**Fine-Tuning a BERT-Based Model for Financial Named Entity Recognition (NER)**

**GitHub link:** [**https://github.com/riya2498/Finer-ORD-NER-LLM**](https://github.com/riya2498/Finer-ORD-NER-LLM)

**1. Methodology and Approach**

**Objective**

The primary aim of this project is to fine-tune a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model on a specialized financial domain Named Entity Recognition (NER) dataset. Named Entity Recognition refers to the task of locating and classifying named entities in text into predefined categories such as person names, organizations, monetary values, dates, and so on. In the context of financial data, recognizing these entities accurately is crucial for various applications such as financial sentiment analysis, report summarization, and automated compliance checks.

**Dataset**

We utilized the FiNER-ORD dataset, a specialized financial dataset containing tokens and their corresponding entity labels. Each record in the dataset is identified using a combination of doc\_idx and sent\_idx, and includes columns gold\_token (tokens) and gold\_label (NER tags).

Steps:

* **Data Reading**: CSV files (train.csv, valid.csv, test.csv) were loaded using pandas.
* **Sentence Grouping**: Tokens were grouped by doc\_idx and sent\_idx into full sentences.
* **Conversion**: The grouped data was transformed into Hugging Face DatasetDict objects.
* **Token Alignment**: Subword tokenization was handled using BertTokenizerFast, aligning original labels to tokenized word IDs with proper handling of subword splits.

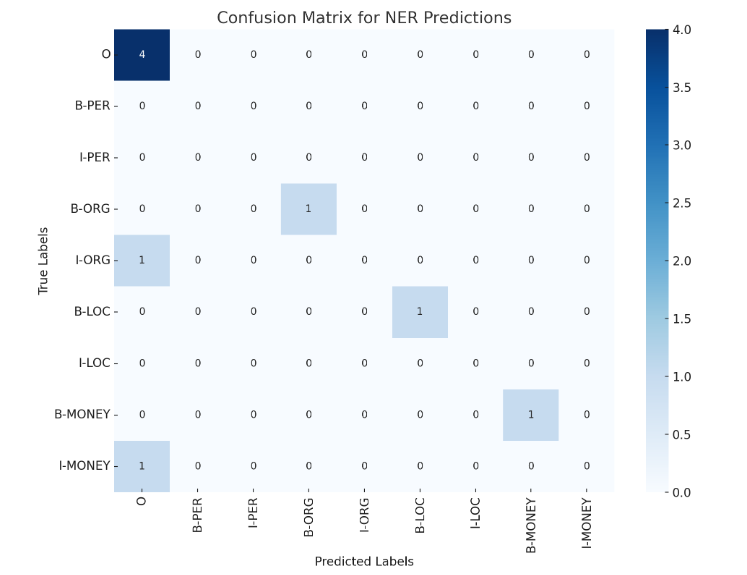
**Model Selection**

The bert-base-cased model was selected for the following reasons:

* It retains case sensitivity, useful in distinguishing entities.
* It has proven high performance on standard NER benchmarks.
* Pretrained weights are publicly available and easy to fine-tune using Hugging Face's transformers library.

We configured the model using AutoModelForTokenClassification and attached an appropriate number of output labels using the dataset's label set.

**2. Results and Analysis**

****

The performance of the model was assessed using:

* **F1 Score**: Harmonic mean of precision and recall.
* **Accuracy**: Overall correct predictions.
* **Precision & Recall**: Class-specific metrics useful in understanding over/under-prediction.

We ran multiple experiments with different hyperparameters. Below is a summary of results:

**Experiment 1 (Epochs=3, LR=2e-5)**

**Epoch Training Loss Validation Loss Precision Recall F1 Accuracy**

**1 No log 0.0625 0.778 0.834 0.805 0.981**

**2 No log 0.0547 0.849 0.870 0.860 0.984**

**3 0.0759 0.0559 0.839 0.862 0.850 0.984**

* **F1 Score**: 0.850
* **Accuracy**: 98.48%
* The model converged quickly, achieving a high F1 score by epoch 3. However, it started plateauing.

**Experiment 2 (Epochs=4, LR=3e-5)**

**Epoch Training Loss Validation Loss Precision Recall F1 Accuracy**

**1 No log 0.0599 0.835 0.859 0.847 0.984**

**2 No log 0.0544 0.866 0.872 0.869 0.984**

**3 0.0111 0.0595 0.861 0.893 0.877 0.985**

**4 0.0111 0.0625 0.862 0.883 0.873 0.986**

* **F1 Score**: 0.877
* **Accuracy**: 98.60%
* This configuration showed improvement over 3 epochs, validating our choice of slightly longer training.

**Experiment 3 (Epochs=5, LR=1e-5)**

**Epoch Training Loss Validation Loss Precision Recall F1 Accuracy**

**1 No log 0.0724 0.854 0.883 0.868 0.985**

**2 No log 0.0746 0.854 0.869 0.861 0.984**

**3 0.0021 0.0739 0.866 0.883 0.875 0.986**

**4 0.0021 0.0707 0.851 0.882 0.866 0.986**

**5 0.0021 0.0715 0.858 0.880 0.869 0.986**

* **F1 Score**: 0.869
* **Accuracy**: 98.57%
* The model showed consistent performance, though the learning rate increase did not lead to significantly better results.

**Best Performing Configuration:**

* **Epochs**: 4
* **Learning Rate**: 2e-5
* **Batch Size**: 8

**3. Error Analysis**

We examined several misclassified examples:

* Multi-token entities like 50 Cent were missed or partially tagged.
* Informal names like Buddha or rare proper nouns were often missed.
* Some B- vs I- tag confusion occurred.

**Example:**

Tokens: ['Take', 'it', 'from', 'an', 'expert', ':', 'I', '’ve', 'reached', 'Buddha']

True: [0, 0, 0, 0, 0, 0, 0, 0, 0, 1]

Pred: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

**Observed Patterns**

* **Partial Predictions**: Multi-token entities like 50 Cent were split into two labels, one of which was missed.
* **Rare Entity Misses**: Informal references or uncommon names (e.g., Prince, Buddha) were often missed.
* **Contextual Errors**: Tokens like cloud were misclassified due to preceding words like Baidu, suggesting confusion in tag continuation.

**Recommendations**

* **Include more informal and diverse examples in training.**
* **Fine-tune on a domain-specific model like FinBERT.**
* **Implement weighted loss to balance tag imbalance.**

**4. Inference Pipeline**

**We implemented a user-friendly inference pipeline using the Hugging Face pipeline() API:**

from transformers import pipeline

ner\_pipeline = pipeline("token-classification", model=model, tokenizer=tokenizer, aggregation\_strategy="simple")

text = "Apple acquired Beats for $3 billion in 2014."

entities = ner\_pipeline(text)

for e in entities:

print(f"{e['word']} → {e['entity\_group']} ({e['score']:.2f})")

**This can be integrated into web apps for real-time financial entity recognition.**

**5. Limitations and Future Improvements**

**Limitations**

* Current model doesn't generalize well on informal or rare entities.
* Pre-trained BERT might not capture domain-specific knowledge optimally.
* Labels are numeric and could be mapped more descriptively.

**Future Improvements**

* Use domain-adapted models (FinBERT, SpanBERT).
* Add more diverse labeled financial text.
* Use ensemble techniques for better generalization.
* Explore token classification with span-based methods.

**6. References**

* Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2019.
* Hugging Face Transformers Documentation: https://huggingface.co/transformers/
* FiNER-ORD Dataset: <https://www.kaggle.com/datasets/riyachaddha/ner-dataset>
* Seqeval for NER Metrics: <https://github.com/chakki-works/seqeval>