****

**International Institute of Information Technology Hyderabad**

**Report on Natural Language Processing**

**By: Muskan Rahangdale [ugmr20230037]**

**(Code Crew)**

***What is Natural Language Processing (NLP)?***

Natural Language Processing (NLP) is a subset of Artificial Intelligence (AI) that focuses on the interaction between computers and human language. It enables computers to understand, interpret, and respond to human language effectively. NLP is used in a wide range of applications, including:

1. **Text Classification:** Sorting and categorizing text data, such as spam detection in emails.
2. **Sentiment Analysis:** Identifying emotions or opinions expressed in text, like classifying reviews as positive or negative.
3. **Machine Translation:** Translating text from one language to another, such as Google Translate.
4. **Chatbots and Virtual Assistants**: Powering conversational interfaces like Siri or Alexa.
5. **Named Entity Recognition (NER):**Identifying entities like names, dates, or organizations in text.
6. **Text Summarization**: Automatically generating concise summaries of longer documents.

**Colab Notebook Link:**

<https://colab.research.google.com/drive/1SDwfRFGtKRCv-xK8jprBofBsmAnP7vYT?usp=sharing>

**Exaplaination for the code:**

**Preprocessing Steps for Sentiment Analysis:** The input text needs to be preprocessed to be compatible with the DistilBERT model. The key preprocessing steps are:

* ***Tokenization****:* The text is split into smaller units (tokens) that the model can process. For example, the sentence "I love Python!" might be tokenized into ["I", "love", "Python", "!"]. These tokens are then converted to numerical IDs corresponding to the model’s vocabulary*.*
* ***Padding and Truncation****:* To ensure uniform input sizes, shorter texts are padded with special tokens, and longer ones are truncated to a fixed length. This step aligns all inputs with the model’s expected dimensions.
* ***Conversion to Tensors****:* The processed tokens are converted into PyTorch tensors, which are the basic data structures used by the model. These tensors are efficient for computation and are required for inference.

**Model Type:** The code uses a pre-trained DistilBERT model fine-tuned for sentiment analysis. DistilBERT is a lightweight version of BERT (Bidirectional Encoder Representations from Transformers), designed for faster computation while retaining high accuracy. It predicts the sentiment of a text as either **positive** or **negative**.

**Model Workflow:**

* **Input**: The tokenized text is passed to the model for analysis.
* **Inference**: The model processes the input and generates logits, which are unscaled probabilities for each class (positive or negative sentiment).
* **Prediction**: The class with the highest logit value is identified, indicating the predicted sentiment.

**Example Usage:**

* **Positive Sentiment**: For a sentence like *"I love programming in Python,"* the model outputs positive sentiment, as the text reflects enthusiasm and positivity.
* **Negative Sentiment**: For a sentence like *"I am very disappointed with the service,"* the model outputs negative sentiment, as the text expresses dissatisfaction.

**Inference Process:** The process involves tokenizing the input text, passing it through the model, and interpreting the output. The model uses advanced language representations to understand the context and semantics of the text, making its predictions highly accurate.

**Advantages of Using Pre-trained Models:**

1. **Time Efficiency**: No need to train a model from scratch; it leverages pre-trained weights.
2. **Accuracy**: Fine-tuned models like DistilBERT achieve state-of-the-art performance on NLP tasks.
3. **Ease of Use**: Pre-trained models and tokenizers from libraries like Hugging Face are easy to integrate and deploy.

**Conclusion:** In this project, we used PyTorch and the Hugging Face Transformers library to perform sentiment analysis with the DistilBERT model. Here are the key points:

* **Model Selection:** We used the pre-trained DistilBERT model fine-tuned for sentiment analysis, which outputs logits that are mapped to sentiment classes (positive or negative).
* **Text Processing:** The input text was tokenized into IDs using Hugging Face's tokenizer, and the model made predictions by analyzing the text’s sentiment.
* **Results:** The model successfully classified text into positive and negative sentiments, demonstrating the power of pre-trained models for quick sentiment analysis.

This project highlighted how pre-trained models can simplify sentiment analysis tasks, offering an efficient solution for processing and understanding text data.

**Github Link:**

https://github.com/132refrhyy/IIIT-HYD-MNIST-PROJECT-CODE-CREW