****

**PROGRAMMING FOR AI (LAB)**

**Lab Task No 09**

**Submitted To :- Sir Rasikh Ali**

**Submitted By :- Aiman Ijaz**

**Roll No :- SU92-BSAIM-F23-010**

**Department :- Software Engineering**

**Section :- BSAI-4A**

**Can do and submit any task from the given subtasks within the Section: Natural Language Processing Tasks**

**Url:** <https://www.geeksforgeeks.org/natural-language-processing-nlp-tutorial/>

This task performs **sentiment analysis**—the process of identifying and categorizing emotions expressed in text. It uses two approaches:

1. **Machine Learning Approach (RNN)**: Learns patterns from data to classify reviews as positive or negative.
2. **Rule-Based Approach (VADER)**: Uses predefined rules and sentiment lexicons to assign scores to text.

**Part 1: Sentiment Analysis Using RNN (Recurrent Neural Network)**

**What Is It Doing?**

This part of the project uses a **Recurrent Neural Network (RNN)** to understand and classify customer reviews from a food delivery dataset (Swiggy). It decides whether a review is **positive** or **negative** based on the average rating.

**How the Code Works**

**1. Importing Libraries**

The code starts by importing libraries for:

* Data handling (pandas, numpy)
* Text preprocessing
* Machine learning model creation (Keras with TensorFlow backend)

**2. Loading the Dataset**

It reads a CSV file (swiggy.csv) that contains customer reviews and their average ratings.

**3. Preprocessing the Reviews**

Before feeding the reviews into a machine learning model, the text needs to be cleaned:

* All text is converted to lowercase for uniformity.
* Special characters, punctuation, and symbols are removed.
* Each review is assigned a label:
  + **Positive (1)** if the rating is above 3.5
  + **Negative (0)** otherwise

This converts subjective review text into a format that a machine learning model can understand and learn from.

**4. Tokenization and Padding**

Words are converted into numbers using a tokenizer. Since machines work with numbers, each word is assigned an index based on how frequently it appears.

Because different reviews have different lengths, padding is added to make all sequences the same size, ensuring consistent input shape for the model.

**5. Splitting the Data**

The dataset is divided into a training set and a testing set. The training set is used to teach the model, and the testing set checks how well the model learned.

**6. Building the RNN Model**

A Recurrent Neural Network is created using Keras. RNNs are especially good at handling sequences like text because they remember previous inputs, helping to capture the meaning and context in a sentence.

The model includes:

* **Embedding layer**: Converts each word index into a dense vector, capturing relationships between words.
* **RNN layer**: Reads the sequence of vectors and learns patterns.
* **Dense output layer**: Predicts 1 (positive) or 0 (negative).

**7. Training the Model**

The model is trained for a few rounds (called epochs) using the training data. Each round helps the model improve by reducing its prediction errors.

**8. Evaluating Accuracy**

After training, the model is tested on unseen data to measure how well it can classify new reviews as positive or negative.

**Why This Approach?**

* RNNs are good for processing sequential data like text.
* Training the model on actual reviews helps it learn real-world patterns and context.
* Once trained, the model can classify any new review with high accuracy.

**Part 2: Sentiment Analysis Using VADER**

**What Is It Doing?**

This part uses the **VADER** tool (Valence Aware Dictionary and sEntiment Reasoner) to analyze the emotional tone of individual sentences.

**How the Code Works (Step-by-Step)**

**1. Using VADER’s Sentiment Analyzer**

VADER is a built-in tool that looks at each word in a sentence and assigns it a sentiment score based on a predefined dictionary.

**2. Calculating Sentiment Scores**

For any input sentence, VADER returns four scores:

* **Positive**: how much of the text is positive
* **Negative**: how much is negative
* **Neutral**: how much is neutral
* **Compound**: a combined score summarizing the overall sentiment (ranges from -1 to 1)

**3. Interpreting the Results**

The compound score is used to categorize the overall sentiment:

* Score ≥ 0.05: **Positive**
* Score ≤ -0.05: **Negative**
* Otherwise: **Neutral**

This gives a quick and simple way to determine the emotional tone of a sentence.

**4. Example Outputs**

The code tests three example sentences:

* A highly positive statement about a website
* A neutral sentence about studying
* A negative sentence expressing sadness

VADER correctly classifies each based on the emotional content.

**Why This Approach?**

* VADER is fast and requires no training.
* It is specially tuned for social media and short text.
* Great for real-time applications where speed matters.
* Doesn't require labeled training data.

**Final Thoughts**

Both methods are effective, and choosing one depends on the project goals:

* Use **RNN** for building custom sentiment models.
* Use **VADER** for quick rule-based analysis.