Sentiment Analysis and Llama 3.1 Fine-Tuning

1. Initial Steps and Dataset Processing

Dataset

- **Source:** Excel file containing Bengali text and corresponding sentiment labels (positive, negative, neutral).
- **Loading Method:** Utilized `pandas.read_excel()` to load the dataset.

Data Preprocessing

- **Text Cleaning:** Leveraged BNLP library for processing Bengali text.
  - **CleanText Module:** Initialized with parameters `fix_unicode=True` and `unicode_norm=True` to handle Unicode errors and clean the text output effectively.
  - **Unwanted Strings Removal:** Removed the string "See Translation" and reduced duplicate punctuation marks such as ‘ ’ (Bengali full stop), ‘,’ (comma), ‘?’ (question mark), and ‘…’ (ellipsis) using `re.sub()`.
  - **Sentence Tokenization:** Used BNLP's `NLTKTokenizer` for tokenizing text at the sentence level. This was necessary because `word_tokenize` removed punctuation, which was crucial for sentiment analysis.
  - **Stemmer Issue:** Initially employed stemmers from `banglanltk`, but they truncated words undesirably (e.g., ‘ ’ to ‘ ’), leading to loss of meaning. Consequently, stemming was excluded from the preprocessing pipeline.
- **Vectorization:**
  - Used `TfidfVectorizer` to transform the cleaned text into TF-IDF feature vectors. Converted the sparse matrix to a dense format with `.toarray()` to facilitate model training.

Challenges

- Ensuring that important parts of Bengali text, especially punctuation, were preserved during preprocessing.
- Finding a reliable NLP library for Bengali text processing, which led to the exploration of both BNLP and `banglanltk`.

2. Sentiment Analysis with Traditional Machine Learning Models

Models Used

Model	Test Accuracy	Best Parameters
Logistic Regression	75%	{'C': 10, 'solver': 'liblinear'}
Multinomial Naive Bayes	65%	Default
Random Forest Classifier	75%	{'max_depth': 10, 'n_estimators': 50'}
XGBoost	60%	{'learning_rate': 0.2, 'n_estimators': 100'}
LightGBM	55%	{'learning_rate': 0.1, 'n_estimators': 100'}
LSTM	55%	Default

Stratified K-Fold Cross-Validation Results

Model	Test Accuracy	Best Parameters
Logistic Regression	0.75	{'C': 10, 'solver': 'liblinear'}
Multinomial Naive Bayes	0.65	Default
Random Forest	0.75	{'max_depth': 10, 'n_estimators': 50'}
XGBoost	0.60	{'learning_rate': 0.2, 'n_estimators': 100'}
LightGBM	0.55	{'learning_rate': 0.1, 'n_estimators': 100'}
LSTM	0.55	Default

K-Fold Results

Model	Test Accuracy
Logistic Regression	0.75
Multinomial Naive Bayes	0.65
Random Forest	0.70
XGBoost	0.60
LightGBM	0.50

Analysis

- **Best Performance:** Logistic Regression and Random Forest were the top performers with 75% accuracy, indicating that simpler models combined with effective text vectorization can be highly effective for sentiment analysis.

- **LSTM Performance:** The LSTM model exhibited poor performance (55% accuracy). The small dataset size and high variance likely contributed to its underperformance.

3. LSTM Model for Sentiment Analysis

Architecture

- **Embedding Layer:** Created word embeddings for the input text.
- **LSTM Layer:** Designed to capture sequential patterns in the text.
- **Dense Layer:** Used for sentiment classification.

Performance

- **Accuracy:** 55%
- **Epochs:** Validation accuracy plateaued at 37.5% after a few epochs, suggesting issues with model convergence.

Challenges

- **Data Format:** Conversion of TF-IDF vectorized data to a dense format suitable for LSTM was required.
- **Dataset Size:** The limited dataset size (99 rows) led to overfitting and hampered the model's ability to generalize.

4. Llama 3.1 Fine-Tuning

Initial Issues

- **Licensing Restrictions:** Faced difficulties accessing Llama 3.1 models due to licensing issues. Applied for access, but was pending approval.
- **Model Choice:** Used Dolphin 2.9.4 Llama 3.1 8B model from Hugging Face as an alternative.

Model Details

- **Source:** Dolphin 2.9.4 Llama 3.1 8B
- **Base Model:** Meta's Llama 3.1 8B with 8.03 billion parameters.
- **Description:** Dolphin was uncensored and the dataset was filtered to remove alignment and bias, thereby making the model more compliant.
- **Training Details:**
  - **Context Length:** 128K
  - **Finetuning Sequence Length:** 8192
  - **Prompt Format:** ChatML prompt template
- **Training Hyperparameters:**
  - Learning rate: 5e-06
  - Train batch size: 2
  - Eval batch size: 2

- Seed: 42
- Distributed type: multi-GPU
- Num devices: 8
- Gradient accumulation steps: 16
- Total train batch size: 256
- Total eval batch size: 16
- Optimizer: Adam with betas=(0.9,0.999) and epsilon=1e-08
- LR scheduler type: cosine
- LR scheduler warmup steps: 100
- Num epochs: 3

Fine-Tuned Training Results (Dolphin 2.9.4 Fine-Tuned Training Results of the Base Model Llama 3.1 8B

- **Epoch 1:** Loss: 0.5837, Validation Loss: 0.5814
- **Epoch 2:** Loss: 0.5525, Validation Loss: 0.5671
- **Epoch 3:** Loss: 0.5514, Validation Loss: 0.5655

Hardware Challenges

- **GPU Limitations:** GTX 1050Ti with 4GB VRAM was insufficient for model fine-tuning.
- **Training Adjustments:**
  - Forced CPU usage with `no_cuda=True` and reduced batch size.
  - Used mixed precision (`fp16=True`) and gradient accumulation (`gradient_accumulation_steps=4`).

Outcome

- **Inability to Fine-Tune:** Due to hardware limitations, the fine-tuning process was unsuccessful, with kernel crashes occurring frequently.

Future Considerations

- **Cloud-Based Services:** Use cloud-based GPUs (e.g., AWS, Google Colab) for handling large models.
- **Lighter Models:** Explore parameter-efficient fine-tuning methods like QLoRA or 4-bit quantized models.

Performance Comparison

- **Best Performing Models:** Logistic Regression and Random Forest achieved the highest accuracy at 75%.
- **Worst Performing Models:** LSTM and LightGBM both had poor performance with 55% accuracy.
- **XGBoost:** Mid-range performance with 60% accuracy.

Improvements

- **Data Augmentation:** Enhance dataset with more diverse examples to improve performance of deep learning models like LSTM.
- **Hardware Upgrades:** Utilize cloud GPUs for large model fine-tuning.
- **Exploring Lightweight Models:** Investigate parameter-efficient methods for fine-tuning large models on resource-constrained setups.

Key Insights

- **Traditional ML Models:** Logistic Regression and Random Forest outperformed more complex models like LSTM for small datasets.
- **Fine-Tuning Requirements:** Significant hardware resources are necessary for fine-tuning large models like Llama.

Limitations Faced

- **NLP Toolkit Selection:** Struggled to find an effective Bangla NLP toolkit for sentiment analysis. Explored and used BNLP and `banglanltk`.
- **Stemmer Issues:** Direct use of `banglanltk` stemmer led to loss of meaning. Decided to skip stemming.
- **Punctuation Removal:** The `word_tokenize` method removed essential punctuation; `sentence_tokenize` from NLTKTokenizer was used to resolve this issue.
- **Sparse Matrix Conversion:** Addressed issues with sparse matrices by converting to dense format using `.toarray()`.

Llama 3.1 Licensing Issues

- **Access Request:** Submitted request for access to Llama 3.1 model repositories, pending review.
- **Model Alternative:** Used Dolphin 2.9.4 Llama 3.1 8B from Hugging Face due to access issues with Llama 3.1.

Hardware Constraints

- **VRAM Limitation:** 4GB VRAM on GTX 1050Ti was insufficient for fine-tuning.
- **CPU Use:** Forced CPU use due to GPU constraints, leading to frequent kernel crashes despite adjustments.