TASK-1

ANALYSING THE SENTENCES USING PARTS OF SPEECH AND NLTK(LIBRARY)

```
import nltk
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('punkt_tab') # Added to resolve the LookupError
sentence = "Students are learning Natural Language Processing"
tokens = nltk.word_tokenize(sentence)
pos_tags = nltk.pos_tag(tokens)
print(pos_tags)
```

```
[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]   Package averaged_perceptron_tagger_eng is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[('Students', 'NNS'), ('are', 'VBP'), ('learning', 'VBG'), ('Natural', 'NNP'), ('Langu
```

ANALYSING THE SENTENCES USING PARTS OF SPEECH AND SPACY(LIBRARY)

```
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
```

```
import spacy
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 .
```

```
import spacy
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, '�': 2, 'MachineLearning': 1})
Verb Frequency: Counter({'Loving': 1})
```

TASK-2

PRACTISING THE EXAMPLES FROM GITHUB LINK

```
nltk.download('averaged_perceptron_tagger')
from nltk.tokenize import word_tokenize
SRUniversity="""The SR University campus is located in Ananthasagar village of Hasanp
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 Incubat
words = word_tokenize(SRUniversity)
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!
[('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', '$'),
```

```python
import nltk
nltk.download('tagsets')
nltk.download('tagsets_json') # Added to resolve the LookupError
nltk.help.upenn_tagset()
```

```
    speculated were appreciated contemplated
    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] Downloading package tagsets to /root/nltk_data...
[nltk_data]   Package tagsets is already up-to-date!
[nltk_data] Downloading package tagsets_json to /root/nltk_data...
[nltk_data]   Unzipping help/tagsets_json.zip.
```

TASK-3

ASSIGNMENT 3

IMPORTED ALL THE LIBRARY TO READ THE CSV FILE

```
# ============================================
# NLP on Tweets/Captions: Tokenization & POS
# Using NLTK and spaCy
# ============================================

# -----------------------
# 1. Install & Imports
# -----------------------
!pip install -q spacy
!python -m spacy download en_core_web_sm

import nltk
from nltk.tokenize import TweetTokenizer
from nltk import pos_tag
from collections import Counter
import spacy
import pandas as pd

# Newer NLTK versions use these resource names for tagging
nltk.download("punkt_tab")        # tokenizer models [web:3]
nltk.download("averaged_perceptron_tagger_eng")  # POS tagger [web:3]

# Load spaCy model
```

```
nlp = spacy.load("en_core_web_sm")  # provides tokenization + POS tagging [web:6]
```

```
Collecting en-core-web-sm==3.8.0
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_s
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 12.8/12.8 MB 52.3 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.
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger_eng.zip.
```

EXTRACTED THE TEXT FROM JSON FILE AND PRINTED THE TEXT

```
import json

# 1. Path to your JSON file
file_path = "/content/enriched_posts.json"   # or "/content/enriched_posts.json" in C

# 2. Load JSON
with open(file_path, "r", encoding="utf-8") as f:
    data = json.load(f)   # dict or list [web:32][web:38]

print("Type of data:", type(data))

# 3. Extract ALL texts into a list
texts = []
count = 0

# Case A: JSON is a list of objects
if isinstance(data, list):
    for item in data:
        text = item.get("text", "")
        texts.append(text) # Collect text
        print(text)
        count += 1
        if count == 5:
            break

# Case B: JSON is a dict with list under a key, e.g. "posts"
elif isinstance(data, dict):
    for key, value in data.items():
        if isinstance(value, list):
            for item in value:
                text = item.get("text", "")
                texts.append(text) # Collect text
                print(text)
                count += 1
                if count == 5:
                    break
        if count == 5:
            break
```

```
# Ensure texts list is created even if count is less than 5
# If data is large, this will store all texts, then the tokenization will use `texts`
# If only a sample is needed, a slice of `texts` can be used later.
```

```
Type of data: <class 'list'>
Just saw a LinkedIn Influencer with 'Organic Growth' written in the profile with 65K+
Jobseekers, this one’s for you.
 Every application, every interview, every follow-up… the pressure is immense.
 And I know what you're thinking: Am I not good enough?
 But let me tell you, this isn’t about you or your skills. It’s about a broken system
 Your mental health is not worth sacrificing for a system that doesn’t acknowledge you
 Please remember, taking care of yourself is the real priority.
 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,
LinkedIn scams be like: 'Congratulations, you've been selected for a role you didn’t e
 The catch? Pay Rs. 50,000 for the honor.
sapne dekhna achi baat hai,
lekin job ka sapna dekh ke 'interested' likhna,
yeh toh achi baat nahi hai na?
```

TOKENISATION METHOD

```
tweet_tokenizer = TweetTokenizer(
    preserve_case=False,  # lowercase
    strip_handles=False,  # keep @mentions
    reduce_len=True       # "cooooool" -> "coool"
)

def tokenize_texts(text_list):
    tokenized = []
    for t in text_list:
        tokens = tweet_tokenizer.tokenize(t)
        tokenized.append(tokens)
    return tokenized

tokenized_texts = tokenize_texts(texts)

print("=== Example tokenization ===")
for original, tokens in zip(texts[:3], tokenized_texts[:3]):
    print("TEXT :", original)
    print("TOKENS:", tokens)
    print("-" * 60)
```

```
=== Example tokenization ===
TEXT : Just saw a LinkedIn Influencer with 'Organic Growth' written in the profile wit
TOKENS: ['just', 'saw', 'a', 'linkedin', 'influencer', 'with', "'", 'organic', 'growth
------------------------------------------------------------
TEXT : Jobseekers, this one’s for you.
 Every application, every interview, every follow-up… the pressure is immense.
 And I know what you're thinking: Am I not good enough?
 But let me tell you, this isn’t about you or your skills. It’s about a broken system
 Your mental health is not worth sacrificing for a system that doesn’t acknowledge you
 Please remember, taking care of yourself is the real priority.
 Your dream job will come, but for now, breathe. 🌻
TOKENS: ['jobseekers', ',', 'this', 'one', '’', 's', 'for', 'you', '.', 'every', 'appl
------------------------------------------------------------
TEXT : Looking for jobs on LinkedIn is like online dating: Full of promises, but in th
TOKENS: ['looking', 'for', 'jobs', 'on', 'linkedin', 'is', 'like', 'online', 'dating',
------------------------------------------------------------
```

PARTS OF SPEECH USING LIBRARY NLTK

```
def pos_tag_nltk(tokenized_list):
    all_tagged = []
    for tokens in tokenized_list:
        tagged = pos_tag(tokens)
        all_tagged.append(tagged)
    return all_tagged

nltk_tagged = pos_tag_nltk(tokenized_texts)

print("=== NLTK POS tags (sample) ===")
for text, tagged in zip(texts[:3], nltk_tagged[:3]):
    print("TEXT :", text)
    print("TAGS :", tagged)
    print("-" * 60)
```

```
=== NLTK POS tags (sample) ===
TEXT : Just saw a LinkedIn Influencer with 'Organic Growth' written in the profile wit
TAGS : [('just', 'RB'), ('saw', 'VBD'), ('a', 'DT'), ('linkedin', 'JJ'), ('influencer'
------------------------------------------------------------
TEXT : Jobseekers, this one’s for you.
 Every application, every interview, every follow-up… the pressure is immense.
 And I know what you're thinking: Am I not good enough?
 But let me tell you, this isn’t about you or your skills. It’s about a broken system
 Your mental health is not worth sacrificing for a system that doesn’t acknowledge you
 Please remember, taking care of yourself is the real priority.
 Your dream job will come, but for now, breathe. 🌻
TAGS : [('jobseekers', 'NNS'), (',', ','), ('this', 'DT'), ('one', 'CD'), ('’', 'NN'),
------------------------------------------------------------
TEXT : Looking for jobs on LinkedIn is like online dating: Full of promises, but in th
TAGS : [('looking', 'VBG'), ('for', 'IN'), ('jobs', 'NNS'), ('on', 'IN'), ('linkedin',
------------------------------------------------------------
```

PARTS OF SPEECH USING SPACY LIBRARY

```
def pos_tag_spacy(text_list):
    all_docs = []
    for t in text_list:
        doc = nlp(t)
        tags = [(token.text, token.pos_, token.tag_) for token in doc]
        all_docs.append(tags)
    return all_docs

spacy_tagged = pos_tag_spacy(texts)

print("=== spaCy POS tags (sample) ===")
for text, tagged in zip(texts[:3], spacy_tagged[:3]):
    print("TEXT :", text)
    print("TAGS :", tagged)
    print("-" * 60)
```

```
=== spaCy POS tags (sample) ===
TEXT : Just saw a LinkedIn Influencer with 'Organic Growth' written in the profile wit
TAGS : [('Just', 'ADV', 'RB'), ('saw', 'VERB', 'VBD'), ('a', 'DET', 'DT'), ('LinkedIn'
------------------------------------------------------------
TEXT : Jobseekers, this one’s for you.
 Every application, every interview, every follow-up… the pressure is immense.
 And I know what you're thinking: Am I not good enough?
 But let me tell you, this isn’t about you or your skills. It’s about a broken system
 Your mental health is not worth sacrificing for a system that doesn’t acknowledge you
 Please remember, taking care of yourself is the real priority.
 Your dream job will come, but for now, breathe. 🌻
TAGS : [('Jobseekers', 'NOUN', 'NNS'), (',', 'PUNCT', ','), ('this', 'DET', 'DT'), ('o
------------------------------------------------------------
TEXT : Looking for jobs on LinkedIn is like online dating: Full of promises, but in th
TAGS : [('Looking', 'VERB', 'VBG'), ('for', 'ADP', 'IN'), ('jobs', 'NOUN', 'NNS'), ('o
------------------------------------------------------------
```

COMPARING THE TAG LINES

```
slang_terms = {
    "lol", "lit", "broooo", "rn", "tmrw", "op", "af",
    "omg", "bussin", "fr", "lowkey", "meh"
}

def collect_nltk_slang_tags(nltk_tagged_data):
    slang_tags = {}
    for sent in nltk_tagged_data:
        for token, tag in sent:
            tok = token.lower()
            if tok in slang_terms:
                slang_tags.setdefault(tok, set()).add(tag)
    return slang_tags

def collect_spacy_slang_tags(spacy_tagged_data):
    slang_tags = {}
    for sent in spacy_tagged_data:
        for token, coarse, fine in sent:
            tok = token.lower()
            if tok in slang_terms:
                slang_tags.setdefault(tok, set()).add(f"{coarse}|{fine}")
    return slang_tags

nltk_slang = collect_nltk_slang_tags(nltk_tagged)
spacy_slang = collect_spacy_slang_tags(spacy_tagged)

rows = []
for term in sorted(slang_terms):
    rows.append({
        "slang": term,
        "NLTK_tags": ", ".join(sorted(nltk_slang.get(term, []))) or "NOT_FOUND",
        "spaCy_tags": ", ".join(sorted(spacy_slang.get(term, []))) or "NOT_FOUND"
    })

slang_df = pd.DataFrame(rows)
print("=== Slang / abbreviation POS comparison (NLTK vs spaCy) ===")
display(slang_df)
```

```
=== Slang / abbreviation POS comparison (NLTK vs spaCy) ===
```

	slang	NLTK_tags	spaCy_tags
0	af	NOT_FOUND	NOT_FOUND
1	broooo	NOT_FOUND	NOT_FOUND
2	bussin	NOT_FOUND	NOT_FOUND
3	fr	NOT_FOUND	NOT_FOUND
4	lit	NOT_FOUND	NOT_FOUND
5	lol	NOT_FOUND	NOT_FOUND
6	lowkey	NOT_FOUND	NOT_FOUND
7	meh	NOT_FOUND	NOT_FOUND
8	omg	NOT_FOUND	NOT_FOUND
9	op	NOT_FOUND	NOT_FOUND
10	rn	NOT_FOUND	NOT_FOUND
11	tmrw	NOT_FOUND	NOT_FOUND

Next steps: Generate code with slang_df New interactive sheet

FREQUENCY OF NOUNS AND VERBS

```
NOUN_TAGS_NLTK = {"NN", "NNS", "NNP", "NNPS"}
VERB_TAGS_NLTK = {"VB", "VBD", "VBG", "VBN", "VBP", "VBZ"}

def extract_nouns_verbs_nltk(nltk_tagged_data):
    noun_freq = Counter()
    verb_freq = Counter()
    for sent in nltk_tagged_data:
        for token, tag in sent:
            tok = token.lower()
            # treat hashtags as topics
            if tag in NOUN_TAGS_NLTK or tok.startswith("#"):
                noun_freq[tok] += 1
            if tag in VERB_TAGS_NLTK:
                verb_freq[tok] += 1
    return noun_freq, verb_freq

nltk_nouns, nltk_verbs = extract_nouns_verbs_nltk(nltk_tagged)

print("=== NLTK noun/topic frequency ===")
print(nltk_nouns.most_common(15))
print("\n=== NLTK verb/action frequency ===")
print(nltk_verbs.most_common(15))

# spaCy: Universal POS tags [web:61][web:62]
NOUN_POS_SPACY = {"NOUN", "PROPN"}
VERB_POS_SPACY = {"VERB", "AUX"}

def extract_nouns_verbs_spacy(spacy_tagged_data):
    noun_freq = Counter()
    verb_freq = Counter()
    for sent in spacy_tagged_data:
        for token, coarse, fine in sent:
            tok = token.lower()
            if coarse in NOUN_POS_SPACY or tok.startswith("#"):
                noun_freq[tok] += 1
            if coarse in VERB_POS_SPACY:
                verb_freq[tok] += 1
    return noun_freq, verb_freq

spacy_nouns, spacy_verbs = extract_nouns_verbs_spacy(spacy_tagged)

print("\n=== spaCy noun/topic frequency ===")
print(spacy_nouns.most_common(15))
print("\n=== spaCy verb/action frequency ===")
print(spacy_verbs.most_common(15))
```

```
=== NLTK noun/topic frequency ===
[('’', 2), ('system', 2), ('t', 2), ('job', 2), ('baat', 2), ('hai', 2), ('influencer'

=== NLTK verb/action frequency ===
[('is', 4), ('’', 3), ('saw', 1), ('written', 1), ('claiming', 1), ('help', 1), ('grow

=== spaCy noun/topic frequency ===
[('linkedin', 3), ('system', 2), ('job', 2), ('achi', 2), ('baat', 2), ('hai', 2), ('i

=== spaCy verb/action frequency ===
[('is', 5), ('’s', 2), ('saw', 1), ('written', 1), ('claiming', 1), ('can', 1), ('help
```