1Q) Aim: Experiment is to learn about morphological features of a word by analyzing it

Theory:

To preprocess your text simply means to bring your text into a fom1 that is predictable and analyzable for your task. A task here is a combination of approach and domain.

Machine Learning needs data in the numeric form. We basically used encoding technique (BagOf.Vlord, Bi-gram, n-gram, TF-IDF, Word2Vec) to encode text into numeric vector. But before encoding we first need to clean the text data and this process to prepare (or clean) text data before encoding is called text preprocessing, this is the very first step to solve the NLP problems.

Tokenization: It is about splitting strings of text into smaller pieces, or "tokens". Paragraphs can be tokenized into sentences and sentences can be tokenized into words.

Filtration: Similarly, if we are doing simple word counts, or trying to visualize our text with a word cloud, stop words are some of the most frequently occurring words but don't really tell us anything. We're often better off tossing the stop words out of the text. By checking the Filter Stop words option in the Text Pre-processing tool, you can automatically filter these words out.

Script Validation:

The script must be validated properly.

Code:

```
import nltk
import numpy as np
nltk.download("punkt")
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_tokenize
text1="this is the first sentence.this is the second sentence.is this third"
print(text1)
sent_tokenize(text1)
len(sent_tokenize(text1))
text1="this is the first sentence. this is the second sentence. is this third?"
print(text1)
sent_tokenize(text1)
len(sent_tokenize(text1))
word_tokenize(text1)
len(word_tokenize(text1))
word_result=word_tokenize(text1)
print(word_result)
word_result[1:5]
word_result[:5]
from nltk.tokenize.punkt import PunktSentenceTokenizer
pst=PunktSentenceTokenizer()
pst.tokenize(text1)
len(pst.tokenize(text1))
```

Output:

```
this is the first sentence.this is the second sentence.is this third
this is the first sentence. this is the second sentence. is this third?
['this', 'is', 'the', 'first', 'sentence', '.', 'this', 'is', 'the', 'second',
'sentence', '.', 'is', 'this', 'third', '?']
```

2Q) Aim: write a nlp program for Un-tagging a sentence?

Untagging a sentence:

The provided Python code utilizes NLTK's untag() function to extract only the words from a tagged list. In this case, it removes the tags from the list [('Tutorials', 'NN'), ('Point', 'NN')], resulting in the output ['Tutorials', 'Point'].

Code:

```
import nltk
from nltk.tag import untag
untag([('Tutorials', 'NN'), ('Point', 'NN')])
```

Output:

```
['Tutorials', 'Point']
```

3Q) Aim: Building Chunker: experiment is to know the importance of selecting proper features for training a model and size of training corpus in learning how to do chunking.

Theory:

These tools can be very helpful for kids who struggle with writing. To use word prediction, your child needs to use a keyboard to write. This can be an onscreen keyboard on a smartphone or digital tablet. Or it can be a physical keyboard connected to a device or computer.

Those suggestions are shown on the screen, like at the top of an onscreen keyboard. The child clicks or taps on a suggested word, and it's inserted into the writing.

There are also advanced word prediction tools available. They include:

Tools that read word choices aloud with text-to-speech. This is important for kids with reading issues who can't read what the suggestions are.

Word prediction tools that make suggestions tailored to specific topics. For instance, the words used in a history paper will differ a lot from those in a science report. To make suggestions more accurate, kids can pick special dictionaries for what they're writing about.

Tools that display word suggestions in example sentences. This can help kids decide between words that are confusing, like to, too and two.

Code:

```
import nltk
from nltk.corpus import state_union
from nltk.tokenize import PunktSentenceTokenizer
train_text=state_union.raw("2005-GWBush.txt")
sample_text=state_union.raw("2006-GWBush.txt")
custom_sent_tokenizer=PunktSentenceTokenizer(train_text)
tokenized=custom_sent_tokenizer.tokenize(sample_text)
def process_content():
    try:
        for i in tokenized:
            words=nltk.word_tokenize(i)
            tagged=nltk.pos_tag(words)
            chunkGram=r"""Chunk:{<RB>?<VB>*<NNP>+<NN>?}"""
            chunkParser=nltk.regexpParser(chunkGram)
            chunked=chunkParser.parse(tagged)
            chunked.draw()
    except Exception as e:
        print(str(e))
process_content()
```

Output:

A picture of draw()

Code:

```
import spacy
nlp = spacy.load("en_core_web_md")
text = "The quick brown fox jumps over the lazy dog."
doc = nlp(text)
print("Noun chunks:")
for chunk in doc.noun_chunks:
    print(chunk.text)
```

Output:

```
Noun chunks:
The quick brown fox
the lazy dog
```

4Q) Aim: Write a python program to find factorial of a number using Recursion.

Code:

```
def recur_fact(n):
   if n == 1:
       return n
   else:
       return n*recur_fact(n-1)
num = int(input("Enter a number: "))
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   print("The factorial of",num,"is",recur_fact(num))
```

Output:

```
Enter a number: 5  The factorial of 5 is 120
```