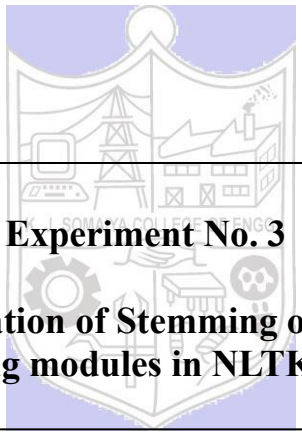

Experiment No. 3

Title: Implementation of Stemming of Text using different stemming modules in NLTK

Batch: B2 **Roll No.: 16010420061** **Experiment No.: 61**

Aim: To implement stemming of Text using different stemming modules in NLTK

Activity:

1. Apply Porter Stemmer, Snowball Stemmer, Lancaster Stemmer and Regexp Stemmer to a large corpus of Text.

```
import nltk
nltk.download('punkt')
from nltk.stem import PorterStemmer, SnowballStemmer, LancasterStemmer,
RegexpStemmer

# Sample corpus
corpus = [
"The quick brown foxes jumped over the lazy dogs.",
"He was playing with the cats in the garden.",
"The books on the shelf were dusty.",
"She loves reading interesting novels.",
"The company's products are top-notch.",
"They quickly ran to catch the last train.",
"I bought several delicious apples from the market.",
"The weather today is quite unpredictable.",
"I need to study for my upcoming exams.",
"The children were happily playing in the park.",
"In the vast forest, the birds chirped loudly.",
"The chef prepared a delicious meal for us.",
"My computer crashed while I was working.",
"The mountains are covered with snow in winter.",
"She enjoys listening to classical music.",
"The museum has an impressive collection of art.",
"The river flows peacefully through the valley.",
"The old house creaked in the night.",
"They celebrated their anniversary with a big party.",
"The detective solved the mysterious case.",
"The students eagerly awaited their exam results.",
"The sunsets on the beach are breathtaking.",
"She wore a beautiful dress to the party.",
"The city is bustling with activity.",
"The athletes trained hard for the competition.",
"The scientist made a groundbreaking discovery.",
```

```
    "I visited the historic castle on my vacation.",
    "The garden is full of colorful flowers.",
    "He gave an inspiring speech at the conference.",
    "The train station was crowded with travelers.",
    "The old tree provided shade in the scorching heat.",
    "The beach was crowded with sunbathers.",
    "The orchestra played a beautiful symphony.",
    "The students formed a study group for the exam.",
    "The car engine roared to life.",
    "The rain fell steadily throughout the night.",
    "The dog barked loudly at the mailman.",
    "The stars shone brightly in the clear night sky.",
    "The chef used fresh ingredients in the recipe.",
    "The cat purred contentedly on the windowsill.",
    "The construction workers were busy on the site.",
    "The flowers bloomed in vibrant colors.",
    "The politician delivered a passionate speech.",
    "The ocean waves crashed against the shore.",
    "The artist painted a stunning landscape.",
    "The laughter of children filled the playground.",
    "The clock ticked loudly in the quiet room.",
    "The scientist conducted experiments in the lab.",
    "The coffee shop was a cozy place to relax.",
    "The hikers enjoyed the beautiful scenery.",
    "The singer had a powerful and melodious voice.",
    "The bicycle race covered a long distance.",
    "The company announced a new product launch.",
    "The spider spun a delicate web.",
    "The fireworks lit up the night sky.",
    "The sunrises in the mountains were breathtaking.",
    "The astronauts explored the depths of space.",
    "The novelist wrote a gripping thriller.",
    "The ship sailed across the open sea.",
    "The teacher explained the complex topic.",
    "The thunderstorm rumbled in the distance.",
    "The river meandered through the countryside.",
    "The mechanic repaired the car's engine.",
    "The children built sandcastles on the beach.",
    "The basketball team won the championship.",
    "The dancers moved gracefully on the stage.",
    "The bakery sold fresh bread and pastries.",
    "The photographer captured stunning landscapes.",
    "The city skyline was illuminated at night.",
    "The archaeologists discovered ancient artifacts.",
    "The rainbows appeared after the rain shower.",
    "The lawyer argued the case in court.",
    "The birds chirped merrily in the morning.",
    "The nurse cared for the patients with kindness.",
```

```
  "The scientist made a breakthrough in research.",
  "The gardeners tended to the blooming flowers.",
  "The snowfall covered the town in white.",
  "The actor performed brilliantly on stage.",
  "The students organized a charity fundraiser.",
  "The waves crashed against the rocky shore.",
  "The chef prepared a gourmet meal.",
  "The athletes competed in the Olympic Games.",
  "The movie director filmed a thrilling scene.",
  "The passengers boarded the train.",
  "The astronauts conducted experiments in space.",
  "The author wrote a best-selling novel.",
  "The wind whispered through the trees.",
  "The children played games in the backyard.",
  "The mechanic fixed the broken car.",
  "The musician played a soulful melody.",
  "The scientists analyzed the data carefully.",
  "The beachcombers searched for seashells.",
  "The artist exhibited paintings in the gallery.",
  "The teacher inspired her students.",
  "The firefighters extinguished the blazing fire.",
  "The tourists explored the ancient ruins.",
  "The software engineer coded a new program.",
  "The marathon runners raced through the city.",
  "The family gathered for a holiday dinner.",
  "The astronaut floated in zero gravity."

  ]

  # Initialize stemmers
  porter = PorterStemmer()
  snowball = SnowballStemmer("english")
  lancaster = LancasterStemmer()
  # Define custom regular expression rules for Regexp Stemmer
  custom_pattern = r'ing$|ed$|es$|s$|ly$'
  regexp = RegexpStemmer(custom_pattern)

  # Apply and print stemming for each line in the corpus
  for line in corpus:
      words = nltk.word_tokenize(line.lower())  # Tokenize and convert to
lowercase
      porter_stemmed = [porter.stem(word) for word in words]
      snowball_stemmed = [snowball.stem(word) for word in words]
      lancaster_stemmed = [lancaster.stem(word) for word in words]
      regexp_stemmed = [regexp.stem(word) for word in words]

      print("Original: ", line)
      print("Porter: ", " ".join(porter_stemmed))
```

```
    print("Snowball: ", " ".join(snowball_stemmed))
    print("Lancaster: ", " ".join(lancaster_stemmed))
    print("Regexp: ", " ".join(regexp_stemmed))
    print()
```

2. Write down the observations about differences in the working of each of the Stemmer by analyzing the result obtained after applying each Stemmer on large corpus of text

Eg:
Original: The quick brown foxes jumped over the lazy dogs.

Porter Stemmer:

The Porter Stemmer tends to be relatively aggressive in stemming words, sometimes resulting in stems that are not actual words. For example, "products" becomes "product," and "interesting" becomes "interest."
It often removes common suffixes like "-es" and "-ed," but it may not handle irregular words effectively.
Eg: Porter: the quick brown fox jump over the lazi dog .

Snowball Stemmer:

The Snowball Stemmer (Porter2) provides more accurate stems compared to the original Porter Stemmer. It generally produces stems that are more likely to be real words.
It handles common suffixes and plural forms well, such as converting "products" to "product" and "interesting" to "interest."
Eg: Snowball: the quick brown fox jump over the lazi dog .

Lancaster Stemmer:

The Lancaster Stemmer is the most aggressive of the stemmers mentioned. It often produces very short stems that may not resemble the original word closely. For example, "products" becomes "produc," and "interesting" becomes "interest."
While Lancaster aggressively reduces words, it can lead to a loss of readability and sometimes produces stems that are not valid words.
Eg: Lancaster: the quick brown fox jump ov the lazy dog .

Regexp Stemmer:

The Regexp Stemmer allows you to define custom rules using regular expressions. In this example, we used a simple pattern to remove common suffixes like "-ing," "-ed," "-es," "-s," and "-ly."
The advantage of the Regexp Stemmer is its flexibility; you can define rules specific to your needs. However, it requires careful crafting of regular expressions for each stemming rule.
Eg: Regexp: the quick brown fox jump over the lazy dog .

Questions:

1. Explain Under-Stemming and Over-Stemming errors with the help of suitable example.

Under-stemming and over-stemming are two common types of errors that can occur when applying stemming algorithms to natural language text. These errors are related to the balance between preserving the original word's meaning and simplifying words to their root or base form.

Under-Stemming (Under-Trimming):

Under-stemming occurs when a stemming algorithm is too conservative and doesn't remove enough of a word's affixes or suffixes. As a result, the stemmed word remains too close to the original word, and variations of the same word are treated as different terms.
Example of under-stemming:

Original words: "jumped," "jumping," "jumps," "jumper"

Stemmed words (under-stemmed): "jumped," "jumping," "jumps," "jumper"

In this case, under-stemming preserves too much of the original words, making it harder to identify that all of these words are forms of the same verb "jump."

Over-Stemming (Over-Trimming):

Over-stemming occurs when a stemming algorithm is too aggressive and removes too many affixes or suffixes, resulting in a stemmed word that is too far removed from the original word. This can lead to the loss of the word's actual meaning and make it difficult to understand the context.
Example of over-stemming:

Original words: "unhappiness," "unhappily," "happiness," "happy"

Stemmed words (over-stemmed): "unhappi," "unhappili," "happi," "happi"

In this case, over-stemming reduces all the words to "happi," which not only loses the negation in "unhappiness" but also collapses the adverb "unhappily" into the same stem, making it challenging to distinguish between different word forms.

Outcomes: CO3- Establish concept of Structure and Semantics

Conclusion: (Conclusion to be based on the outcomes achieved)
Successfully implemented stemming in large corpus of data and noted the observations.

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

Books/ Journals/ Websites:

1. Allen.James, Natural Language Understanding, Benjamin Cumming, Second Edition, 1995
2. Jurafsky, Dan and Martin, James, Speech and Language Processing, Prentice Hall, 2008
3. Palash Goyal, Karan Jain, Sumit Pandey,Deep Learning for Natural Language Processing: Creating Neural Networks with Python, Apress, 2018