

Comprehensive NLP Project Prompt: Punctuation Restoration

Objective

Develop, train, and evaluate a Punctuation Restoration System using a domain-specific dataset (Mental Health Conversations). The goal is to accurately restore missing punctuation (., ?, !, ,) in unpunctuated text. The project requires a comparative analysis between a fine-tuned model and a pre-trained baseline, and detailed documentation of all steps and findings.

Phase 1: Data Preparation

- Source Data:** Utilize the Response column of the provided Mental Health Conversations dataset.
- Cleaning:** Perform thorough text cleaning: lowercasing, handling special characters, standardizing whitespace, and resolving duplicates/missing values.
- Synthetic Dataset Generation (Mandatory):** Create the final training data based on one of the two architectural approaches defined below.
- Splitting:** Split the final labeled data into 80% Training, 10% Validation, and 10% Test sets.
- EDA (Exploratory Data Analysis):**
 - Analyze the distribution of word lengths in the responses.
 - Crucially, analyze the **Punctuation Label Distribution** in the training set to quantify the class imbalance (e.g., % of NONE vs. % of COMMA).

Phase 2: Model Architecture and Implementation

Select **one** of the following two architectures for your fine-tuned system, but ensure you understand and document the alternative.

Approach A (Recommended): Sequence Labeling

This approach treats punctuation restoration as a **Token Classification** task.

Input	→	Model	→	Output
Word Token		Transformer Encoder		Punctuation Label

1. Data Format (X, Y):
- * Input (X): Unpunctuated word tokens (e.g., ['therapy', 'is', 'essential', 'for']).
 - * Output (Y): Punctuation Label following the token (e.g., ['NONE', 'NONE', 'NONE', 'COMMA']).
2. Model:
- * Architecture: A pre-trained Transformer Encoder (e.g., BERT, RoBERTa, or DistilBERT) with a linear Classification Head added on top of the output layer.
 - * Process: Fine-tune the entire model on the domain-specific training data.

3. Baseline: Select a publicly available pre-trained punctuation restoration model or implement a Bi-LSTM-CRF model for comparison.

Approach B (Alternative): Sequence-to-Sequence (Seq2Seq)

This approach treats punctuation restoration as a **Text Translation** task.

Input	→	Model	→	Output
Unpunctuated Sentence		Encoder-Decoder		Punctuated Sentence

1. Data Format (X, Y):

* Input (X): Full unpunctuated string (e.g., 'therapy is essential for').

* Output (Y): Full corresponding punctuated string (e.g., 'Therapy is essential, for').

2. Model:

* Architecture: A full Encoder-Decoder Transformer (e.g., T5 or BART).

* Process: Fine-tune the entire Encoder-Decoder model on the parallel text data.

3. Baseline: A simpler, non-Transformer based language model or a Seq2Seq model trained on general text.

Phase 3: Evaluation and Documentation

1. **Metrics (Mandatory):**

- **Primary Metric: Macro/Weighted F1-Score.** This must be used due to the confirmed class imbalance found during EDA.
- **Secondary Metric:** Token-level Accuracy.

2. **Comparative Analysis:**

- Evaluate the **Fine-Tuned Model** (on your selected Approach A or B) on the Test Set.
- Evaluate the **Pre-Trained Baseline Model** on the same Test Set.
- Document the performance increase achieved by fine-tuning on domain-specific data.

3. **Documentation:** Provide a detailed report covering:

- **Methodology Rationale:** Justify the chosen architecture (A or B) and explain why the other was rejected/not pursued.
- **Challenges and Solutions:** Discuss the class imbalance challenge and how your choice of loss function or metrics addressed it.
- **Results:** Present the comparison table of all metrics for both models.
- **Error Analysis:** Perform EDA on the test results (e.g., confusion matrix for Approach A, or specific common errors in Approach B).