RoBERTa-Base Fine-Tuned for Named Entity Recognition using LoRA

This document outlines the process of fine-tuning a RoBERTa-base model for a Named Entity Recognition task. The fine-tuning leverages Low-Rank Adaptation for efficient training.

1. Dataset and Domain

The model was fine-tuned on a dataset specifically curated for Named Entity Recognition. Based on the provided entity labels (`B-ID`, `I-ID`, `B-DATE`, `I-DATE`, `B-LOC`, `I-LOC`, `B-PATIENT`, `I-PATIENT`, `B-AGE`, `I-AGE`, `B-PHONE`, `I-PHONE`), the domain of the dataset is related to **medical**. The entities the extraction of information such as patient identifiers, dates, locations, patient names, ages, and phone numbers.

The dataset is structured in a `DatasetDict` containing a training set and a test set.

- **Training set**: 1000 examples
- **Test set**: 100 examples

Each example in the dataset includes the following features:

- `text`: The input text sequence.
- `ner_tags`: The original NER tags associated with the text.

The defined NER labels follow the IOB2 (Inside, Outside, Beginning) tagging scheme:

- **O**: Outside of a named entity.
- **B-ENTITY**: Beginning of a named entity.
- **I-ENTITY**: Inside of a named entity.

2. Preprocessing & Tokenization

The tokenization process is handled by `AutoTokenizer` using the pre-trained `roberta-base` tokenizer. A custom function, `tokenize_and_align_labels`, is employed to prepare the data for the NER task.

Key aspects of the tokenization and label alignment process:

- **Tokenizer**: `roberta-base` tokenizer (a Fast Tokenizer is utilized).

- **Padding**: Texts are padded to a `max_length` of 256 tokens.
- **Truncation**: Texts longer than `max_length` are truncated.
- **Offset Mapping**: `return_offsets_mapping=True` is used to get the start and end character offsets of each token in the original text. This is crucial for aligning labels with tokens after word splitting by the tokenizer.
- **Label Alignment Strategy**:
  - The function iterates through each example and its `offset_mapping`.
  - It initializes labels for all tokens to "O".
  - For each entity span (`start`, `end`, `label`) in the original `ner_tags`:
    - It identifies all tokens that fall within or overlap with this character-based span.
    - The first token within the span is labeled as `B-LABEL`.
    - Subsequent tokens within the same span are labeled as `I-LABEL`.
    - Special tokens (where `off_start == 0 and off_end == 0`) are ignored.
  - The aligned token labels are then converted to their corresponding IDs using the `label2id` mapping.
- **Column Removal**: Original `text` and `ner_tags` columns, along with `offset_mapping`, are removed after this process, as they are no longer needed in their original form for training.

A `DataCollatorForTokenClassification` is used to create batches of data during training and evaluation.

---

3. LoRA Finetuning

Low-Rank Adaptation is a parameter-efficient fine-tuning technique that significantly reduces the number of trainable parameters in large language models like RoBERTa. Instead of fine-tuning the entire model, LoRA freezes the pre-trained model weights and injects trainable rank decomposition matrices into specific layers of the Transformer architecture.

How LoRA Works:

- **Freezing the Core:** The vast majority of RoBERTa's original parameters are not changed. This preserves its powerful general language understanding.
- **Injecting Adapters:** LoRA inserts pairs of very small matrices (the "adapters," represented mathematically as A and B) into the model's structure. These adapters are designed to learn the *changes* or *updates* needed for the new task.

- **Training Only the Adapters:** When the model is fine-tuned, only these tiny adapters are trained. This is why the number of trainable parameters drops dramatically (from over 126 million to just about 2.7 million in this project – only about 2%!).
- **Combined Power:** During operation, the original model's knowledge is combined with the learned adjustments from these small adapters to perform the new task.

Key Advantages of Using LoRA:

1. **Drastically Fewer Parameters to Train:**
    - **Smaller Model Files:** The trained LoRA adapters are tiny compared to a fully fine-tuned model, making them easy to store and share.
    - **Faster Training:** Training fewer parameters generally means training can be quicker.
    - **Less Powerful Computers Needed:** It significantly reduces the demand for GPU memory.
2. **No Slowdown During Use (Inference):** After training, the small adapter can be mathematically merged back into the original model's layers. This means the final model used for predictions is the same size and just as fast as if you had fine-tuned the whole thing, but you achieved it much more efficiently.
3. **Easily Switch Tasks:** You can train different LoRA adapters for different tasks and then "plug them in" to the same base RoBERTa model as needed, without having to store many large models.

LoRA Configuration in this Project:

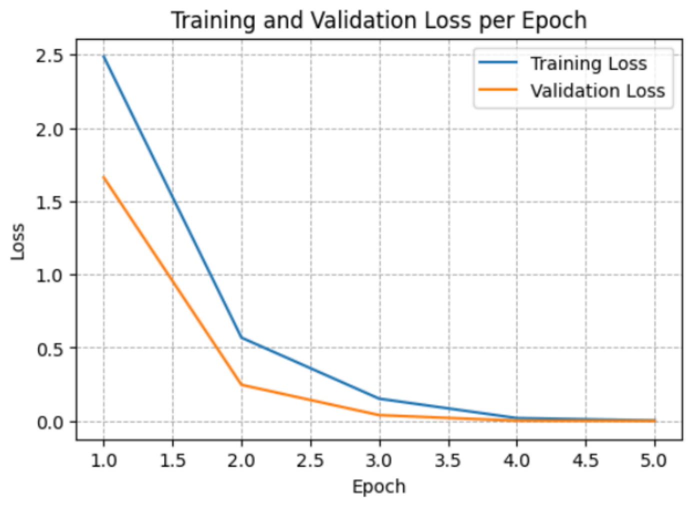
The `LoraConfig` specifies the following parameters:

- `task_type=TaskType.TOKEN_CLS`: Indicates that LoRA is being applied for a token classification task.
- `r=16`: The rank of the decomposition matrices (A and B). A smaller `r` means fewer trainable parameters.
- `lora_alpha=32`: A scaling factor for the LoRA update. The LoRA output BAx is often scaled. This means the effective learning rate for the LoRA weights is influenced by this parameter.
- `lora_dropout=0.05`: Dropout probability applied to the LoRA layers.
- `bias="lora_only"`: Only the biases of the LoRA layers are trained.
- `target_modules`: A list of the names of the modules within the RoBERTa model where LoRA adapters will be injected. These include:
    - `query`, `key`, `value`: Components of the self-attention mechanism.

- `output.dense`: The dense layer in the attention output.
- `intermediate.dense`: The dense layer in the feed-forward network.
- More specific paths like `attention.self.query` ensure precise targeting of attention components.

The `get_peft_model` function then applies this configuration to the `roberta-base` model.

Losses During training:



4. RoBERTa-base Architecture

RoBERTa (Robustly Optimized BERT Pretraining Approach) is an optimized version of BERT (Bidirectional Encoder Representations from Transformers).**1** Both models are based on the Transformer architecture, utilizing self-attention mechanisms to process input sequences and generate contextualized representations of words. However, RoBERTa introduces several key modifications to the pre-training process and data, leading to improved performance on various NLP benchmarks.

Core Architecture:

- **Transformer Encoders**: consist of a stack of Transformer encoder layers.
  - 12 encoder layers.
  - 768 hidden units in each layer.
  - 12 self-attention heads in each layer.
  - An intermediate (feed-forward) layer size of 3072.

- **Self-Attention**: Allows the model to weigh the importance of different tokens in the input sequence when representing a particular token.
- **Feed-Forward Networks**: Applied independently to each position after the attention mechanism.
- **Layer Normalization and Residual Connections**: Used to stabilize training and enable deeper networks.

Key Differences from BERT:

1. **Training Data and Size**:
    - **BERT**: Pre-trained on BookCorpus (800M words) and English Wikipedia (2.5B words).
    - **RoBERTa**: Pre-trained on a much larger and more diverse dataset, including BookCorpus, English Wikipedia, CC-News (Common Crawl News), OpenWebText, and Stories, totaling over 160GB of text. This larger dataset contributes significantly to RoBERTa's robustness.
2. **Next Sentence Prediction (NSP) Task**:
    - **BERT**: Pre-trained with both a Masked Language Model (MLM) task and a Next Sentence Prediction (NSP) task. The NSP task trains the model to predict if two input sentences are consecutive.
    - **RoBERTa**: **Removes the NSP task** during pre-training. The RoBERTa authors found that removing NSP improved downstream task performance, suggesting it might have been detrimental or not as beneficial as initially thought. RoBERTa is trained only with the MLM objective, often using full sentences sampled contiguously from documents.
3. **Masking Strategy (MLM)**:
    - **BERT**: Uses **static masking**. The input sequence is masked once during data preprocessing, and the same masked version is fed to the model across multiple epochs.
    - **RoBERTa**: Uses **dynamic masking**. The masking pattern is generated every time a sequence is fed to the model. This means the model sees different masked versions of the same sentence across different epochs, making the training process more robust and preventing the model from overfitting to specific masks.
4. **Tokenizer and Vocabulary**:
    - **BERT**: Uses WordPiece tokenization with a vocabulary size of approximately 30,000.

   - **RoBERTa**: Uses a byte-level Byte Pair Encoding (BPE) tokenizer, similar to GPT-2, with a larger vocabulary size (typically around 50,000 subword units). This allows RoBERTa to handle a wider range of text and potentially represent out-of-vocabulary words more effectively.

5. **Training Hyperparameters**:
   - **RoBERTa**: Trained with significantly larger batch sizes (e.g., 8K sequences vs. BERT's 256) and for a longer duration or more steps. It also explores different learning rate schedules and optimization strategies.

6. **Sequence Input Format**:
   - **BERT**: Uses `token_type_ids` (segment embeddings) to distinguish between different input segments (e.g., sentence A and sentence B for NSP).
   - **RoBERTa**: Does not explicitly use `token_type_ids` in the same way for sentence-pair tasks because it drops the NSP objective. When processing pairs of sentences for downstream tasks, they are typically concatenated with a special separation token.

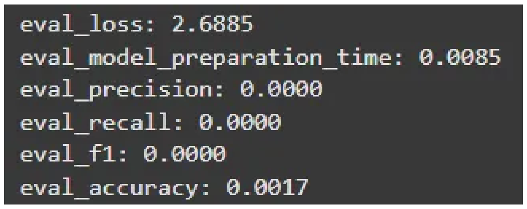
In essence, RoBERTa builds upon the successful architecture of BERT but refines the pre-training methodology by using more data, training for longer with larger batches, and making crucial changes like removing NSP and adopting dynamic masking. These optimizations generally lead to RoBERTa outperforming BERT on a wide range of NLP tasks.

---

5. Evaluations

The model's performance was evaluated on the test set using standard NER metrics: precision, recall, F1-score, and accuracy. The `seqeval` library, which is designed for sequence labeling tasks, was used for this purpose. The `compute_metrics` function processes the model's predictions and true labels, ensuring that padding tokens (where labels are -100) are ignored.

Evaluation Results Before Finetuning:



```
eval_loss: 2.6885
eval_model_preparation_time: 0.0085
eval_precision: 0.0000
eval_recall: 0.0000
eval_f1: 0.0000
eval_accuracy: 0.0017
```

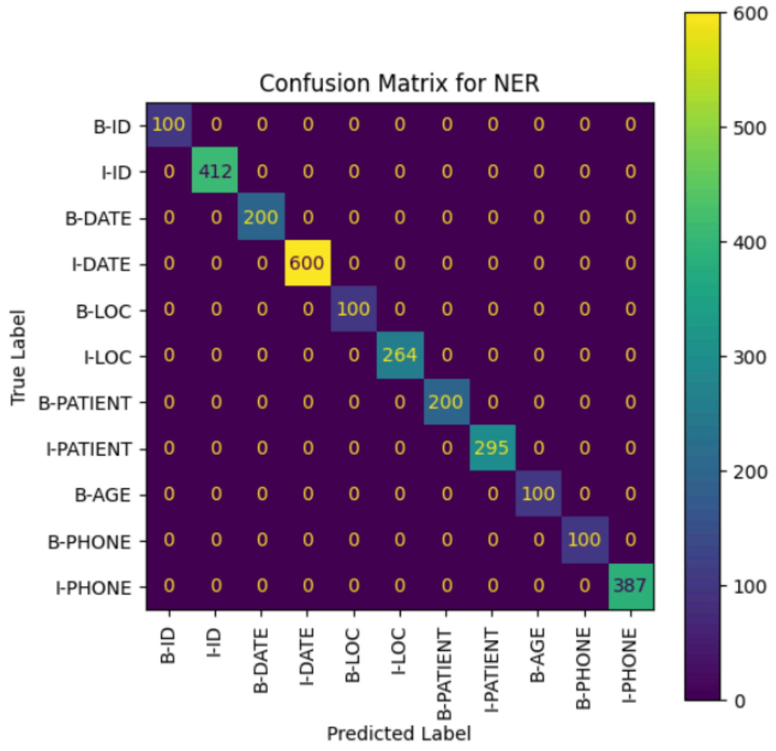
Evaluation Results After Finetuning:

The reported evaluation metrics on the test set are:

- `eval_loss`: 0.0005

- `eval_precision` : 1.0000
- `eval_recall` : 1.0000
- `eval_f1` : 1.0000
- `eval_accuracy` : 1.0000
- `eval_runtime` : 0.9076 seconds
- `eval_samples_per_second` : 110.1760
- `eval_steps_per_second` : 7.7120
- `epoch` : 5.0000

Confusion Matrix Over Entities:



Interpretation of Results:

The evaluation metrics (precision, recall, F1-score, and accuracy) are all perfect (1.0000). This indicates that the model achieved flawless performance on the provided test set of 100 samples. The `eval_loss` is also extremely low (0.0005), further supporting this.

Test Sample Prediction:

A provided test sample demonstrates the model's prediction capabilities:

```
Original Text:
locale: nb
noteType: discharge summary
translatedNoteType: epikrise
givenName: Marlon
familyName: Furu
age: 41
phoneNumber: 004777116631
city: Tveitsund
healthCareUnit: Sykehuset Innlandet, DPS Elverum-Hamar, ruspoliklinikk Hamar
diagnosis: S3722XD Contusion of bladder
birthDate: January 01. 1982
admissionDate: December 14. 2018
socialSecurityNumber: 98430148234
findings: ['regular breathing', 'low core temperature', 'clear signs of dehydration',
 'low blood pressure', 'regular pulse', 'no pain', 'full mobility']

--- Predicted Entities ---
Name: Marlon, Furu
Age: 41
Phone: 004777116631
Location: Tveitsund
Date: January 01. 1982, December 14. 2018
ID: 98430148234

--- Real Entities ---
Name: Marlon, Furu
Age: 41
Phone: 004777116631
Location: Tveitsund
Date: January 01. 1982, December 14. 2018
ID: 98430148234
```