**NLP PBL PROJECT**

**Neural Multilingual Machine Translation System**

**Project by:**

**Vedang Divekar 1032211877**

**Samarth More 1032221224**

**1. Introduction**

The NLP\_PBL project is a Natural Language Processing (NLP) pipeline implemented in a Jupyter Notebook (NLP\_PBL.ipynb). The project focuses on processing textual data through tokenization and modeling using modern NLP techniques. It demonstrates the workflow of loading data, configuring tokenizers, and preparing inputs for language models, with interactive progress monitoring.

**2. Objectives**

* To build an NLP pipeline that effectively preprocesses and tokenizes text data.
* To utilize subword tokenization for efficient vocabulary handling.
* To integrate transformer-based models for downstream NLP tasks.
* To provide interactive feedback during data loading and processing using Jupyter widgets.

**3. Technologies and Frameworks Used**

* **Python 3**: The programming language used for implementation.
* **Jupyter Notebook**: Interactive environment for running and demonstrating the project.
* **Hugging Face Transformers** (inferred): For tokenizer and model handling, supporting state-of-the-art transformer architectures.
* **SentencePiece or Similar Subword Tokenizer**: Indicated by the use of .spm files for vocabulary and tokenizer configuration.
* **ipywidgets**: Used to create progress bars and interactive UI elements in the notebook.
* **PyTorch or TensorFlow** (inferred): Deep learning backends for model training and inference.
* **Numpy and Pandas** (likely): For data manipulation and preprocessing.

4. Project Components and Workflow

**4.1 Data Loading**

* The project loads essential files such as source.spm (SentencePiece model file) and tokenizer\_config.json to configure the tokenizer.
* Interactive progress bars display the loading status of these files, enhancing user experience.

**4.2 Tokenization**

* The tokenizer converts raw text into token IDs using subword units, enabling the model to handle out-of-vocabulary words and large vocabularies efficiently.
* Tokenizer configuration is loaded from JSON and model files to ensure consistency.

**4.3 Model Integration**

* While the exact model architecture is not explicitly stated, the project is designed to work with transformer-based models compatible with the tokenizer.
* These models can be used for tasks such as text classification, generation, or other NLP applications.

**4.4 Interactive Progress Monitoring**

* The notebook employs ipywidgets to show progress bars and status messages during file loading and processing.
* This interactivity helps track long-running operations and improves usability.

**5. Results and Observations**

* The project successfully demonstrates the setup and execution of an NLP pipeline with tokenizer configuration and data processing.
* Interactive widgets provide real-time feedback, making the process transparent and user-friendly.
* The infrastructure supports scalable handling of large vocabularies and datasets.

**6. Challenges and Limitations**

* The exact downstream NLP task and model training details are not fully specified, leaving room for extension.
* Dependency on external files (source.spm, tokenizer\_config.json) requires proper file management.
* The notebook environment is necessary for full interactivity; running in non-interactive environments may limit usability.

**7. Future Work**

* Extend the pipeline to include explicit model training and evaluation steps.
* Incorporate more detailed data preprocessing and augmentation techniques.
* Add support for multiple NLP tasks such as sentiment analysis, named entity recognition, or machine translation.
* Develop a standalone application or API for easier deployment.

**8. Conclusion**

The NLP\_PBL project provides a foundational NLP pipeline leveraging modern tokenization and transformer-based modeling techniques. It combines efficient data handling with interactive progress visualization, making it a useful reference for NLP practitioners and learners.