**Fine-Tuning BERT for Beer Style Classification**

**Methodology and Approach**

**Dataset Selection and Preparation**

I selected the **Beer Reviews dataset** for this assignment, which contains descriptions and categories of various beer styles. This dataset was appropriate because it provides natural language descriptions of beers, paired with their respective style labels. The classification task aligns well with the objective of fine-tuning a language model to predict categorical outputs based on text input.

**Why This Dataset?**

* **Diversity**: The dataset contains a wide range of beer styles, which helps the model learn subtle differences in text descriptions.
* **Real-World Application**: Classifying beer styles based on textual descriptions can be valuable for recommendation systems or inventory categorization.

**Preprocessing Steps**

* **Column Selection**: Retained the Description (input text) and Categories (target labels) columns.
* **Cleaning**: Removed rows with missing values and standardized category labels by stripping whitespace and converting to lowercase.
* **Filtering**: To ensure balanced classes, retained only beer styles with at least **50 samples**, resulting in **40 unique styles**.
* **Label Encoding**: Transformed categorical labels into numeric format using LabelEncoder for compatibility with model training.
* **Train/Validation/Test Split**: We stratified the data into **80% train**, **10% validation**, and **10% test** to ensure balanced class distribution across splits.
* **Subsampling**: To reduce computational overhead, especially during hyperparameter tuning, we limited the dataset to the first **1000 rows** as a proof-of-concept.

**Model Selection**

I selected **BERT-base-uncased** from the Hugging Face Transformers library for this task.

**Why BERT?**

* **Contextual Understanding**: BERT captures context through bidirectional attention, making it ideal for complex text classification tasks.
* **Pre-trained Knowledge**: Leveraging a model pre-trained on large corpora (Wikipedia, BookCorpus) allows for effective fine-tuning on domain-specific tasks with limited data.
* **Architecture Adaptability**: BERT's architecture supports additional classification layers, which simplifies adaptation for supervised learning.

**Fine-Tuning Setup**

* **Model Architecture**: I used BertForSequenceClassification, adding a classification head on top of the BERT encoder.
* **Tokenization**: I used BertTokenizer to preprocess the input text, ensuring consistency with the model's pre-training.
* **Training Environment**: The training was conducted in **Google Colab** using **GPU acceleration** for efficiency.
* **Training Loop**: Implemented using PyTorch, with **AdamW optimizer** and a **linear learning rate scheduler**. Checkpoints were saved after each epoch.

**Hyperparameter Tuning Strategy**

* **Learning Rates Tested**: [2e-5, 3e-5, 5e-5]
* **Why These Rates?**: These values are commonly recommended for fine-tuning BERT models. Testing across this range helps identify optimal convergence behavior without overfitting.
* **Epochs**: Limited to **1 epoch** per configuration during tuning to expedite the search.

**Results and Analysis**

**Hyperparameter Optimization Results**

The following table summarizes the validation performance for different learning rates:

|  |  |  |
| --- | --- | --- |
| Learning Rate | Validation Loss | Validation Accuracy |
| 2e-5 | 3.2096 | 20.00% |
| 3e-5 | 3.2096 | 20.00% |
| 5e-5 | 3.0815 | 28.00% |

**Analysis**

* The **5e-5 learning rate** provided the best performance in terms of validation loss and accuracy.
* **Accuracy** remained relatively low across configurations due to dataset limitations (small sample size, high class imbalance).
* The fine-tuned model shows potential but would benefit from additional training on a larger dataset.

**Model Evaluation on Test Set**

Using the best learning rate (5e-5), I evaluated the model on the test set:

* **Test Accuracy**: 28.00%
* **Classification Report**: Provided precision, recall and F1-scores across all classes.
* **Confusion Matrix**: Visualized to identify misclassification patterns.

**Limitations and Future Improvements**

**Limitations**

* **Dataset Size**: The subsampling to 1000 rows limits the model’s ability to generalize. A larger dataset would enhance learning.
* **Class Imbalance**: Even after filtering, some beer styles had significantly fewer samples, impacting classification performance.
* **Model Complexity**: BERT may be overparameterized for this task given the small dataset size.

**Future Improvements**

* **Scale Up Dataset**: Fine-tune on the entire dataset (10,000+ samples) for better generalization.
* **Advanced Hyperparameter Search**: Incorporate more sophisticated methods (e.g., grid search, Bayesian optimization).
* **Class Balancing Techniques**: Apply oversampling or class-weight adjustments to mitigate imbalance.
* **Experiment with Other Models**: Try lighter models (e.g., DistilBERT) or domain-specific transformers to compare performance.
* **Evaluate on Other Metrics**: Include macro-averaged metrics to better assess performance across imbalanced classes.

**References**

* Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. arXiv preprint arXiv:1810.04805.
* Wolf, T., et al. (2020). **Transformers: State-of-the-art Natural Language Processing**. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations.
* Hugging Face Transformers Documentation: <https://huggingface.co/docs/transformers>
* PyTorch Documentation: <https://pytorch.org/docs/stable/index.html>
* Beer Reviews Dataset (Wikiliq): <https://www.kaggle.com/datasets/>