

**DEPARTMENT OF COMPUTER ENGINEERING**

## CSL804 Computational Lab II

**Eighth Semester, 2021-2022 (Even Semester)**

**Name of Student :** Saurav Kumar

## Roll No. 23

**Division :** BE – CMPN

**Day/ Session :** Monday/Afternoon

**Venue :** SLRTCE Lab 305

## Experiment No. 10

**Title of Experiment :** To study and implement the concept of Senti-Wordnet.

## Date of Conduction :

**Date of Submission :**

|  |  |  |
| --- | --- | --- |
| **Particulars Max. Marks Marks Obtained** | | |
| Preparedness and Efforts(PE) | **3** |  |
| Knowledge of tools(KT) | **3** |  |
| Debugging and results(DR) | **3** |  |
| Documentation(DN) | **3** |  |
| Punctuality & Lab Ethics(PL) | **3** |  |
| **Total** | **15** |  |

**Grades – Meet Expectations (3 Marks), Moderate Expectations (2 Marks), Below Expectations (1 Mark)**

**Checked and Verified by Name of Faculty :** Prof. Neelam Kulkarni

## Signature :

**Date :**

EXPERIMENT NO: 10

SENTI-WORDNET

**AIM:** To study and implement the concept of Senti-Wordnet.

**SOFTWARE:** Python, NLTK.

# THEORY:

SENTIMENT ANALYSIS

Sentiment Analysis is the computational study of opinions, sentiments and emotions expressed in text. Earlier, most text information processing methods (e.g., web search, text mining) worked with factual information. Senti-WordNet is built via a semi supervised method and could be a valuable resource for performing opinion mining tasks it provides a readily available database of term sentiment information for the English language, and could be used as a replacement to the process of manually deriving ad- hoc opinion lexicons.

# IMPLEMENTATION:

SENTI-WORDNET CODE

import nltk #nltk.download('averaged\_perceptron\_tagger') # #nltk.download('sentiwordnet')

from nltk.corpus import wordnet as wn

from nltk.corpus import sentiwordnet as swn from nltk.stem import PorterStemmer

def penn\_to\_wn(tag):

# Convert between the PennTreebank tags to simple Wordnet tags if tag.startswith('J'):

return wn.ADJ

elif tag.startswith('N'): return wn.NOUN

elif tag.startswith('R'): return wn.ADV

elif tag.startswith('V'): return wn.VERB

return None

from nltk.stem import WordNetLemmatizer lemmatizer = WordNetLemmatizer()

def get\_sentiment(word,tag):

# returns list of pos neg and objective score. But returns empty list if not present in senti wordnet.

wn\_tag = penn\_to\_wn(tag)

if wn\_tag not in (wn.NOUN, wn.ADJ, wn.ADV): return []

lemma = lemmatizer.lemmatize(word, pos=wn\_tag) if not lemma:

return []

synsets = wn.synsets(word, pos=wn\_tag) if not synsets:

return []

# Take the first sense, the most common synset = synsets[0]

swn\_synset = swn.senti\_synset(synset.name()) return

[swn\_synset.pos\_score(),swn\_synset.neg\_score(),swn\_synset.obj\_score()

]

ps = PorterStemmer()

words\_data = ['this','movie','is','wonderful']

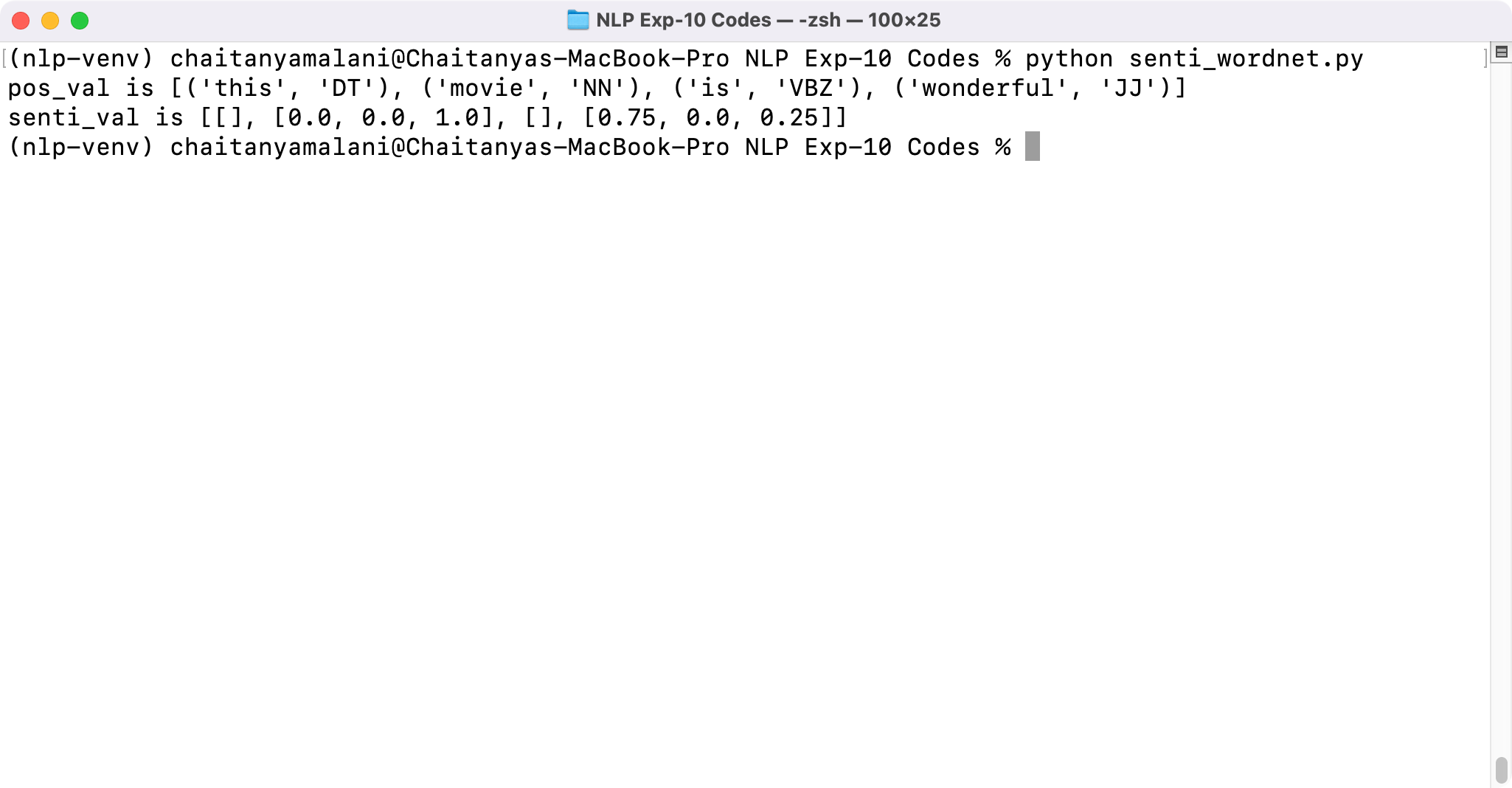
# words\_data = [ps.stem(x) for x in words\_data] # if you want to further stem the word

pos\_val = nltk.pos\_tag(words\_data)

senti\_val = [get\_sentiment(x,y) for (x,y) in pos\_val] print(f"pos\_val is {pos\_val}")

print(f"senti\_val is {senti\_val}")

OUTPUT



# CONCLUSION:

Thus we have studied and implemented concept of Senti-Wordnet.