NEXT WORD PREDICTION USING DEEP LEARNING

PROJECT TEAM-15 CSE-ARTIFICIAL INTELLIGENCE&DATA SCIENCE:

1) MANI SAI CHARAN (TL)

2) VINEETH

3)BHANU SIDDHARTHA

4) SUNIL VIKAS

5) DURGA PAVAN

ABSTRACT

Next word prediction is a crucial task in Natural Language Processing (NLP), enabling intelligent systems to predict the next word in a sequence of text based on context. This task finds wide applications in areas such as text autocompletion, smart keyboards, chatbots, voice assistants, and search engines. The rise of deep learning techniques has significantly advanced the accuracy and capabilities of next word prediction models by enabling systems to learn complex relationships within textual data. This project focuses on the development and application of deep learning techniques for the next word prediction task, specifically utilizing Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformer models. The purpose of this abstract is to provide an overview of the methodologies, challenges, and achievements of applying deep learning to next word prediction.

Next word prediction involves determining the most probable word to follow a given sequence of words. For instance, in the sentence "The cat sat on the," the model would predict that the next word might be "mat" or "floor." Traditionally, this task was approached using statistical models such as n-grams, which utilized the frequency of word occurrences to predict the next word. However, these models struggled to account for long-range dependencies and complex semantic relationships in language.

Deep learning models, particularly RNNs, and their variants such as LSTMs and GRUs, have brought significant improvements in handling sequential data, offering a more sophisticated approach for next word prediction by capturing long-term dependencies in textual data.

INPUT:

In the context of a next-word prediction model, the **input** is typically a sequence of words or tokens that represent a partially written sentence or phrase. The goal is to predict the next word that follows this sequence.

- **Text Preprocessing**: Before feeding text into the model, several preprocessing steps are applied to make the data suitable for the deep learning model:
    - **Tokenization**: Breaking the input text into smaller units like words or subwords (tokens).
    - **Vectorization**: Converting these tokens into numerical representations (e.g., using word embeddings like Word2Vec, GloVe, or contextual embeddings like BERT).
    - **Padding/Truncation**: Since models typically require inputs of fixed length, the input sequence is padded with zeros if it is shorter than the desired length, or truncated if it's too long.

- **Contextual Information**: If you are using a model like LSTM, GRU, or Transformer, the sequence might contain a fixed-length history (e.g., the last N words) to provide context for prediction.

PROCESS

The **processing stage** involves feeding the input data (the sequence of words) into the deep learning model and using its architecture to learn and generate predictions.

- **Model Architecture**: Several deep learning architectures can be used for next-word prediction, including:
    - **Recurrent Neural Networks (RNNs)**: RNNs are designed to handle sequential data. They maintain a hidden state that captures information about previous words in the sequence, which is passed on as the model processes each new word. This helps the network retain context across time steps.
    - **Long Short-Term Memory (LSTM)**: LSTM is a type of RNN that is specifically designed to capture long-term dependencies. Unlike vanilla RNNs, LSTMs have mechanisms (gates) that help them retain relevant information over many timesteps.
    - **Gated Recurrent Units (GRU)**: A simplified variant of LSTM, GRUs are also used to handle sequential data and maintain context while being more computationally efficient.
    - **Transformer Models**: Transformers, such as GPT (Generative Pre-trained Transformers), are particularly powerful for next-word prediction. They use attention mechanisms to capture relationships between words across long distances, allowing them to understand context more effectively than RNN-based models.

I choose RNN model for next word prediction because, Recurrent Neural Networks (RNNs) are well-suited for next-word prediction in deep learning due to their ability to process sequential data. They work by maintaining an internal state that updates with each new word, allowing the model to retain context from previous words in a sentence. This is crucial for predicting the next word in a sequence, as the model learns patterns and relationships in the data. RNNs typically use word embeddings, which represent words as vectors, capturing semantic relationships between them. During training, the model learns from large text corpora, minimizing the difference between predicted and actual words using a loss function like categorical cross-entropy. Advanced RNN variants like LSTMs (Long Short-Term Memory) and GRUs (Gated Recurrent Units) address limitations of basic RNNs, such as the vanishing gradient problem, and are more effective at capturing long-term dependencies in text. RNNs are commonly used in real-time applications like autocomplete, chatbots, and text

generation, where they predict the next word based on dynamic input. Overall, RNNs, particularly with LSTM/GRU variants, are powerful tools for next-word prediction, capturing both short- and long-term context in natural language processing tasks.

OUTPUT:

The **output** of a next-word prediction model is the predicted next word or token that is most likely to follow the given input sequence.

- **Probability Distribution**:
    - The model outputs a probability distribution across the entire vocabulary for the next word. The word with the highest probability is often selected as the predicted word.
- **Predicted Word**: The final prediction is usually the word with the highest probability in the output distribution. For example, if the input is "The cat is on the", the model might predict "mat" as the next word.
- **Softmax Output Layer**:
    - The model typically has a softmax layer in the final output, which converts raw logits (unnormalized predictions) into probabilities for each word in the vocabulary.
    - The softmax function ensures that all the probabilities sum to 1, so you can interpret them as the likelihood of each word being the next word in the sequence.

Example Output:

If the input is:

- **Input Sequence**: "The sun is shining"
- **Output (Prediction)**: "brightly"

The model predicts that the most likely next word in the sequence is “brightly,” based on its training on a large corpus of text.