1. Tokenization

Tokenization is the process of breaking text into individual words or sentences.

Explanation:

Downloading 'punkt': nltk.download('punkt') downloads the Punkt tokenizer models, which are pre-trained models for tokenizing text.

Tokenizing Text: word_tokenize(text) splits the text into individual words and punctuation marks.

```
import nltk
nltk.download("punkt")
from nltk.tokenize import word_tokenize

text="Hello I am there Where are you"
tokens=nltk.word_tokenize(text)
tokens
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
['Hello', 'I', 'am', 'there', 'Where', 'are', 'you']
```

2. Removing Stop Words

Stop words are common words that are often removed from text data.

```
import nltk
from nltk.corpus import stopwords

# Ensure the necessary NLTK data files are downloaded
nltk.download('punkt')
nltk.download('stopwords')

# Load English stopwords
stop_words = set(stopwords.words("english"))

# Example text
text = "Where are you I am here doing natural language processing tasks"

# Tokenize the text
tokens = nltk.word_tokenize(text)

# Filter out stopwords
filtered_text = [word for word in tokens if word.lower() not in stop_words]

print(filtered_text)
```

```
['natural', 'language', 'processing', 'tasks']
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

3. Stemming

Stemming is the process of reducing words to their base or root form.

```
from nltk.stem import PorterStemmer
stemmer=PorterStemmer()
words=["jumping","running","runs","played"]
stemmed_words=[stemmer.stem(word) for word in words]
print(stemmed_words)
```

```
['jump', 'run', 'run', 'play']
```

4. Lemmatization

Lemmatization is the process of reducing words to their base or dictionary form.

```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()
words = ["running", "jumps", "easily", "fairly"]
lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
print(lemmatized_words)
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
['running', 'jump', 'easily', 'fairly']
```

5. Part-of-Speech Tagging

POS tagging assigns parts of speech to each word in a sentence.

DT: Determiner (e.g., "the", "a") JJ: Adjective (e.g., "quick", "lazy") NN: Noun, singular or mass (e.g., "fox", "dog") VBZ: Verb, 3rd person singular present (e.g., "jumps") IN: Preposition or subordinating conjunction (e.g., "over") .: Punctuation mark (e.g., ".")

```
nltk.download('averaged_perceptron_tagger')

sentence = "The quick brown fox jumps over the lazy dog."
pos_tags = nltk.pos_tag(word_tokenize(sentence))
print(pos_tags)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('lazy
```

6. Named Entity Recognition

NER identifies named entities in text.

```
#This imports the spaCy library and loads the English language model ("en_core_web_sm").

import spacy
nlp = spacy.load("en_core_web_sm")

text = "Apple is looking at buying U.K. startup for $1 billion"
doc = nlp(text)
#Iterates over the entities recognized in the processed document (doc) and prints each entity's text (ent.text) along with it
for ent in doc.ents:
    print(ent.text, ent.label_)
```

```
Apple ORG
U.K. GPE
$1 billion MONEY
```

Output Explanation The output shows the recognized entities and their corresponding labels:

Apple: Recognized as an organization (ORG). U.K.: Recognized as a geopolitical entity (GPE). $1 billion: Recognized as a monetary value (MONEY).

```
# 7. Sentence Tokenization
# Sentence tokenization splits text into sentences.

from nltk.tokenize import sent_tokenize

text = "Hello world! How are you today? Welcome to NLP."
sentences = sent_tokenize(text)
print(sentences)
```

```
['Hello world!', 'How are you today?', 'Welcome to NLP.']
```

```
# 8. Text Normalization
# Text normalization converts text to a standard format.

import re

text = "This is an example text with punctuation, numbers 123 and UPPERCASE letters."
normalized_text = re.sub(r'\d+', '', text).lower()
print(normalized_text)
```

```
this is an example text with punctuation, numbers  and uppercase letters.
```

re.sub(r'\d+', '', text): Uses the re.sub() function to substitute (replace) all sequences of digits (\d+) in the text with an empty string '', effectively removing the digits. .lower(): Converts the resulting text to lowercase.

Text normalization is an important preprocessing step in natural language processing tasks. While this example focuses on removing digits and converting text to lowercase

```
#9. Spell Checking
#Spell checking corrects spelling errors in text.

from textblob import TextBlob

text = "I havv a spelking errror."
blob = TextBlob(text)
print(blob.correct())
```

```
I have a speaking error.
```

Start coding or generate with AI.