TASK1:TEXT DOC

```
import nltk
nltk.download('punkt')
```

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

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

```
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
True
```

```
import nltk
nltk.download('punkt_tab') # Added to download the mi
sentence = "Students are learning Natural Language Pr
tokens = nltk.word_tokenize(sentence)
pos_tags = nltk.pos_tag(tokens)
```

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

```
print(pos_tags)
```

```
), ('learning', 'VBG'), ('Natural', 'NNP'), ('Language', 'NNP'), ('Processing', '
```

```
import spacy
```

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

```
doc = nlp("Students are learning Natural Language Processing")
for token in doc:
    print(token.text, token.pos_)
```

```
Students NOUN
are AUX
learning VERB
Natural PROPN
Language PROPN
Processing NOUN
```

```
nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying a startup in India.")

for token in doc:
    print(token.text, token.pos_, token.tag_)
```

```
Apple PROPN NNP
is AUX VBZ
looking VERB VBG
at ADP IN
buying VERB VBG
a DET DT
startup NOUN NN
in ADP IN
India PROPN NNP
. PUNCT .
```

```
from collections import Counter
nlp = spacy.load("en_core_web_sm")
text = """Loving the new AI features

 #AI #MachineLearning"""
doc = nlp(text)
nouns = []
verbs = []
for token in doc:
    if token.pos_ in ["NOUN", "PROPN"]:
        nouns.append(token.text)
    elif token.pos_ == "VERB":
        verbs.append(token.text)
noun_freq = Counter(nouns)
verb_freq = Counter(verbs)
print("Noun Frequency:", noun_freq)
print("Verb Frequency:", verb_freq)
```

```
Noun Frequency: Counter({'AI': 2, 'MachineLearning': 1})
Verb Frequency: Counter({'Loving': 1, 'features': 1})
```

TASK2)TAGGING OF PARTS OF SPEECH

```
nltk.download('averaged_perceptron_tagger')
from nltk.tokenize import word_tokenize
SRUniversity="""The SR University campus is located in Ananthasagar village of
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 I

words = word_tokenize("SRUniversity") # Changed to a string
nltk.pos_tag(words)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
```

```
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[('SRUniversity', 'NN')]
```

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

```
[('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'),
 ('.', '.')]
```

```
import nltk
nltk.download('tagsets')
nltk.download('tagsets_json') # Added to download the missing resource
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
```

TASK3) ASSIGNMENT_3.2

TEXT PROCESSING WITH NLTK AND SPACY

AIM:Understand POS tagging challenges in informal, noisy text.

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

```
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: spacy in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: joblib in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/pyt
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/pyt
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/pytho
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.12/
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.1
Requirement already satisfied: thinc<8.4.0,>=8.3.4 in /usr/local/lib/python3.12/
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.12
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.12/
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3
Requirement already satisfied: weasel<0.5.0,>=0.4.2 in /usr/local/lib/python3.12
Requirement already satisfied: typer-slim<1.0.0,>=0.3.0 in /usr/local/lib/python
Requirement already satisfied: numpy>=1.19.0 in /usr/local/lib/python3.12/dist-p
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/python3
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/loca
Requirement already satisfied: jinja2 in /usr/local/lib/python3.12/dist-packages
Requirement already satisfied: setuptools in /usr/local/lib/python3.12/dist-pack
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.
Requirement already satisfied: pydantic-core==2.41.4 in /usr/local/lib/python3.1
Requirement already satisfied: typing-extensions>=4.14.1 in /usr/local/lib/pytho
Requirement already satisfied: typing-inspection>=0.4.2 in /usr/local/lib/python
Requirement already satisfied: charset_normalizer<4,>=2 in /usr/local/lib/python
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.12/dist-pa
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.12/d
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.12/d
Requirement already satisfied: blis<1.4.0,>=1.3.0 in /usr/local/lib/python3.12/d
Requirement already satisfied: confection<1.0.0,>=0.0.1 in /usr/local/lib/python
Requirement already satisfied: cloudpathlib<1.0.0,>=0.7.0 in /usr/local/lib/pyth
Requirement already satisfied: smart-open<8.0.0,>=5.2.1 in /usr/local/lib/python
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.12/dist
Requirement already satisfied: wrapt in /usr/local/lib/python3.12/dist-packages
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_cor
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 93.9 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.
```

```python
import json
import nltk
import spacy
from nltk.tokenize import TweetTokenizer
from nltk import pos_tag
from collections import Counter

nltk.download('averaged_perceptron_tagger')
```

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

```python
file_path = ("/content/enriched_posts.json")
with open(file_path, "r", encoding="utf-8") as f:
    data = json.load(f)

type(data), len(data)
```

```
(list, 60)
```

1)Extract text from posts

```python
texts = []

for item in data:
    if 'text' in item:
        texts.append(item['text'])
    elif 'caption' in item:
        texts.append(item['caption'])
    elif 'content' in item:
        texts.append(item['content'])
texts[:5]
```

```
["Just saw a LinkedIn Influencer with 'Organic Growth' written in the profile
with 65K+ followers claiming that he can help you in growing your platform,
copying the posts from other influencers.",
 "Jobseekers, this one’s for you.\n Every application, every interview, every
follow-up… the pressure is immense.\n And I know what you're thinking: Am I not
good enough? \n But let me tell you, this isn’t about you or your skills. It’s
about a broken system where 60% of applicants never hear back. \n Your mental
health is not worth sacrificing for a system that doesn’t acknowledge your
worth. \n Please remember, taking care of yourself is the real priority. \n
Your dream job will come, but for now, breathe. 🌻 ",
 'Looking for jobs on LinkedIn is like online dating: Full of promises, but in
the end, you’re just left ghosted.',
 "LinkedIn scams be like: 'Congratulations, you've been selected for a role you
didn’t even apply for!' \n The catch? Pay Rs. 50,000 for the honor.",
 "sapne dekhna achi baat hai,\nlekin job ka sapna dekh ke 'interested'
likhna,\nyeh toh achi baat nahi hai na?"]
```

2)Tokenize informal text (hashtags, emojis, slang)

```
# Tokenize informal text (hashtags, emojis, slang)
tokenizer = TweetTokenizer(
    preserve_case=False,
    strip_handles=True,
    reduce_len=True
)

sample_tokens = tokenizer.tokenize(texts[0])
sample_tokens
```

```
['just',
 'saw',
 'a',
 'linkedin',
 'influencer',
 'with',
 "'",
 'organic',
 'growth',
 "'",
 'written',
 'in',
 'the',
 'profile',
 'with',
 '65k',
 '+',
 'followers',
 'claiming',
 'that',
 'he',
 'can',
 'help',
 'you',
 'in',
 'growing',
 'your',
 'platform',
 ',',
 'copying',
 'the',
 'posts',
 'from',
 'other',
 'influencers',
 '.']
```

3)POS tagging using NLTK

```
nltk_pos = pos_tag(sample_tokens)
nltk_pos
```

```
[('just', 'RB'),
 ('saw', 'VBD'),
 ('a', 'DT'),
 ('linkedin', 'JJ'),
 ('influencer', 'NN'),
 ('with', 'IN'),
 ("'", "''"),
 ('organic', 'JJ'),
 ('growth', 'NN'),
 ("'", "''"),
 ('written', 'VBN'),
 ('in', 'IN'),
 ('the', 'DT'),
 ('profile', 'NN'),
 ('with', 'IN'),
 ('65k', 'CD'),
 ('+', 'JJ'),
 ('followers', 'NNS'),
 ('claiming', 'VBG'),
 ('that', 'IN'),
 ('he', 'PRP'),
 ('can', 'MD'),
 ('help', 'VB'),
 ('you', 'PRP'),
 ('in', 'IN'),
 ('growing', 'VBG'),
 ('your', 'PRP$'),
 ('platform', 'NN'),
 (',', ','),
 ('copying', 'VBG'),
 ('the', 'DT'),
 ('posts', 'NNS'),
 ('from', 'IN'),
 ('other', 'JJ'),
 ('influencers', 'NNS'),
 ('.', '.')]
```

4)POS tagging using spaCy

```
nlp = spacy.load("en_core_web_sm")
doc = nlp(texts[0])

[(token.text, token.pos_) for token in doc]
```

```
[('Just', 'ADV'),
 ('saw', 'VERB'),
 ('a', 'DET'),
 ('LinkedIn', 'NOUN'),
 ('Influencer', 'NOUN'),
 ('with', 'ADP'),
 ("'", 'PUNCT'),
 ('Organic', 'PROPN'),
 ('Growth', 'PROPN'),
 ("'", 'PUNCT'),
 ('written', 'VERB'),
 ('in', 'ADP'),
 ('the', 'DET'),
 ('profile', 'NOUN'),
 ('with', 'ADP'),
 ('65K+', 'PROPN'),
 ('followers', 'NOUN'),
 ('claiming', 'VERB'),
 ('that', 'SCONJ'),
 ('he', 'PRON'),
 ('can', 'AUX'),
 ('help', 'VERB'),
 ('you', 'PRON'),
 ('in', 'ADP'),
 ('growing', 'VERB'),
 ('your', 'PRON'),
 ('platform', 'NOUN'),
 (',', 'PUNCT'),
 ('copying', 'VERB'),
 ('the', 'DET'),
 ('posts', 'NOUN'),
 ('from', 'ADP'),
 ('other', 'ADJ'),
 ('influencers', 'NOUN'),
 ('.', 'PUNCT')]
```

5)Compare tag sets (slang & abbreviations)

```
# This is formatted as code
```

```
print("NLTK POS Tags:")
print(nltk_pos)

print("\nspaCy POS Tags:")
print([(token.text, token.pos_) for token in doc])
```

```
NLTK POS Tags:
[('just', 'RB'), ('saw', 'VBD'), ('a', 'DT'), ('linkedin', 'JJ'), ('influencer',

spaCy POS Tags:
[('Just', 'ADV'), ('saw', 'VERB'), ('a', 'DET'), ('LinkedIn', 'NOUN'), ('Influen
```

NLTK

```
nltk_nouns = [w for w, t in nltk_pos if t.startswith('NN')]
nltk_verbs = [w for w, t in nltk_pos if t.startswith('VB')]
```

spaCy

```
spacy_nouns = [t.text for t in doc if t.pos_ == 'NOUN']
spacy_verbs = [t.text for t in doc if t.pos_ == 'VERB']
```

```
print("Top NLTK Nouns:", Counter(nltk_nouns).most_common(10))
print("Top NLTK Verbs:", Counter(nltk_verbs).most_common(10))
```

```
Top NLTK Nouns: [('influencer', 1), ('growth', 1), ('profile', 1), ('followers',
Top NLTK Verbs: [('saw', 1), ('written', 1), ('claiming', 1), ('help', 1), ('grc
```

Close

```
Discussion

Informal social media text
contains emojis, hashtags,
slang, and abbreviations
that make POS tagging
challenging. NLTK often
assigns generic or
incorrect tags
to slang and emojis, while
```

Discussion

Informal social media text contains emojis, hashtags, slang, and abbreviations that make POS tagging challenging. NLTK often assigns generic or incorrect tags to slang and emojis, while spaCy performs better due to contextual embeddings. However,

spaCy performs better due to contextual embeddings. However, both tools struggle with noisy text. Hashtags are usually treated as nouns, helping in topic extraction, while verbs represent opinions and actions.	both tools struggle with noisy text. Hashtags are usually treated as nouns, helping in topic extraction, while verbs represent opinions and actions.