```
SRUniversity="""The SR University campus is located in Ananthasagar village of Hasanparthy Mandal in Warangal, Telangana, Ind
It is in 150 acres, with both separate hostel facilities for boys and girls.
There is a huge central library along with Indias largest Technology Business Incubator (TBI) in tier 2 cities."""
```

```
SRUniversity
```

```
'The SR University campus is located in Ananthasagar village of Hasanparthy Mandal in Warangal, Telangana, India. \nIt is in 1
50 acres, with both separate hostel facilities for boys and girls. \nThere is a huge central library along with Indias largest
Technology Business Incubator (TBI) in tier 2 cities.'
```

```
import nltk
nltk.download('punkt')
nltk.download('punkt_tab')
from nltk.tokenize import word_tokenize
word_tokenize(SRUniversity)
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
['The',
 'SR',
 'University',
 'campus',
 'is',
 'located',
 'in',
 'Ananthasagar',
 'village',
 'of',
 'Hasanparthy',
 'Mandal',
 'in',
 'Warangal',
 ',',
 'Telangana',
 ',',
 'India',



 'is',
 'in',
 '150',
 'acres',
 ',',
 'with',
 'both',
 'separate',
 'hostel',
 'facilities',
 'for',
 'boys',
 'and',
 'girls',
 '.',
 'There',
 'is',
 'a',
 'huge',
 'central',
 'library',
 'along',
 'with',
 'Indias',
 'largest',
 'Technology',
 'Business',
 'Incubator',
 '(',
 'TBI',
 ')',
 'in',
 'tier',
 '2',
 'cities',
 '.']
```

```
from nltk.tokenize import sent_tokenize
sent_tokenize(SRUniversity)
```

```
['The SR University campus is located in Ananthasagar village of Hasanparthy Mandal in Warangal, Telangana, India.',
 'It is in 150 acres, with both separate hostel facilities for boys and girls.',
 'There is a huge central library along with Indias largest Technology Business Incubator (TBI) in tier 2 cities.']
```

```
nltk.download("stopwords")
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

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

```
words_in_quote = word_tokenize(SRUniversity)
words_in_quote
```

```
['The',
 'SR',
 'University',
 'campus',
 'is',
 'located',
 'in',
 'Ananthasagar',
 'village',
 'of',
 'Hasanparthy',
 'Mandal',
 'in',
 'Warangal',
 ',',
 'Telangana',
 ',',
 'India',
 '.',
 'It',
 'is',
 'in',
 '150',
 'acres',
 ',',
 'with',
 'both',
 'separate',
 'hostel',
 'facilities',
 'for',
 'boys',
 'and',
 'girls',
 '.',
 'There',
 'is',
 'a',
 'huge',
 'central',
 'library',
 'along',
 'with',
 'Indias',
 'largest',
 'Technology',
 'Business',
 'Incubator',
 '(',
 'TBI',
 ')',
 'in',
 'tier',
 '2',
 'cities',
 '.']
```

```
stop_words = set(stopwords.words("english"))
filtered_list = []
for word in words_in_quote:
  if word.casefold() not in stop_words:
    filtered_list.append(word)
filtered_list
```

```
['SR',
 'University',
 'campus',
 'located',
 'Ananthasagar',
 'village',
 'Hasanparthy',
 'Mandal',
 'Warangal',
 ',',
 'Telangana',
 ',',
 'India',
 '.',
 '150',
 'acres',
 ',',
 'separate',
 'hostel',
 'facilities',
 'boys',
 'girls',
 '.',
 'huge',
 'central',
 'library',
 'along',
 'Indias',
 'largest',
 'Technology',
 'Business',
 'Incubator',
 '(',
 'TBI',
 ')',
 'tier',
 '2',
 'cities',
 '.']
```

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer = PorterStemmer()
words = word_tokenize(SRUniversity)
stemmed_words = [stemmer.stem(word) for word in words]
stemmed_words
```

```
['the',
 'sr',
 'univers',
 'campu',
 'is',
 'locat',
 'in',
 'ananthasagar',
 'villag',
 'of',
 'hasanparthi',
 'mandal',
 'in',
 'warang',
 ',',
 'telangana',
 ',',
 'india',
 '.',
 'it',
 'is',
 'in',
 '150',
 'acr',
 ',',
 'with',
 'both',
 'separ',
 'hostel',
 'facil',
 'for',
 'boy',
 'and',
 'girl',
 '.',
 'there',
 'is',
 'a',
 'huge',
 'central',
 'librari',
 'along',
 'with',
 'india',
 'largest',
 'technolog',
 'busi',
 'incub',
 '(',
 'tbi',
 ')',
 'in',
 'tier',
 '2',
 'citi',
 '.']
```

```
from nltk.stem import SnowballStemmer
snowball = SnowballStemmer(language='english')
words = word_tokenize(SRUniversity)
for word in words:
    print(word,"--->",snowball.stem(word))
```

```
The ---> the
SR ---> sr
University ---> univers
campus ---> campus
is ---> is
located ---> locat
in ---> in
Ananthasagar ---> ananthasagar
village ---> villag
of ---> of
Hasanparthy ---> hasanparthi
Mandal ---> mandal
in ---> in
Warangal ---> warang
, ---> ,
Telangana ---> telangana
, ---> ,
India ---> india
. ---> .
It ---> it
is ---> is
in ---> in
150 ---> 150
acres ---> acr
, ---> ,
with ---> with
both ---> both
separate ---> separ
hostel ---> hostel
facilities ---> facil
for ---> for
boys ---> boy
and ---> and
girls ---> girl
. ---> .
There ---> there
is ---> is
a ---> a
huge ---> huge
central ---> central
library ---> librari
along ---> along
with ---> with
Indias ---> india
largest ---> largest
Technology ---> technolog
Business ---> busi
Incubator ---> incub
( ---> (
TBI ---> tbi
) ---> )
in ---> in
tier ---> tier
2 ---> 2
cities ---> citi
. ---> .
```

```
from nltk import LancasterStemmer
Lanc = LancasterStemmer()
words = word_tokenize(SRUniversity)
for word in words:
    print(word,"--->",Lanc.stem(word))
```

```
The ---> the
SR ---> sr
University ---> univers
campus ---> camp
```

```
is ---> is
located ---> loc
in ---> in
Ananthasagar ---> ananthasag
village ---> vil
of ---> of
Hasanparthy ---> hasanparthy
Mandal ---> mand
in ---> in
Warangal ---> warang
, ---> ,
Telangana ---> telangan
, ---> ,
India ---> ind
. ---> .
It ---> it
is ---> is
in ---> in
150 ---> 150
acres ---> acr
, ---> ,
with ---> with
both ---> both
separate ---> sep
hostel ---> hostel
facilities ---> facil
for ---> for
boys ---> boy
and ---> and
girls ---> girl
. ---> .
There ---> ther
is ---> is
a ---> a
huge ---> hug
central ---> cent
library ---> libr
along ---> along
with ---> with
Indias ---> india
largest ---> largest
Technology ---> technolog
Business ---> busy
```

```
Incubator ---> incub
( ---> (
TBI ---> tbi
) ---> )
in ---> in
tier ---> tier
2 ---> 2
cities ---> city
. ---> .
```

```python
from nltk.stem import RegexpStemmer
regexp = RegexpStemmer('ing|e', min=4)
words = word_tokenize(SRUniversity)
for word in words:
    print(word,"--->",regexp.stem(word))
```

```
The ---> The
SR ---> SR
University ---> Univrsity
campus ---> campus
is ---> is
located ---> locatd
in ---> in
Ananthasagar ---> Ananthasagar
village ---> villag
of ---> of
Hasanparthy ---> Hasanparthy
Mandal ---> Mandal
in ---> in
Warangal ---> Warangal
, ---> ,
Telangana ---> Tlangana
, ---> ,
India ---> India
. ---> .
It ---> It
is ---> is
in ---> in
150 ---> 150
acres ---> acrs
, ---> ,
with ---> with
```

```
both ---> both
separate ---> sparat
hostel ---> hostl
facilities ---> facilitis
for ---> for
boys ---> boys
and ---> and
girls ---> girls
. ---> .
There ---> Thr
is ---> is
a ---> a
huge ---> hug
central ---> cntral
library ---> library
along ---> along
with ---> with
Indias ---> Indias
largest ---> largst
Technology ---> Tchnology
Business ---> Businss
Incubator ---> Incubator
( ---> (
TBI ---> TBI
) ---> )
in ---> in
tier ---> tir
2 ---> 2
cities ---> citis
. ---> .
```

```
nltk.download('omw-1.4')
nltk.download('wordnet') # Added to download the missing wordnet corpus
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
words = word_tokenize(SRUniversity)
for word in words:
    print(word,"--->",lemmatizer.lemmatize(word))
```

```
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
```

```
SR ---> SR
University ---> University
campus ---> campus
is ---> is
located ---> located
in ---> in
Ananthasagar ---> Ananthasagar
village ---> village
of ---> of
Hasanparthy ---> Hasanparthy
Mandal ---> Mandal
in ---> in
Warangal ---> Warangal
, ---> ,
Telangana ---> Telangana
, ---> ,
India ---> India
. ---> .
It ---> It
is ---> is
in ---> in
150 ---> 150
acres ---> acre
, ---> ,
with ---> with
both ---> both
separate ---> separate
hostel ---> hostel
facilities ---> facility
for ---> for
boys ---> boy
and ---> and
girls ---> girl
. ---> .
There ---> There
is ---> is
a ---> a
huge ---> huge
central ---> central
library ---> library
along ---> along
with ---> with

largest ---> largest
largest ---> largest
Technology ---> Technology
Business ---> Business
Incubator ---> Incubator
( ---> (
TBI ---> TBI
) ---> )
in ---> in
tier ---> tier
2 ---> 2
cities ---> city
```

```
lemmatizer.lemmatize("worst")
```

```
'worst'
```

```
lemmatizer.lemmatize("worst", pos="a")
```

```
'bad'
```

```
from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer, RegexpStemmer, WordNetLemmatizer
porter = PorterStemmer()
lancaster = LancasterStemmer()
snowball = SnowballStemmer(language='english')
regexp = RegexpStemmer('ing|e', min=4)
lemmatizer = WordNetLemmatizer()

word_list = ["friend", "friendship", "friends", "friendships"]
print("{0:20}{1:20}{2:20}{3:30}{4:40}{5:50}".format("Word","Porter Stemmer","Snowball Stemmer","Lancaster Stemmer",'Regexp St
for word in word_list:
    print("{0:20}{1:20}{2:20}{3:30}{4:40}{5:50}".format(word,porter.stem(word),snowball.stem(word),lancaster.stem(word),regex
```

```
Word                Porter Stemmer      Snowball Stemmer    Lancaster Stemmer             Regexp Stemmer
friend              friend              friend              friend                        frind
friendship          friendship          friendship          friend                        frindship
friends             friend              friend              friend                        frinds
friendships         friendship          friendship          friend                        frindships
```

```
nltk.download('averaged_perceptron_tagger_eng')
```

```
from nltk.tokenize import word_tokenize
words = word_tokenize(SRUniversity)
nltk.pos_tag(words)
```

```
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
[('The', 'DT'),
 ('SR', 'NNP'),
 ('University', 'NNP'),
 ('campus', 'NN'),
 ('is', 'VBZ'),
 ('located', 'VBN'),
 ('in', 'IN'),
 ('Ananthasagar', 'NNP'),
 ('village', 'NN'),
 ('of', 'IN'),
 ('Hasanparthy', 'NNP'),
 ('Mandal', 'NNP'),
 ('in', 'IN'),
 ('Warangal', 'NNP'),
 (',', ','),
 ('Telangana', 'NNP'),
 (',', ','),
 ('India', 'NNP'),
 ('.', '.'),
 ('It', 'PRP'),
 ('is', 'VBZ'),
 ('in', 'IN'),
 ('150', 'CD'),
 ('acres', 'NNS'),
 (',', ','),
 ('with', 'IN'),
 ('both', 'DT'),
 ('separate', 'JJ'),
 ('hostel', 'NN'),
 ('facilities', 'NNS'),
 ('for', 'IN'),
 ('boys', 'NNS'),
 ('and', 'CC'),
 ('girls', 'NNS'),
 ('.', '.'),
 ('is', 'VBZ'),
 ('a', 'DT'),
 ('huge', 'JJ'),
 ('central', 'JJ'),
 ('library', 'NN'),
 ('along', 'IN'),
 ('with', 'IN'),
 ('Indias', 'NNP'),
 ('largest', 'JJS'),
 ('Technology', 'NN'),
 ('Business', 'NNP'),
 ('Incubator', 'NNP'),
 ('(', '('),
 ('TBI', 'NNP'),
 (')', ')'),
 ('in', 'IN'),
 ('tier', '$'),
 ('2', 'CD'),
 ('cities', 'NNS'),
 ('.', '.')]
```

```
nltk.download('tagsets_json') # Corrected to download tagsets_json
nltk.help.upenn_tagset()
```

```
UH: interjection
```

```
UH: interjection
    Goodbye Goody Gosh Wow Jeepers Jee-sus Hubba Hey Kee-reist Oops amen
    huh howdy uh dammit whammo shucks heck anyways whodunnit honey golly
    man baby diddle hush sonuvabitch ...
VB: verb, base form
    ask assemble assess assign assume atone attention avoid bake balkanize
    bank begin behold believe bend benefit bevel beware bless boil bomb
    boost brace break bring broil brush build ...
VBD: verb, past tense
    dipped pleaded swiped regummed soaked tidied convened halted registered
    cushioned exacted snubbed strode aimed adopted belied figgered
    speculated wore appreciated contemplated ...
VBG: verb, present participle or gerund
    telegraphing stirring focusing angering judging stalling lactating
    hankerin' alleging veering capping approaching traveling besieging
    encrypting interrupting erasing wincing ...
VBN: verb, past participle
    multihulled dilapidated aerosolized chaired languished panelized used
    experimented flourished imitated reunifed factored condensed sheared
    unsettled primed dubbed desired ...
VBP: verb, present tense, not 3rd person singular
    predominate wrap resort sue twist spill cure lengthen brush terminate
    appear tend stray glisten obtain comprise detest tease attract
    emphasize mold postpone sever return wag ...
VBZ: verb, present tense, 3rd person singular
    bases reconstructs marks mixes displeases seals carps weaves snatches
    slumps stretches authorizes smolders pictures emerges stockpiles
    seduces fizzes uses bolsters slaps speaks pleads ...
WDT: WH-determiner
    that what whatever which whichever
WP: WH-pronoun
    that what whatever whatsoever which who whom whosoever
WP$: WH-pronoun, possessive
    whose
WRB: Wh-adverb
    how however whence whenever where whereby whereever wherein whereof why
``: opening quotation mark
    ` ``
[nltk_data]   Unzipping help/tagsets_json.zip.
```

```
!pip install nltk spacy
!python -m spacy download en_core_web_sm
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: spacy in /usr/local/lib/python3.12/dist-packages (3.8.11)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages (from nltk) (1.5.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.0.15)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.13)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (8.3.10)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.12/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.5.2)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.4.3)
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (0.20.0)
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.0.2)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (2.32.4)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.12/dist-packages (from spacy) (2
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages (from spacy) (3.1.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-packages (from spacy) (75.2.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from spacy) (25.0)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1,<3
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python3.12/dist-packages (from pydantic!=1.8,!=1.8.1
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->spacy) (
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->sp
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/dist-packages (from requests<3.0.0,>=2.13.0->sp
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->spacy)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python3.12/dist-packages (from thinc<8.4.0,>=8.3.4->
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python3.12/dist-packages (from weasel<0.5.0,>=0.4.2-
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist-packages (from jinja2->spacy) (3.0.3)
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages (from smart-open<8.0.0,>=5.2.1->weasel<0.5.0,>
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-a
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 102.0 MB/s eta 0:00:00
✔ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

```
⚠ Restart to reload dependencies
If you are in a Jupyter or Colab notebook, you may need to restart Python in
order to load all the package's dependencies. You can do this by selecting the
'Restart kernel' or 'Restart runtime' option.
```

```
import nltk
import spacy
import string
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[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!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
True
```

```
# Load medical text file
file_path = "/content/data.txt"
with open(file_path, 'r', encoding='utf-8') as file:
    medical_text = file.read()
medical_text
```

```
'Excision of limbal dermoids. We reviewed the clinical files of 10 patients who had undergone excision of unilateral epibulbar
limbal dermoids. Preoperatively, all of the affected eyes had worse visual acuity (P less than .02) and more astigmatism (P le
ss than .01) than the contralateral eyes. Postoperatively, every patient was cosmetically improved. Of the eight patients for
whom both preoperative and postoperative visual acuity measurements had been obtained, in six it had changed minimally (less t
han or equal to 1 line), and in two it had improved (less than or equal to 2 lines). Surgical complications included persisten
t epithelial defects (40%) and peripheral corneal vascularization and opacity (70%). These complications do not outweigh the c
osmetic and visual benefits of dermoid excision in selected patients. \nBell's palsy. A diagnosis of exclusion. In cases of ac
ute unilateral facial weakness, a careful and systematic evaluation is necessary to identify the cause. Idiopathic fac…'
```

```
from nltk.tokenize import sent_tokenize
sentences = sent_tokenize(medical_text)
```

```
sentences
```

```
['Excision of limbal dermoids.',
 'We reviewed the clinical files of 10 patients who had undergone excision of unilateral epibulbar limbal dermoids.',
 'Preoperatively, all of the affected eyes had worse visual acuity (P less than .02) and more astigmatism (P less than .01)
than the contralateral eyes.',
 'Postoperatively, every patient was cosmetically improved.',
 'Of the eight patients for whom both preoperative and postoperative visual acuity measurements had been obtained, in six it
had changed minimally (less than or equal to 1 line), and in two it had improved (less than or equal to 2 lines).',
 'Surgical complications included persistent epithelial defects (40%) and peripheral corneal vascularization and opacity
(70%).',
 'These complications do not outweigh the cosmetic and visual benefits of dermoid excision in selected patients.',
 "Bell's palsy.",
 'A diagnosis of exclusion.',
 'In cases of acute unilateral facial weakness, a careful and systematic evaluation is necessary to identify the cause.',
 "Idiopathic facial paralysis (Bell's palsy) is a diagnosis of exclusion.",
 'It is also the most common cause of unilateral facial weakness seen by primary care physicians.',
 'The most important aspect of initial treatment is eye protection.',
 'Administration of systemic oral corticosteroids may lessen severity and duration of symptoms.']
```

```
import nltk
from nltk.tokenize import sent_tokenize
nltk.download('punkt_tab') # Download the missing resource
sentences = sent_tokenize(medical_text)
sentences
```

```
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
['Excision of limbal dermoids.',
 'We reviewed the clinical files of 10 patients who had undergone excision of unilateral epibulbar limbal dermoids.',
 'Preoperatively, all of the affected eyes had worse visual acuity (P less than .02) and more astigmatism (P less than .01)
than the contralateral eyes.',
 'Postoperatively, every patient was cosmetically improved.',
 'Of the eight patients for whom both preoperative and postoperative visual acuity measurements had been obtained, in six it
had changed minimally (less than or equal to 1 line), and in two it had improved (less than or equal to 2 lines).',
 'Surgical complications included persistent epithelial defects (40%) and peripheral corneal vascularization and opacity
(70%).',
 'These complications do not outweigh the cosmetic and visual benefits of dermoid excision in selected patients.',
 "Bell's palsy.",
 'A diagnosis of exclusion.',
 'In cases of acute unilateral facial weakness, a careful and systematic evaluation is necessary to identify the cause.',
 "Idiopathic facial paralysis (Bell's palsy) is a diagnosis of exclusion.",
 'It is also the most common cause of unilateral facial weakness seen by primary care physicians.',
 'The most important aspect of initial treatment is eye protection.',
 'Administration of systemic oral corticosteroids may lessen severity and duration of symptoms.']
```

```
from nltk.tokenize import word_tokenize
words = word_tokenize(medical_text)
words
```

```
 'most',
 'important',
 'aspect',
 'of',
 'initial',
 'treatment',
 'is',
 'eye',
 'protection',
 '.',
 'Administration',
 'of',
 'systemic',
 'oral',
 'corticosteroids',
 'may',
 'lessen',
 'severity',
 'and',
 'duration',
 'of',
 'symptoms',
 '.']
```

```
from nltk.corpus import stopwords
stop_words = set(stopwords.words("english"))
filtered_words = []
for word in words:
    if word.lower() not in stop_words and word not in string.punctuation:
        filtered_words.append(word)
filtered_words
```

```
    'corticosteroids'
```

```
 'corticosteroids',
 'may',
 'lessen',
 'severity',
 'duration',
```

```
from nltk.stem import PorterStemmer
porter = PorterStemmer()
porter_stemmed = [porter.stem(word) for word in filtered_words]
porter_stemmed
```

```
 'diagnosi'
 'diagnosi',
 'exclus',
 'also',
 'common',
 'caus',
 'unilater',
 'facial',
 'weak',
 'seen',
 'primari',
 'care',
 'physician',
 'import',
 'aspect',
 'initi',
 'treatment',
 'eye',
 'protect',
 'administr',
 'system',
 'oral',
 'corticosteroid',
 'may',
 'lessen',
 'sever',
 'durat',
 'symptom']
```

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
lemmatized_words = [lemmatizer.lemmatize(word) for word in filtered_words]
lemmatized_words
```

[illegible]

```
 'may',
 'lessen',
 'severity',
 'duration',
 'symptom']
```

```
lemmatizer.lemmatize("worse", pos="a")
```

```
'bad'
```

```
nlp = spacy.load("en_core_web_sm")
doc = nlp(medical_text)
spacy_tokens = []
for token in doc:
    if not token.is_stop and not token.is_punct:
        spacy_tokens.append((token.text, token.lemma_))
spacy_tokens
```

```
 ('10', '10'),
 ('patients', 'patient'),
 ('undergone', 'undergone'),
 ('excision', 'excision'),
 ('unilateral', 'unilateral'),
 ('epibulbar', 'epibulbar'),
 ('limbal', 'limbal'),
 ('dermoids', 'dermoid'),
 ('Preoperatively', 'preoperatively'),
 ('affected', 'affected'),
 ('eyes', 'eye'),
 ('worse', 'bad'),
 ('visual', 'visual'),
 ('acuity', 'acuity'),
 ('P', 'p'),
 ('.02', '.02'),
 ('astigmatism', 'astigmatism'),
 ('P', 'p'),







 ('improved', 'improve'),
 ('patients', 'patient'),
 ('preoperative', 'preoperative'),
 ('postoperative', 'postoperative'),
 ('visual', 'visual'),
 ('acuity', 'acuity'),
 ('measurements', 'measurement'),
 ('obtained', 'obtain'),
 ('changed', 'change'),
 ('minimally', 'minimally'),
 ('equal', 'equal'),
 ('1', '1'),
 ('line', 'line'),
 ('improved', 'improve'),
 ('equal', 'equal'),
 ('2', '2'),
 ('lines', 'line'),
 ('Surgical', 'surgical'),
 ('complications', 'complication'),
 ('included', 'include'),
 ('persistent', 'persistent'),
 ('epithelial', 'epithelial'),
 ('defects', 'defect'),
 ('40', '40'),
 ('peripheral', 'peripheral'),
 ('corneal', 'corneal'),
 ('vascularization', 'vascularization'),
 ('opacity', 'opacity'),
 ('70', '70'),
 ('complications', 'complication'),
 ('outweigh', 'outweigh'),
 ('cosmetic', 'cosmetic'),
 ('visual', 'visual'),
 ('benefits', 'benefit'),
 ('dermoid', 'dermoid'),
```

```
print("Original Word | Porter Stem | Lemma")
print("----------------------------------")
for word in filtered_words[:20]:
    print(word, "|", porter.stem(word), "|", lemmatizer.lemmatize(word))
```

```
Original Word | Porter Stem | Lemma
----------------------------------
```

```
Excision | excis | Excision
limbal | limbal | limbal
dermoids | dermoid | dermoids
reviewed | review | reviewed
clinical | clinic | clinical
files | file | file
10 | 10 | 10
patients | patient | patient
undergone | undergon | undergone
excision | excis | excision
unilateral | unilater | unilateral
epibulbar | epibulbar | epibulbar
limbal | limbal | limbal
dermoids | dermoid | dermoids
Preoperatively | preoper | Preoperatively
affected | affect | affected
eyes | eye | eye
worse | wors | worse
visual | visual | visual
acuity | acuiti | acuity
```

```
NLP="NLP Models are transforming the world rapidly"
```

```
NLP
```

```
'NLP Models are transforming the world rapidly'
```

```
import nltk
nltk.download('punkt')
from nltk.tokenize import word_tokenize
word_tokenize(NLP)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
['NLP', 'Models', 'are', 'transforming', 'the', 'world', 'rapidly']
```

```
from nltk.tokenize import sent_tokenize
sent_tokenize(NLP)
```

```
['NLP Models are transforming the world rapidly']
```

```
nltk.download("stopwords")
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
words_in_quote = word_tokenize(NLP)
words_in_quote
```

```
['NLP', 'Models', 'are', 'transforming', 'the', 'world', 'rapidly']
```

```
stop_words = set(stopwords.words("english"))
filtered_list = []
for word in words_in_quote:
  if word.casefold() not in stop_words:
    filtered_list.append(word)
filtered_list
```

```
['NLP', 'Models', 'transforming', 'world', 'rapidly']
```

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
stemmer = PorterStemmer()
words = word_tokenize(NLP)
stemmed_words = [stemmer.stem(word) for word in words]
stemmed_words
```

```
['nlp', 'model', 'are', 'transform', 'the', 'world', 'rapidli']
```

```
from nltk.stem import SnowballStemmer
snowball = SnowballStemmer(language='english')
words = word_tokenize(NLP)
for word in words:
    print(word,"--->",snowball.stem(word))
```

```
NLP ---> nlp
Models ---> model
are ---> are
transforming ---> transform
the ---> the
world ---> world
rapidly ---> rapid
```

```
from nltk import LancasterStemmer
Lanc = LancasterStemmer()
words = word_tokenize(NLP)
for word in words:
    print(word,"--->",Lanc.stem(word))
```

```
NLP ---> nlp
Models ---> model
are ---> ar
transforming ---> transform
the ---> the
world ---> world
rapidly ---> rapid
```

```
from nltk.stem import RegexpStemmer
regexp = RegexpStemmer('ing|e', min=4)
words = word_tokenize(NLP)
for word in words:
    print(word,"--->",regexp.stem(word))
```

```
NLP ---> NLP
Models ---> Modls
are ---> are
transforming ---> transform
the ---> the
world ---> world
rapidly ---> rapidly
```

```
nltk.download('omw-1.4')
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

```
words = word_tokenize(NLP)
for word in words:
    print(word,"--->",lemmatizer.lemmatize(word))
```

```
NLP ---> NLP
Models ---> Models
are ---> are
transforming ---> transforming
the ---> the
world ---> world
rapidly ---> rapidly
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

```
lemmatizer.lemmatize("worst")
```

```
'worst'
```

```
lemmatizer.lemmatize("worst", pos="a")
```

```
'bad'
```

```
Start coding or generate with AI.
```