NATURAL LANGUAGE PROCESSING

EXPT 1

```
!pip install nltk
!pip install spacy
!pip install transformers
!pip install scikit-learn
!pip install stanza
```

```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')

import spacy

import stanza
stanza.download('en')
```

Show hidden output

EXPT 2

To study Preprocessing of text

- Tokenization: Uses NLTK to tokenize the text into sentences and words.
- Filtration: Removes punctuation using isalpha() to filter out non-alphabetic characters.
- Script Validation: Checks if each word is valid by ensuring it contains only alphabetic characters.
- Stop Word Removal: Removes common stopwords from the text using NLTK's built-in English stopword list.
- Stemming: Applies Porter's Stemming Algorithm to reduce words to their root forms.

```
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
import string

text = """Natural Language Processing is a sub-field of linguistics, computer science, and artificial intelligence. It is concerned with t

print("=== Tokenization ===")
words = word_tokenize(text)
print("Word Tokens:", words)

print("\n=== Filtration ===")
words_filtered = [word for word in words if word.isalpha()]
print("Filtered Words (Without Punctuation):", words_filtered)

print("\n=== Script Validation ===")
valid_script = [word for word in words_filtered if word.encode().isalpha()]
print("Valid Script Words:", valid_script)

print("\n=== Stop Word Removal ===")
stop_words = set(stopwords.words('english'))
filtered_text = [word for word in valid_script if word.lower() not in stop_words]
print("Text After Stop Word Removal:", filtered_text)

print("\n=== Stemming ===")
ps = PorterStemmer()
```

```
stemmed_words = [ps.stem(word) for word in filtered_text]
print("Stemmed Words:", stemmed_words)
```

Show hidden output

EXPT 3

To perform Morphological Analysis

- Lemmatized Words: Returns the meaningful root forms, e.g., "play" from "playing" or "played."
- POS Tags: Returns part of speech (like noun, verb, etc.) for each word in the sentence.
- Stemmed Words: Returns the base form of words by cutting off suffixes, e.g., "play" from "playing" or "cat" from "cats."

```
import spacy
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from nltk.corpus import wordnet

nlp = spacy.load("en_core_web_sm")
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
```

Show hidden output

```
text = "The cats are playing in the garden. They have played there before."

def english_lemmatizer(text):
    doc = nlp(text)
    return [token.lemma_ for token in doc]
print("Lemmatized Words:", english_lemmatizer(text))

def pos_tagger(text):
    tokens = word_tokenize(text)
    return nltk.pos_tag(tokens)
print("POS Tags:", pos_tagger(text))

stemmer = PorterStemmer()

def english_stemmer(text):
    tokens = word_tokenize(text)
    return [stemmer.stem(token) for token in tokens]
print("Stemmed Words:", english_stemmer(text))
```

Show hidden output

EXPT 4

To study N-Gram Language Model

```
from nltk.util import ngrams
from collections import defaultdict

corpus = [
    "मैं स्कूल जा रहा हूँ।",
    "वह बहुत खुश है।",
    "बिल्ली घर के अंदर है।",
    "हम सब काम कर रहे हैं।",
    "तुम कहाँ जा रहे हो?",
    "यहाँ बहुत गर्मी है।",
    "वह जल्दी आ जाएगा।",
    "मैं तुम्हें देख सकता हूँ।",
```

```
    "हम मिलेंगे।",
    "तुम्हें क्या चाहिए?"
]

tokenized_sentences = [nltk.word_tokenize(sentence) for sentence in corpus]

def calculate_ngram_probabilities(n, tokenized_sentences):
    counts = defaultdict(lambda: defaultdict(int))
    for sentence in tokenized_sentences:
        for ngram in ngrams(sentence, n, pad_left=True, pad_right=True, left_pad_symbol='<s>', right_pad_symbol='</s>'):
            counts[ngram[:-1]][ngram[-1]] += 1
    return {context: {word: count/sum(words.values()) for word, count in words.items()} for context, words in counts.items()}

def display_ngram_probabilities(n, input_sentence, ngram_probs):
    tokens = nltk.word_tokenize(input_sentence)
    for ngram in ngrams(tokens, n, pad_left=True, pad_right=True, left_pad_symbol='<s>', right_pad_symbol='</s>'):
        context, word = ngram[:-1], ngram[-1]
        prob = ngram_probs.get(context, {}).get(word, 0)
        print(f"P({word} | {' '.join(context)}) = {prob:.4f}")


bigram_probs = calculate_ngram_probabilities(2, tokenized_sentences)
trigram_probs = calculate_ngram_probabilities(3, tokenized_sentences)
input_sentence = "वह बहुत खुश है।"

print("BIGRAM")
display_ngram_probabilities(2, input_sentence, bigram_probs)
print("TRIGRAM")
display_ngram_probabilities(3, input_sentence, trigram_probs)
```

Show hidden output

EXPT 5

- NN: Noun, singular or mass
- NNP: Proper noun, singular
- NST: Noun denoting spatial/temporal
- PRP: Personal pronoun
- VM: Verb, main
- VAUX: Verb, auxiliary
- JJ: Adjective
- RB: Adverb
- PSP: Postposition
- QC: Quantifier, cardinal
- QF: Quantifier, fractional
- RP: Particle
- WQ: Wh-word
- SYM: Symbol

```
from nltk.tag import UnigramTagger, BigramTagger, DefaultTagger
from nltk.corpus import indian
from nltk.tokenize import word_tokenize

nltk.download('indian')
nltk.download('punkt')

corpus = [
    "मैं स्कूल जा रहा हूँ।",
    "वह बहुत खुश है।",
    "बिल्ली घर के अंदर है।",
    "हम सब काम कर रहे हैं।",
    "तुम कहाँ जा रहे हो?",
]
```

Show hidden output

```
hindi_corpus = indian.tagged_sents('hindi.pos')
unigram_tagger = UnigramTagger(hindi_corpus, backoff=DefaultTagger('NN'))


def pos_tagging(sentences):
    for sentence in sentences:
        tokenized_sentence = word_tokenize(sentence)
        tagged_sentence = unigram_tagger.tag(tokenized_sentence)
        print(f"Sentence: {sentence}")
        print(f"POS Tags: {tagged_sentence}")
        print("\n")

pos_tagging(corpus)
```

Show hidden output

EXPT 6

To study chunking

```
import nltk
from nltk import word_tokenize
from nltk.chunk import RegexpParser
from nltk.corpus import indian
from nltk.tag import UnigramTagger

nltk.download('punkt')
nltk.download('indian')

sentences = [
    "मैं स्कूल जा रहा हूँ।",
    "वह बहुत खुश है।",
    "बिल्ली घर के अंदर है।",
    "हम सब काम कर रहे हैं।",
    "तुम कहाँ जा रहे हो?",
]

hindi_corpus = indian.tagged_sents('hindi.pos')
unigram_tagger = UnigramTagger(hindi_corpus, backoff=DefaultTagger('NN'))
```

Show hidden output

```
def pos_tag_and_chunk(sentences):
    grammar = r"""
    NP: {<DT>?<JJ>*<NN|NNS>}   # Noun phrase
    VP: {<VB.*><NP|PP|CLAUSE>+$}  # Verb phrase
    CLAUSE: {<NP><VP>}  # Clause
    """
    chunk_parser = RegexpParser(grammar)

    for sentence in sentences:
        tokens = word_tokenize(sentence)
        tagged = unigram_tagger.tag(tokens)
        tree = chunk_parser.parse(tagged)
        print(f"Sentence: {sentence}\nPOS Tags: {tagged}\nChunked: {tree}\n")

pos_tag_and_chunk(sentences)
```

Show hidden output

EXPT 7

To study NER (Named entity recognition)

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

Show hidden output

```
import spacy

nlp = spacy.load("en_core_web_sm")

text = """ England won the 2019 world cup. The 2019 world cup happened in England. Washington is the capital of the US. The first presider

doc = nlp(text)

for ent in doc.ents:
    print(f"Entity: {ent.text}, Label: {ent.label_}")
```

Show hidden output