```
import nltk
import spacy
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import PorterStemmer

nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
medical_text = """
Patients diagnosed with diabetes mellitus require regular monitoring.
Insulin injections help control blood glucose levels.
Early detection of cardiovascular diseases reduces mortality.
Doctors recommend physical exercise and balanced nutrition.
"""
```

Sentence tokenization

```
import nltk
nltk.download('punkt_tab')
sentences = sent_tokenize(medical_text)

print("Sentence Tokens:\n")
for sentence in sentences:
    print(sentence)
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
Sentence Tokens:


Patients diagnosed with diabetes mellitus require regular monitoring.
Insulin injections help control blood glucose levels.
Early detection of cardiovascular diseases reduces mortality.
Doctors recommend physical exercise and balanced nutrition.
```

Word tokenization

```
words = word_tokenize(medical_text)

print("Word Tokens:\n")
for word in words:
    print(word)
```

```
Word Tokens:

Patients
diagnosed
with
diabetes
mellitus
require
regular
monitoring
.
Insulin
injections
help
control
blood
glucose
levels
.
Early
detection
of
cardiovascular
diseases
reduces
mortality
```

```
.
Doctors
recommend
physical
exercise
and
balanced
nutrition
.
```

Filtering stop words

```
# Filtering Stop Words using NLTK

import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

nltk.download('punkt')
nltk.download('stopwords')

medical_text = "Patients diagnosed with diabetes require regular monitoring"

tokens = word_tokenize(medical_text)

stop_words = set(stopwords.words('english'))

filtered_words = [word for word in tokens if word.lower() not in stop_words and word.isalpha()]

print("Filtered Words (NLTK):\n")
for word in filtered_words:
    print(word)
```

```
Filtered Words (NLTK):

Patients
diagnosed
diabetes
require
regular
monitoring
[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]   Unzipping corpora/stopwords.zip.
```

Stemming

```
stemmer = PorterStemmer()

print("Stemmed Output:\n")
for word in words:
    if word.isalpha():
        print(f"{word} → {stemmer.stem(word)}")
```

```
Stemmed Output:

Patients → patient
diagnosed → diagnos
with → with
diabetes → diabet
mellitus → mellitu
require → requir
regular → regular
monitoring → monitor
Insulin → insulin
injections → inject
help → help
control → control
blood → blood
glucose → glucos
levels → level
Early → earli
detection → detect
of → of
cardiovascular → cardiovascular
```

```
diseases → diseas
reduces → reduc
mortality → mortal
Doctors → doctor
recommend → recommend
physical → physic
exercise → exercis
and → and
balanced → balanc
nutrition → nutrit
```

Under stemming

```
# Demonstration of UNDERSTEMMING using NLTK (medical terms)

from nltk.stem import PorterStemmer

stemmer = PorterStemmer()

medical_terms = [
    "diagnose",
    "diagnosis",
    "diagnostic",
    "disease",
    "diseases"
]

print("Understemming Output:\n")

for term in medical_terms:
    print(f"{term} → {stemmer.stem(term)}")
```

```
Understemming Output:

diagnose → diagnos
diagnosis → diagnosi
diagnostic → diagnost
disease → diseas
diseases → diseas
```

Lemmatization

```
import spacy

nlp = spacy.load('en_core_web_sm')
doc = nlp(medical_text)

print("Lemmatized Output:\n")
for token in doc:
    if token.is_alpha:
        print(f"{token.text} → {token.lemma_}")
```

```
Lemmatized Output:

Patients → patient
diagnosed → diagnose
with → with
diabetes → diabete
mellitus → mellitus
require → require
regular → regular
monitoring → monitoring
Insulin → Insulin
injections → injection
help → help
control → control
blood → blood
glucose → glucose
levels → level
Early → early
detection → detection
of → of
cardiovascular → cardiovascular
diseases → disease
reduces → reduce
mortality → mortality
Doctors → doctor
recommend → recommend
```

```
physical → physical
exercise → exercise
and → and
balanced → balanced
nutrition → nutrition
```

Comparison

```
# Comparison of UNDERSTEMMING (Stemming) vs Lemmatization for medical terms

from nltk.stem import PorterStemmer
import spacy

# Initialize stemmer and spaCy model
stemmer = PorterStemmer()
nlp = spacy.load("en_core_web_sm")

medical_terms = [
    "diagnose",
    "diagnosis",
    "diagnostic",
    "disease",
    "diseases"
]

print("STEMMING (Understemming):\n")
for term in medical_terms:
    print(f"{term} → {stemmer.stem(term)}")

print("\nLEMMATIZATION:\n")
doc = nlp(" ".join(medical_terms))
for token in doc:
    print(f"{token.text} → {token.lemma_}")
```

```
STEMMING (Understemming):

diagnose → diagnos
diagnosis → diagnosi
diagnostic → diagnost
disease → diseas
diseases → diseas

LEMMATIZATION:

diagnose → diagnose
diagnosis → diagnosis
diagnostic → diagnostic
disease → disease
diseases → disease
```

Parts of speech

```
# Part-of-Speech (POS) Tagging using NLTK

import nltk
from nltk import word_tokenize, pos_tag

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger_eng') # Changed to download specific English tagger

medical_text = "Patients diagnosed with diabetes require regular monitoring"

# Tokenize
tokens = word_tokenize(medical_text)

# POS Tagging
pos_tags = pos_tag(tokens)

print("POS Tagging Output:\n")
for word, tag in pos_tags:
    print(f"{word} → {tag}")
```

```
[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.
POS Tagging Output:

Patients → NNS
diagnosed → VBN
with → IN
diabetes → NNS
require → VBP
regular → JJ
monitoring → NN
```

```
nltk.download('tagsets_json')
nltk.help.upenn_tagset()
```

```
$: dollar
    $ -$ --$ A$ C$ HK$ M$ NZ$ S$ U.S.$ US$
'': closing quotation mark
    ' ''
(: opening parenthesis
    ( [ {
): closing parenthesis
    ) ] }
,: comma
    ,
--: dash
    --
.: sentence terminator
    . ! ?
:: colon or ellipsis
    : ; ...
CC: conjunction, coordinating
    & 'n and both but either et for less minus neither nor or plus so
    therefore times v. versus vs. whether yet
CD: numeral, cardinal
    mid-1890 nine-thirty forty-two one-tenth ten million 0.5 one forty-
    seven 1987 twenty '79 zero two 78-degrees eighty-four IX '60s .025
    fifteen 271,124 dozen quintillion DM2,000 ...
DT: determiner
    all an another any both del each either every half la many much nary
    neither no some such that the them these this those
EX: existential there
    there
FW: foreign word
    gemeinschaft hund ich jeux habeas Haementeria Herr K'ang-si vous
    lutihaw alai je jour objets salutaris fille quibusdam pas trop Monte
    terram fiche oui corporis ...
IN: preposition or conjunction, subordinating
    astride among uppon whether out inside pro despite on by throughout
    below within for towards near behind atop around if like until below
    next into if beside ...
JJ: adjective or numeral, ordinal
    third ill-mannered pre-war regrettable oiled calamitous first separable
    ectoplasmic battery-powered participatory fourth still-to-be-named
    multilingual multi-disciplinary ...
JJR: adjective, comparative
    bleaker braver breezier briefer brighter brisker broader bumper busier
    calmer cheaper choosier cleaner clearer closer colder commoner costlier
    cozier creamier crunchier cuter ...
JJS: adjective, superlative
    calmest cheapest choicest classiest cleanest clearest closest commonest
    corniest costliest crassest creepiest crudest cutest darkest deadliest
    dearest deepest densest dinkiest ...
LS: list item marker
    A A. B B. C C. D E F First G H I J K One SP-44001 SP-44002 SP-44005
    SP-44007 Second Third Three Two * a b c d first five four one six three
    two
MD: modal auxiliary
    can cannot could couldn't dare may might must need ought shall should
    shouldn't will would
NN: noun, common, singular or mass
    common-carrier cabbage knuckle-duster Casino afghan shed thermostat
    investment slide humour falloff slick wind hyena override subhumanity
```