**NLP - LAB 5 Date : 16 - 02- 24**

# AIM :

1. Write a Python programs to find synonyms and antonyms of a word with WordNet and NLTK corpus.
2. Write a Python programs to POS Tagging for synonyms and antonyms with NLTK WordNet.

**RESOURCES :** Google Colab / Jupyter notebook.

# OBJECTIVES :

# The objective is to develop a Python program using WordNet and NLTK to find synonyms and antonyms of a specified word. By integrating WordNet's lexical database, the program extracts related words. Subsequently, utilizing NLTK for POS tagging, it assigns grammatical tags to both synonyms and antonyms . And the output tells about the synonyms, antonyms, and their corresponding POS tags in a readable format.

# PROCEDURE:

1) Install NLTK and download WordNet data.

2) Import necessary libraries: nltk, nltk.corpus, and nltk.tokenize.

3) Define a function to find synonyms and antonyms using WordNet.

4) Implement a function to perform POS tagging for synonyms and antonyms using NLTK's pos\_tag.

5) Represent the output by displaying synonyms and antonyms along with their POS tags.

# Code :

import nltk

nltk.download('wordnet')

from nltk.corpus import wordnet as wordnet

[nltk\_data] Downloading package wordnet to /root/nltk\_data...

### **Checking if word is in Wordnet**

try:

    word='putin'

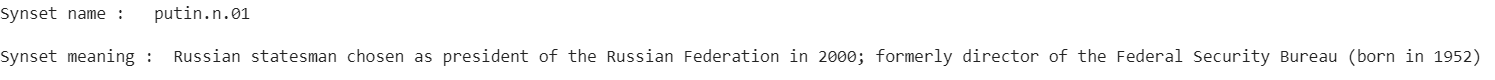
    syn = wordnet.synsets(word)[0]

    print ("Synset name :  ", syn.name())

    print ("\nSynset meaning : ", syn.definition())

except IndexError:

    print("Word not in wordnet")



### In this part of code we are trying to find the synsets (a set of synonyms) and meanings of a word using WordNet.A try-catch block is applied which is used for error handling. It tries to execute the code within the try block, and if an error occurs, it executes the code within the except block.

### A word = 'putin' is assigned ,and then it retrieves the synsets for the word 'putin' using WordNet.And prints the name of the synset,and it's meaning.

### If the word 'putin' exists in WordNet, it prints the name of the first synset and its definition.If the word 'putin' is not found in WordNet, it prints "Word not in wordnet".

### **Extracting synonyms and antonyms of the selected word**

synonyms = []

antonyms = []

for syn in wordnet.synsets(word):

    for l in syn.lemmas():

        synonyms.append(l.name())

        if l.antonyms():

            antonyms.append(l.antonyms()[0].name())

print('Synonyms:',set(synonyms)) if len(synonyms)>0 else print("No synonyms")

print('Antonyms:',set(antonyms)) if len(antonyms)>0 else print("No Antonyms")



### First we initialize two empty lists, synonyms and antonyms, to store the synonyms and antonyms of the word.

### It iterates over each synset of the given word obtained from WordNet using a nested loop.

### For each synset, it iterates over each lemma using another loop.

### It appends the names of all lemmas (synonyms) to the synonyms list.

### If an antonym exists for a lemma, it appends the name of the antonym to the antonyms list.

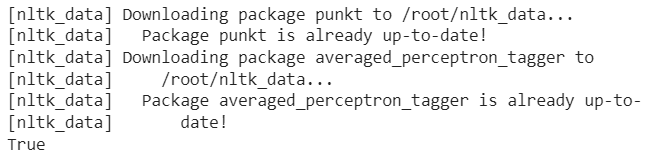
### It prints the set of synonyms and antonyms for the given word if they exist in WordNet.If there are no synonyms or antonyms found, it prints No synonyms /antonyms.

### **POS Tagging of synonyms and antonyms**

nltk.download('punkt')

from nltk.tokenize import word\_tokenize

nltk.download('averaged\_perceptron\_tagger')



In this code it downloads the Punkt tokenizer models. It is a pre-trained unsupervised machine learning model for tokenizing words and sentences. And then we import the word\_tokenize function is used for tokenizing a text into words. It's a convenient way to break a text into individual words, handling things like punctuation and special characters properly. And further we download the pre-trained model for part-of-speech (POS) tagging called the Averaged Perceptron Tagger.

# Converting list to string with space

def listToString(s):

    # initialize an empty string

    str1 = " "

    # return string

    return (str1.join(s))

#Synonym POS tagging

syn=listToString(synonyms)

print(syn)

tokens = nltk.word\_tokenize(syn)

pos\_tags = nltk.pos\_tag(tokens)

print("POS tagging for synonyms:")

print(pos\_tags)

#Antonym POS tagging

ant=listToString(antonyms)

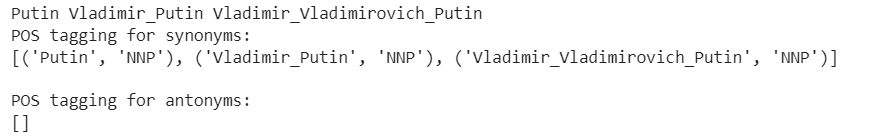
print(ant)

tokens = nltk.word\_tokenize(ant)

pos\_tags = nltk.pos\_tag(tokens)

print("POS tagging for antonyms:")

print(pos\_tags)



In this code listToString function is used which is a function which takes a list of strings (s) as input and joins its elements into a single string separated by spaces. It initializes an empty string (str1) with a space as a separator. Then, it uses the join() method to concatenate all elements of the list with spaces in between and returns the resulting string.

Then POS tagging is performed and it converts the list of synonyms to a string using the listToString function. It tokenizes the resulting string into words using nltk.word\_tokenize. Then, it performs (POS) tagging on the tokens. Finally, it prints the POS tagging results for the synonyms.

Same we perform for Antonyms also.

So simply this code demonstrates how to convert lists of synonyms and antonyms into strings, tokenize the strings into words, and then perform POS tagging on the words using NLTK.

----------------------------------------------------------X X X --------------------------------------------------------------