STEMMING

1) Coding is like solving puzzles with the computer

2) Natural language processing involves understanding human languages

3) Machine learning algorithms improve over time

4) The quick brown fox jumps over the lazy dog

5) Sushi is my favourite food , but i also love pizza

6) Penguins waddle gracefullly on icy terrain

7) Rainbows and unicorns brighten up even the cloudiest days

```
pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.1)
```

```
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

# Download NLTK data (if not already downloaded)
nltk.download('punkt')

# Create a stemmer object
stemmer = PorterStemmer()

# Number of times to perform stemming
num_inputs = 7

for i in range(num_inputs):
    # Take input sentence from the user
    sentence = input(f"Enter sentence {i+1}: ")

    # Tokenize the sentence into words
    words = word_tokenize(sentence)

    # Stem each word and print the result
    stemmed_words = [stemmer.stem(word) for word in words]
    stemmed_sentence = ' '.join(stemmed_words)

    print(f"Stemmed sentence {i+1}: {stemmed_sentence}\n")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Enter sentence 1: Coding is like solving puzzles with the computer
Stemmed sentence 1: code is like solv puzzl with the comput

Enter sentence 2: Natural language processing involves understanding human languages
Stemmed sentence 2: natur languag process involv understand human languag

Enter sentence 3: Machine learning algorithms improve over time
Stemmed sentence 3: machin learn algorithm improv over time

Enter sentence 4: The quick brown fox jumps over the lazy dog
Stemmed sentence 4: the quick brown fox jump over the lazi dog

Enter sentence 5: Sushi is my favourite food , but i also love pizza
Stemmed sentence 5: sushi is my favourit food , but i also love pizza

Enter sentence 6: Penguins waddle gracefullly on icy terrain
Stemmed sentence 6: penguin waddl gracefullli on ici terrain

Enter sentence 7: Rainbows and unicorns brighten up even the cloudiest days
Stemmed sentence 7: rainbow and unicorn brighten up even the cloudiest day
```

LEMMATIZATION

1) running

2) better

3) cats

4) went

5) happily

6) geese

```
#importing necessary libraries
from nltk.stem import WordNetLemmatizer
nltk.download("wordnet")
nltk.download("omw-1.4")
nltk.download('punkt')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
# Initialize wordnet lemmatizer
wnl = WordNetLemmatizer()
```

```
# Example inflections to reduce
example_words = ["running", "better", "cats", "went", "happily", "geese"]
```

PARTS OF SPEECH - VERB

```
# Perform lemmatization
print("{0:20}{1:20}".format("--Word--","--Lemma--"))
for word in example_words:
   print ("{0:20}{1:20}".format(word, wnl.lemmatize(word, pos="v")))
```

```
--Word--            --Lemma--
running             run
better              better
cats                cat
went                go
happily             happily
geese               geese
```

PARTS OF SPEECH - NOUN

```
# Perform lemmatization
print("{0:20}{1:20}".format("--Word--","--Lemma--"))
for word in example_words:
   print ("{0:20}{1:20}".format(word, wnl.lemmatize(word, pos="n")))
```

```
--Word--            --Lemma--
running             running
better              better
cats                cat
went                went
happily             happily
geese               goose
```

PARTS OF SPEECH - ADJECTIVE

```
# Perform lemmatization
print("{0:20}{1:20}".format("--Word--","--Lemma--"))
for word in example_words:
   print ("{0:20}{1:20}".format(word, wnl.lemmatize(word, pos="a")))
```

```
--Word--            --Lemma--
running             running
```

```
     better              good
     cats                cats
     went                went
     happily             happily
     geese               geese
```

CONCLUSION :

1) Happily is the only word which was not affected by any POS.

2) When we used the POS as 'a',the lemmatized word for better was good.

3) Running changes to run when pos = v

4) the word geese changes to goose when parts of speech = noun

LEMMATIZATON VS STEMMING :

~ Lemmatization is the process of reducing words to their base or dictionary form (lemma). The result is always a valid word.

Produces meaningful words. Maintains grammatical accuracy. Useful for tasks like text analysis, sentiment analysis, and language generation.

vs

~ Stemming is the process of reducing words to their base or root form by removing suffixes or prefixes. The result may not always be a valid word.

Simple and computationally efficient. Can be useful for tasks like information retrieval and search engines.

```
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk import pos_tag
from nltk.tokenize import word_tokenize

# Download necessary resources (you only need to do this once)
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')

# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()

# Example sentences
sentences = [
    "I am running in the park",
    "The dogs are barking loudly",
    "She was painting a beautiful picture"
]

for sentence in sentences:
    words = word_tokenize(sentence)
    pos_tags = pos_tag(words)

    lemmatized_words = []
    for word, pos in pos_tags:
        pos_letter = pos[0].lower()
        if pos_letter in ['a', 'v', 'n', 'r']:
            # Adjective, Verb, Noun, Adverb
            lemma = lemmatizer.lemmatize(word, pos_letter)
        else:
            # Default to noun lemmatization
            lemma = lemmatizer.lemmatize(word, 'n')
        lemmatized_words.append((lemma, pos))

    print("Original:", sentence)
    print("Lemmatized:")
    for lemma, pos in lemmatized_words:
        print(f"{lemma} ({pos})", end=" ")
    print("\n")
```