency(tokens):
    unique_words = set(tokens)
    document_frequencies = {word: 1 for word in unique_words}
    return document_frequencies,unique_words

def calculate_inverse_document_frequency(tokens):
    document_frequencies,unique_words = calculate_document_frequency(tokens)
    N = 1  # Assuming we have only one document
    inverse_document_frequencies = {word: math.log(N / document_freq) for word, document_freq in document_frequencies.items()}
    return inverse_document_frequencies,unique_words
```

```
calculate_inverse_document_frequency(filtered_tokens)
```

```
({'onn': 0.0, 'student': 0.0, 'hello': 0.0, 'going': 0.0, 'session': 0.0},
 {'going', 'hello', 'onn', 'session', 'student'})
```