Assignment-2.4

```
import nltk
import spacy
from nltk.tokenize import sent_tokenize, word_tokenize
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')
nltk.download("stopwords")
nltk.download('punkt_tab')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
True
```

medical text corpus

```
medical_text = """
Diabetes mellitus is a chronic condition characterized by elevated blood glucose levels.
Patients with diabetes often require insulin therapy and regular monitoring.
Early diagnosis and lifestyle intervention can reduce complications.
"""
```

Sentence Tokenization

```
nltk_sentences = sent_tokenize(medical_text)
print("NLTK Sentences:")
nltk_sentences
```

```
NLTK Sentences:
['\nDiabetes mellitus is a chronic condition characterized by elevated blood glucose levels.',
 'Patients with diabetes often require insulin therapy and regular monitoring.',
 'Early diagnosis and lifestyle intervention can reduce complications.']
```

Word Tokenization

```
nltk_words = word_tokenize(medical_text)
print("NLTK Words:")
nltk_words
```

```
NLTK Words:
['Diabetes',
 'mellitus',
 'is',
 'a',
 'chronic',
 'condition',
 'characterized',
 'by',
 'elevated',
 'blood',
 'glucose',
 'levels',
 '.',
 'Patients',
 'with',
 'diabetes',
 'often',
 'require',
 'insulin',
 'therapy',
 'and',
 'regular',
 'monitoring',
 '.',
 'Early',
 'diagnosis',
 'and',
```

```
 'lifestyle',
 'intervention',
 'can',
 'reduce',
 'complications',
 '.']
```

Stemming

```
stemmer = PorterStemmer()
words = nltk_words
stemmed_words = [stemmer.stem(word) for word in words]
stemmed_words
```

```
['diabet',
 'mellitu',
 'is',
 'a',
 'chronic',
 'condit',
 'character',
 'by',
 'elev',
 'blood',
 'glucos',
 'level',
 '.',
 'patient',
 'with',
 'diabet',
 'often',
 'requir',
 'insulin',
 'therapi',
 'and',
 'regular',
 'monitor',
 '.',
 'earli',
 'diagnosi',
 'and',
 'lifestyl',
 'intervent',
 'can',
 'reduc',
 'complic',
 '.']
```

Lemmatization

```
lemmatizer = WordNetLemmatizer()
nltk_lemmas = [lemmatizer.lemmatize(token) for token in nltk_words]
print("NLTK Lemmatization Output:")
nltk_lemmas
```

```
NLTK Lemmatization Output:
['Diabetes',
 'mellitus',
 'is',
 'a',
 'chronic',
 'condition',
 'characterized',
 'by',
 'elevated',
 'blood',
 'glucose',
 'level',
 '.',
 'Patients',
 'with',
 'diabetes',
 'often',
 'require',
 'insulin',
 'therapy',
 'and',
 'regular',
 'monitoring',
 '.',
 'Early',
 'diagnosis',
 'and',
```

```
 'lifestyle',
 'intervention',
 'can',
 'reduce',
 'complication',
 '.']
```

Why Lemmatization is Critical in Healthcare NLP

Key Observations:

Stemming:

- Rule-based and aggressive
- Can truncate medical terms incorrectly
- Produces non-dictionary words (e.g., "diagnos") **Lemmatization:**
- Context-aware
- Preserves medically valid root forms
- Produces dictionary-standard terminology

Importance in Healthcare NLP

```
Lemmatization is critical in healthcare NLP because:
```

Clinical Accuracy: Medical terms must retain semantic correctness. Incorrect stems can distort diagnoses, symptoms, or treatments.

Interoperability: Healthcare NLP systems integrate with ontologies (e.g., ICD, SNOMED). Lemmas align better with standardized vocabularies.

Patient Safety: Misinterpretation of clinical language can lead to incorrect insights or automated decisions.

Improved Model Performance: Lemmatized tokens improve downstream tasks such as:

- Clinical entity recognition
- Medical document classification
- Clinical decision support systems

Thank you

Assignment in ppt

```
text=" NLP models are transforming the world rapidly! "
```

Word tokens

```
nltk_text = word_tokenize(text)
print("NLTK Words:")
nltk_text
```

```
NLTK Words:
['NLP', 'models', 'are', 'transforming', 'the', 'world', 'rapidly', '!']
```

Stemmed words

```
stemmer = PorterStemmer()
words = nltk_text
stemmed_words = [stemmer.stem(word) for word in words]
stemmed_words
```

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

Lemmatized words

```
lemmatizer = WordNetLemmatizer()
nltk_lemmas = [lemmatizer.lemmatize(token) for token in nltk_text]
```

```
print("NLTK Lemmatization Output:")
nltk_lemmas
```

```
NLTK Lemmatization Output:
['NLP', 'model', 'are', 'transforming', 'the', 'world', 'rapidly', '!']
```

Thank you

github repositorie

Tokenizing

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

Tokenizing by words:

```
word_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']
```

Tokenizing by sentence:

```
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']
```

Filtering Stop Words

```
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']
```

Stemming

```
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
```

```
from nltk.stem import RegexpStemmer
regexp = RegexpStemmer('ing|s|e|able', min=4)
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
```

Lemmatization

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

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
The ---> The
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
Indias ---> Indias
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'
```

Comparison

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

```
Word                Porter Stemmer      Snowball Stemmer    Lancaster Stemmer            Regexp Stemmer
friend              friend              friend              friend                       frind
friendship          friendship          friendship          friend                       frindhip
friends             friend              friend              friend                       frind
friendships         friendship          friendship          friend                       frindhip
```

Tagging Parts of **Speech**

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

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[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'),
('.', '.'),
('There', 'EX'),
('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')
nltk.help.upenn_tagset()
```

```
    multinulled 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] Downloading package tagsets_json to /root/nltk_data...
[nltk_data]   Unzipping help/tagsets_json.zip.
```

Chunking

```
from nltk.tokenize import word_tokenize
lotr_quote = "It's a dangerous business, Frodo, going out your door."
words_in_lotr_quote = word_tokenize(lotr_quote)
lotr_pos_tags = nltk.pos_tag(words_in_lotr_quote)
grammar = "NP: {<DT>?<JJ>*<NN.*>+}" # Corrected grammar for Noun Phrases
chunk_parser = nltk.RegexpParser(grammar)
tree = chunk_parser.parse(lotr_pos_tags)
print(tree)
```

```
(S
  It/PRP
  's/VBZ
  (NP a/DT dangerous/JJ business/NN)
  ,/,
  (NP Frodo/NNP)
  ,/,
  going/VBG
  out/RP
  your/PRP$
  (NP door/NN)
  ./.)
```

Chinking

```
from nltk.tokenize import word_tokenize
lotr_quote = "It's a dangerous business, Frodo, going out your door."
words_in_lotr_quote = word_tokenize(lotr_quote)
lotr_pos_tags = nltk.pos_tag(words_in_lotr_quote)
grammar = """
          CHUNK: {<.*>+}    # Chunk everything
                 }<,>{        # Chink out commas
          """
chunk_parser = nltk.RegexpParser(grammar)
tree = chunk_parser.parse(lotr_pos_tags)
print(tree)
```

```
(S
  (CHUNK It/PRP 's/VBZ a/DT dangerous/JJ business/NN)
  ,/,
  (CHUNK Frodo/NNP)
  ,/,
  (CHUNK going/VBG out/RP your/PRP$ door/NN ./.))
```

Using Named Entity Recognition (NER)

```
def extract_ne(quote):
  words = word_tokenize(quote)
  tags = nltk.pos_tag(words)
  tree = nltk.ne_chunk(tags, binary=True)
  return set(" ".join(i[0] for i in t)
             for t in tree
             if hasattr(t, "label") and t.label() == "NE"
             )
```

```
nltk.download('maxent_ne_chunker')
nltk.download('words')
nltk.download('maxent_ne_chunker_tab') # Added missing download
extract_ne("I am Venkatamana, from India, working as professor")
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading package maxent_ne_chunker_tab to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker_tab.zip.
{'India', 'Venkatamana'}
```

Close

```
# **Thank you**
```

Thank you