MLops Project Report

Introduction

This report details an MLOps project focused on sentiment analysis of an English to Konkani parallel corpus. Sentiment analysis, the task of identifying and classifying subjective information in text, is crucial for understanding public opinion, conducting market research, and improving customer service. By automatically determining the emotional tone behind a piece of text, businesses and organizations can gain valuable insights into customer feedback, brand perception, and overall market trends. This project explores the effectiveness of various pre-trained transformer-based models for sentiment analysis in the Konkani language. The project addresses the challenges of working with low-resource languages like Konkani, leveraging transfer learning from pre-trained multilingual models to overcome the scarcity of labeled data. This approach allows us to adapt powerful language models, initially trained on massive datasets, to a specific language and task with relatively limited data.

Methodology

The project followed these key steps:

1. **Dataset Preparation:**

    - An existing dataset of English sentences translated into Konkani was used. This parallel corpus provides a valuable resource for cross-lingual analysis, enabling the transfer of sentiment information from English, a high-resource language, to Konkani.

- The sentiment labels (positive, negative, neutral) from the English sentences were used as annotations for their corresponding Konkani translations. This is a crucial step, leveraging the assumption that the sentiment of the original English sentence is preserved in its Konkani translation.

- The sentiment labels (positive, negative, neutral) were converted into numerical format using Sklearn's LabelEncoder. Label encoding is a crucial step in preparing categorical data for machine learning models, as these models typically require numerical inputs. By converting the textual labels into numerical values, we ensure compatibility with the chosen models.

2. **Data Splitting:** The dataset was divided into training and testing sets, with an 80:20 ratio. This split ensures that the models are trained on a large portion of the data, allowing them to learn the underlying patterns and relationships between text and sentiment, while reserving a subset for evaluating their performance. The test set provides an unbiased assessment of the model's ability to generalize to unseen data, which is critical for determining its real-world applicability.

3. **Hyperparameter Setup:** The following hyperparameters were used for model training:

   - Learning Rate: 2×10^{-5}: The learning rate is a critical parameter that controls the step size at which the model's weights are updated during training. A smaller learning rate often leads to more accurate but slower convergence, while a larger learning rate can speed up training but may result in overshooting the optimal solution.

   - Train Batch Size: 8: The batch size determines the number of training examples used in each iteration of the training process. Smaller batch sizes can provide a more regularizing effect and may be beneficial for smaller datasets, while larger batch sizes can speed up training and improve gradient estimation.

   - Evaluation Batch Size: 8: Similar to the training batch size, the evaluation batch size defines the number of examples used during the evaluation phase.

Consistent batch sizes between training and evaluation can help ensure a fair comparison of model performance.

- Training Epochs: 4: An epoch represents one complete pass through the entire training dataset. The number of epochs is a key factor in determining how long the model trains. Too few epochs may lead to underfitting, where the model fails to learn the training data adequately, while too many epochs can result in overfitting, where the model learns the training data too well and performs poorly on unseen data.

- Weight Decay: 0.01: Weight decay is a regularization technique that adds a penalty term to the loss function, discouraging the model from assigning excessively large weights to any single feature. This helps to prevent overfitting and improve the model's generalization ability.

4. **Model Selection and Training:**

   - Several pre-trained transformer models were selected for evaluation:

     - AI4Bharat's IndicBERT: IndicBERT is a family of models specifically trained on a large corpus of Indian language text. It is designed to better capture the nuances and complexities of these languages compared to general-purpose multilingual models.

     - Google MuRIL Based Cased: MuRIL is another powerful model specifically trained for Indian languages. The "cased" variant indicates that it preserves the case of words, which can be important for certain NLP tasks.

     - Google MuRIL Based Multilingual Cased: This is a multilingual version of MuRIL, further expanding its capabilities to handle a wider range of languages in addition to the Indian languages it was primarily trained on.

     - Ibraheem Muhammad Moosa XLMIndic Base Uniscript: This model focuses

on representing Indian languages in a unified script, which can be beneficial for cross-lingual understanding.

- Ibraheem Muhammad Moosa XLMIndic Base Multiscript: This model is trained on the original scripts of the Indian languages, allowing for a comparison of how different scripting approaches affect model performance.

- Custom BERT Model: A custom BERT model was also created with the following configuration: 256 hidden node size, 4 layers, 4 attention heads, 512 intermediate size, and a vocabulary size based on the tokenizer output. It used the same hyperparameters as the other models in the experiment.

- The models were fine-tuned on the Konkani sentiment analysis task. Fine-tuning involves taking a pre-trained model and adapting it to a specific downstream task by training it on a smaller, task-specific dataset. This process allows the model to leverage the knowledge it gained during pre-training, leading to improved performance and faster convergence compared to training a model from scratch.

5. **Tokenization:**

- The input text data was tokenized using the AutoTokenizer from the Hugging Face Transformers library. This ensured that the text was processed in a format suitable for each specific model (SentencePiece and Tiktoken). Tokenization is the process of breaking down text into smaller units, such as words or sub-word units, which can then be represented as numerical values and fed into the model. The AutoTokenizer automatically selects the appropriate tokenizer for each model, ensuring optimal performance.

- Tokenized sequences were padded to a maximum length of 128 to maintain consistent input sizes for the models. Padding is a technique used to ensure that all input sequences have the same length, which is required by most deep learning models. Sequences shorter than the maximum length are padded with

special tokens, while longer sequences may be truncated.

6. **Evaluation Metrics:** Model performance was evaluated using accuracy and the average weighted F1 score. Accuracy measures the overall correctness of the model's predictions, indicating the proportion of correctly classified examples. While accuracy is a straightforward metric, it can be misleading in cases where the class distribution is imbalanced. The F1 score provides a balanced measure of precision and recall, particularly useful for imbalanced datasets. Precision measures the proportion of correctly predicted positive examples out of all examples predicted as positive, while recall measures the proportion of correctly predicted positive examples out of all actual positive examples. The weighted average F1 score calculates the F1 score for each class and then averages them, weighting each class's score by its support (i.e., the number of true instances for that class). This provides a more robust evaluation metric for multi-class classification problems with potential class imbalance.

Results

The project results, as provided, indicate that Google's MuRIL Based Cased model achieved the best performance, with a maximum accuracy of 60.75%. The table below summarizes the key results.

Model	Accuracy	F1 Score	Loss
ai4bharat/indic-bert	0.49875	0.44373	1.00944
google/muril-base-cased	0.60750	0.60643	0.92359
bert-base-multilingual-cased	0.57250	0.57322	1.06046
ibraheemmoosa/xlmindic-base-multiscript	0.51625	0.52110	1.14770
ibraheemmoosa/xlmindic-base-uniscript	0.40375	0.27310	1.07705
Custom Bert Model	0.35850	0.19690	1.09450

Conclusion

This project demonstrates the application of several pre-trained transformer models to the task of sentiment analysis for Konkani, a low-resource language. The results indicate that while transfer learning provides a viable approach, the performance of the models is still relatively modest. Google's MuRIL Based Cased model showed the strongest performance, achieving an accuracy of 60.75%. This suggests that while these models can capture some sentiment information in Konkani, there is still significant room for improvement.

References

1. google-bert/bert-base-multilingual-cased · Hugging Face. (2024, March 11).

Huggingface.co. https://huggingface.co/google-bert/bert-base-multilingual-cased

2. google/muril-base-cased · Hugging Face. (2018). Huggingface.co.

https://huggingface.co/google/muril-base-cased

3. ai4bharat/indic-bert · Hugging Face. (n.d.). Huggingface.co.

https://huggingface.co/ai4bharat/indic-bert

4. ibraheemmoosa/xlmindic-base-uniscript · Hugging Face. (2024, March 21).

Huggingface.co. https://huggingface.co/ibraheemmoosa/xlmindic-base-uniscript

5. ibraheemmoosa/xlmindic-base-multiscript · Hugging Face. (2024, March 21).

Huggingface.co. https://huggingface.co/ibraheemmoosa/xlmindic-base-multiscript