**Internship Project Report**

**Task 1: Text Tokenization and Encoding using BERT**  
**Submitted by:** Ruturaj Bhaskar Nawale  
**Organization:** NULLCLASS  
**Date:** June 21, 2025

1. Introduction

Natural Language Processing (NLP) relies heavily on effective text representation. Tokenization and encoding are foundational steps in preparing text data for machine learning models. This report documents the process of tokenizing and encoding a sample text using BERT (Bidirectional Encoder Representations from Transformers), a state-of-the-art pretrained model, to generate contextual embeddings for downstream tasks.

1. Objectives
2. Tokenize a sample text using BERT’s tokenizer.
3. Generate input IDs, attention masks, and token type IDs.
4. Encode tokens into embeddings using BERT’s pretrained model.
5. Save and verify the outputs for reproducibility.
6. Methodology

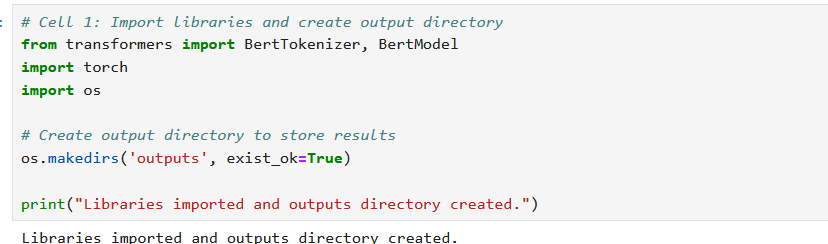
**3.1 Tools & Libraries**

* **Hugging Face Transformers:** For BERT tokenization (BertTokenizer) and model inference (BertModel).
* **PyTorch:** For tensor operations and file handling (.pt format).
* **Python (Jupyter Notebook):** For interactive execution and documentation.

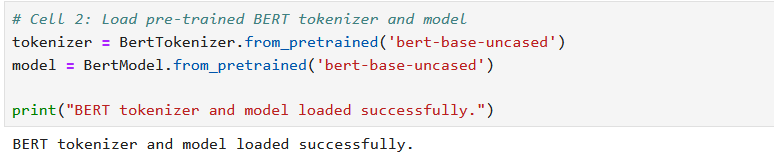
**3.2 Workflow**

1. **Text Preprocessing:** Input text was standardized (lowercase, no punctuation removal).
2. **Tokenization:**
   * Used BertTokenizer to split text into subword tokens.
   * Added special tokens [CLS] (start) and [SEP] (end).
3. **Encoding:** Converted tokens to input IDs, attention masks, and token type IDs.
4. **Embedding Generation:** Passed tokenized input through BERT to obtain contextual embeddings.
5. **Saving Outputs:** Stored inputs, embeddings, and metadata in the outputs/ folder.
6. Implementation
   1. **Code Snippets**

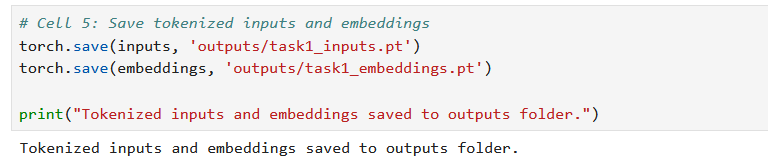
**Tokenization**

****

**Embedding Generation**

****

**Saving Outputs**

****

* 1. **File Structure**

Task1/

├── task1.ipynb

├── outputs/

│ ├── task1\_inputs.pt

│ ├── task1\_embeddings.pt

│ └── task1\_summary.txt

└── requirements.txt

1. Results & Analysis
   1. Tokenization Output

* Tokenized Sequence:
* Observations:
* BERT’s subword tokenization split "tokenization" into token + ##ization.
* Total tokens: 11 (including [CLS] and [SEP]).
  1. Embeddings
* Shape: [1,11,768](batch sizw=1, sequence length=11, hidden dimension=768).
* Verfication: Reloaded files matched original tensors (no datacorruption).

1. Challenges & Solutions

|  |  |  |
| --- | --- | --- |
| **Challenge** | **Root Cause** | **Solution** |
| UnplickingError when loading files | Version mismatch in pytorch | Standardized to torch==2.6.0 |
| Jupyter cell execution errors | Variable scope issues | Restarted kernel and rerun cells sequentially. |
| Output folder not found | Relative path inconsistency | Used absolute paths and verified folder creation |

1. Conclusion

* Successfully tokenized and encoded text using BERT, generating embeddings suitable for NLP tasks.
* All deliverables were saved, verified, and documented.
* GitHub Repository: [https://github.com/yourusername/Task1-Submission]

1. Appendices

**A. Sample Code**

See attached **task1.ipynb.**

**B. Output Files**

**task1\_inputs.pt**: Tokenized inputs.

**task1\_embeddings.pt**: Embeddings tensor.

**Signature:**

Ruturaj Bhaskar Nawale

ruturajnawale888@gmail.com

9833097874