|  |  |  |
| --- | --- | --- |
| **Name : Harsh Jain** | **Class/Roll No. : D16AD / 25** | **Grade :** |

**Title of Experiment :**

Word Cloud Generation in Natural Language Processing (NLP)

**Problem Statement :**

The problem is to generate a word cloud from a given text or set of texts, representing the frequency of words visually. The objective is to create an appealing and informative visualization that showcases the most prominent words in the input text.

**Description / Theory :**

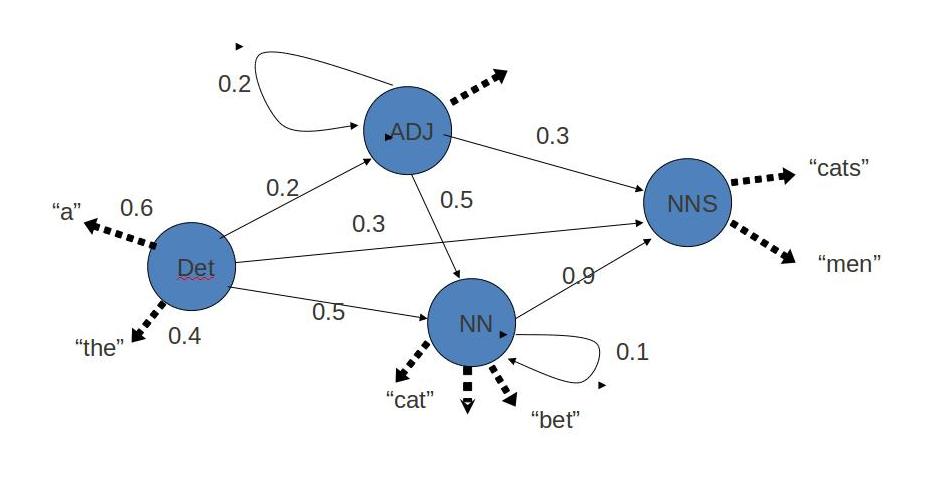
A word cloud is a visual representation of text data, where words are displayed in varying sizes and colors based on their frequency or importance. Commonly used words are usually displayed in a larger font to emphasize their occurrence. The creation of word clouds involves text preprocessing, word frequency calculation, and graphical representation to generate an informative and aesthetically pleasing visualization.

**Hidden Markov Model :**

POS tagging or part-of-speech tagging is the procedure of assigning a grammatical category like noun, verb, adjective etc. to a word. In this process both the lexical information and the context play an important role as the same lexical form can behave differently in a different context.

For example the word "Park" can have two different lexical categories based on the context.

* The boy is playing in the park. ('Park' is Noun)
* Park the car. ('Park' is Verb)

****

Assigning part of speech to words by hand is a common exercise one can find in an elementary grammar class. But here we wish to build an automated tool which can assign the appropriate part-of-speech tag to the words of a given sentence. One can think of creating hand crafted rules by observing patterns in the language, but this would limit the system's performance to the quality and number of patterns identified by the rule crafter. Thus, this approach is not practically adopted for building POS Tagger. Instead, a large corpus annotated with correct POS tags for each word is given to the computer and algorithms then learn the patterns automatically from the data and store them in form of a trained model. Later this model can be used to POS tag new sentences.

In this experiment we will explore how such a model can be learned from the data.

STEP1: Select the corpus.

STEP2: For the given corpus fill the emission and transition matrix. Answers are rounded to 2 decimal digits.

STEP3: Press Check to check your answer.

**Viterbi Model :**

Viterbi Decoding is based on dynamic programming. This algorithm takes emission and transmission matrix as the input. Emission matrix gives us information about proabities of a POS tag for a given word and transmission matrix gives the probability of transition from one POS tag to another POS tag. It observes sequence of words and returns the state sequences of POS tags along with its probability.

STEP1:Select the corpus.

OUTPUT: Emission and Transmission matrix will appear.

STEP2: Fill the column with the probabilty of possible POS tags given the word (i.e. form the viterbi matrix by filling colum for each observation). Answers submitted are rounded off to 3 digits after decimal and are than checked.

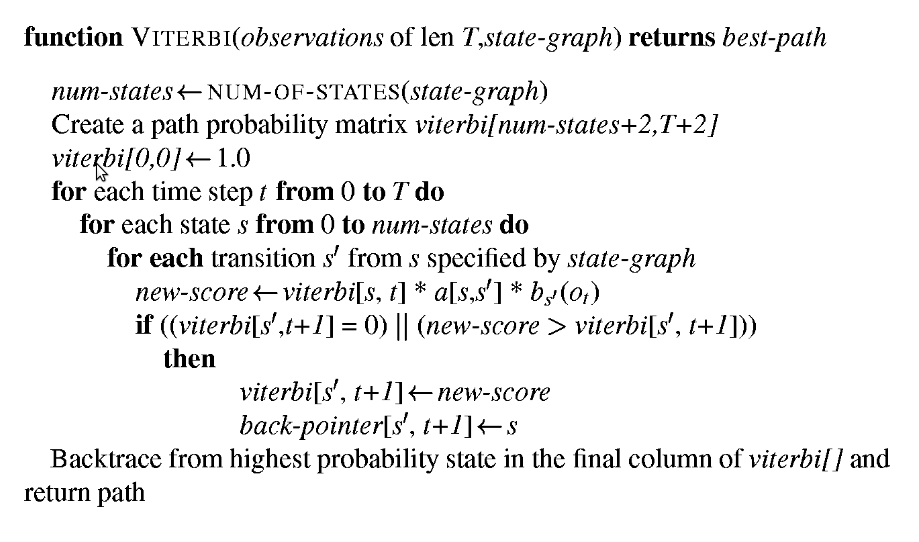
STEP3: Check the column.

Wrong answers are indicated by red backgound in a cell.

If answers are right, then go to step2

STEP4: Repeat steps 2 and 3 untill all words of a sentence are covered.

STEP5: At last check the POS tag for each word obtained from backtracking



**Flowchart** :

1. Input the text data.
2. Preprocess the text (tokenization, lowercasing, stopword removal, etc.).
3. Calculate the frequency of each word.
4. Determine the font size and color for each word based on its frequency.
5. Generate the word cloud using the layout algorithm.
6. Display the word cloud.

|  |
| --- |
| **Program:**  import matplotlib.pyplot as plt  from wordcloud import WordCloud  # Sample text for word cloud  sample\_text = "Word clouds are a great way to visualize text data and display the most frequent words."  # Generate the word cloud  wordcloud = WordCloud(width=800, height=400, background\_color='white').generate(sample\_text)  # Display the word cloud using matplotlib  plt.figure(figsize=(10, 5))  plt.imshow(wordcloud, interpolation='bilinear')  plt.axis("off")  plt.show()  **Output:** |

**Results and Discussions :**

Text preprocessing plays a pivotal role in Natural Language Processing (NLP) by refining raw textual Results and Discussions: The results will display the generated word cloud, showcasing the words with varying font sizes and colors based on their frequency. The discussion will focus on the visual representation, the prominence of words, and how it aids in understanding the most significant terms in the given text.

**Conclusion:**

Word clouds are a useful visualization tool in NLP for presenting the most frequent words in a visually appealing manner. They offer a quick understanding of the key terms in a text, aiding in analysis and interpretation. The frequency-based representation in a word cloud allows for a simplified yet effective visualization of textual data.

**\*\*\*\*\*\***