In this submission I continue with the literature reviews as per the specifications given in the Task 2. This is being done however, in conjunction with me trying out the BERT/BART models as mentioned in the meeting but have been facing quite some difficulty in getting the environment setup. Moving forward based on the insights drawn from the meetings- I have started employing the huggingface sentiment analysis using the guide given here **(https://huggingface.co/blog/sentiment-analysis-python)**- The focus as of now is using the pre-trained models and at the same time analyze how we can train the models based on our dataset.

Fine Tuning BERT models for Classification:
(https://link.springer.com/chapter/10.1007/978-3-030-32381-3_16)

Fine-tuning the Bidirectional Encoder Representations from Transformers (BERT) model for text classification tasks has become a standard practice in natural language processing (NLP). BERT, a transformer-based model pre-trained on a massive corpus of text, has shown remarkable performance across various NLP applications, including text classification. This paper explores the techniques and strategies for effectively fine-tuning BERT for text classification tasks, providing insights into the technical details and mathematical foundations of the process.

Some Technical Details

Pretrained BERT Model: The starting point for fine-tuning is a pre-trained BERT model. This model captures deep contextual information from a large text corpus.

Classification Layer: BERT is a transformer model with multiple layers of self-attention and feedforward networks. For text classification, a classification layer is added on top of the pre-trained BERT. This layer typically includes one or more dense (fully connected) layers, followed by a softmax activation function to produce class probabilities.

Mathematically, this can be represented as:

$P(\text{class} \mid \text{text}) = \text{softmax}(W * h\_cls + b)$

Where $P(\text{class} \mid \text{text})$ is the probability of the text belonging to a specific class, h_cls is the representation of the [CLS] token (a special token used in BERT), W represents the weights of the classification layer, and b is the bias term.

Loss Function: The choice of the loss function is crucial in fine-tuning. Cross-entropy loss is commonly used for text classification. Given a ground-truth class label y, and predicted class probabilities $P(\text{class} \mid \text{text})$, the loss can be defined as:

$L = -\sum y * \log(P(\text{class} \mid \text{text}))$

Fine-Tuning on Task-Specific Data: Fine-tuning involves training BERT on a task-specific dataset. The model's weights are updated during this process, optimizing its ability to classify text according to the target labels. The key is to choose an appropriate learning rate, batch size, and number of training epochs for effective fine-tuning.

Warm-Up Learning Rate: BERT fine-tuning often employs a warm-up learning rate schedule. This means that the learning rate starts small, increases gradually, and then decays. This schedule helps stabilize training and can prevent catastrophic forgetting, where the model forgets what it has learned during pre-training.

Mathematically, the learning rate schedule can be defined as:

$$lr = d\_model^{(-0.5)} * \min(step\_num^{(-0.5)}, step\_num * warmup\_steps^{(-1.5)})$$

Gradient Clipping: To prevent exploding gradients during training, gradient clipping is often applied. This technique ensures that gradients are scaled if they exceed a certain threshold.

Mathematically, gradient clipping can be represented as:

$$g = \min(grad, clip\_threshold)$$

Batch Size: The choice of batch size is essential. A larger batch size can lead to faster convergence, but it requires more memory. It's essential to find a balance that suits the available hardware.

Strategies for Effective Fine-Tuning

Fine-tuning BERT for text classification involves several strategies to optimize its performance:

Task-Specific Data: The fine-tuning dataset should be representative of the target task. It's essential to have high-quality labeled data for the task to ensure successful fine-tuning.

Classification Layer Initialization: The weights of the classification layer are often initialized randomly. However, you can also initialize them from a pre-trained model specific to your task for better performance.

Hyperparameter Tuning: Experimenting with hyperparameters such as learning rate, batch size, and the number of epochs is essential to find the optimal combination for your specific task.

Model Size: BERT comes in various sizes, from base to large models. Smaller models may be sufficient for some tasks and can be more computationally efficient.

Mathematical Foundations

The mathematical foundations of fine-tuning BERT for text classification revolve around the choice of loss functions, optimization techniques, and learning rate schedules. These ensure that the model learns the task-specific patterns from the data while preserving the valuable pre-trained knowledge.

The cross-entropy loss function used for text classification is well-established, as mentioned earlier. The optimization process involves finding the optimal weights for the classification layer by minimizing this loss function.

The learning rate schedule, including the warm-up phase, regulates the update steps and helps control the convergence of the model during fine-tuning. It ensures that the model learns at an appropriate rate, preventing it from diverging or overfitting.

Conclusion

The mathematical foundations of cross-entropy loss, learning rate schedules, and gradient clipping ensure that the fine-tuning process is efficient and effective. By following these strategies and technical details, we can harness the full potential of BERT for text classification across a wide range of NLP applications.

Next after having a look at all the BERT related models I was able to find a study which compares the BERT model against the traditional methods we have currently been exploring like clustering, tf-idf analysis, etc.

Comparing BERT against traditional machine learning text classification: (https://arxiv.org/abs/2005.13012)

Traditional Machine Learning Approaches: The paper begins by introducing traditional machine learning methods used for text classification. These include feature engineering techniques, such as TF-IDF (Term Frequency-Inverse Document Frequency) and word embeddings, in combination with classifiers like SVM or Naive Bayes.

Pre-trained BERT Model: BERT, has gained prominence due to its ability to capture contextual information and semantic relationships in text. The pre-trained BERT model is a stack of transformer layers, with attention mechanisms and feedforward networks. In the context of text classification, BERT is fine-tuned on a labeled dataset to adapt it to the specific classification task.

Feature Extraction with BERT: To use BERT for text classification, the paper describes the process of feature extraction as described earlier. BERT tokenizes input text and produces contextualized embeddings for each token. These embeddings can be aggregated in various ways to represent the entire document, such as taking the mean or using the [CLS] token embedding.

Mathematically, this can be represented as:

$$\text{BERT}(\text{text}) = [e_1, e_2, ..., e_n]$$

Where e_i represents the embedding for the i-th token in the text.

Classification Layer: Both traditional machine learning approaches and BERT-based methods employ a classification layer. In the case of BERT, this layer includes one or more dense layers with a softmax activation function to produce class probabilities.

Mathematically, the classification layer can be represented as:

$$P(\text{class} \mid \text{BERT}(\text{text})) = \text{softmax}(W * h_{cls} + b)$$

Where $P(\text{class} \mid \text{BERT}(\text{text}))$ is the probability of the text belonging to a specific class, h_{cls} is the representation of the [CLS] token, W represents the weights of the classification layer, and b is the bias term.

Loss Function: The choice of loss function is essential for both traditional and BERT-based methods. Cross-entropy loss is commonly used in text classification, measuring the dissimilarity between predicted class probabilities and ground-truth labels.

Mathematically, the cross-entropy loss can be represented as:

$$L = -\sum y * \log(P(\text{class} \mid \text{BERT}(\text{text})))$$

Where L is the loss, y is the ground-truth label, and $P(\text{class} \mid \text{BERT}(\text{text}))$ is the predicted probability of the text belonging to a specific class.

Mathematical Foundations

The paper provides a detailed overview of the mathematical foundations for both traditional and BERT-based text classification. It highlights that traditional methods rely on engineered features and simple classifiers, often using techniques like TF-IDF and bag-of-words representations. BERT, on the other hand, takes advantage of deep learning and transformer architecture.

BERT's mathematical foundations revolve around the fine-tuning process. During fine-tuning, the model's parameters are updated to minimize the cross-entropy loss, making it specific to the classification task. The choice of loss function, learning rate, batch size, and the number of training epochs are critical in the fine-tuning process.

Comparative Analysis

The paper conducts a comparative analysis of traditional machine learning methods and BERT for text classification, taking into account various factors:

Performance: **BERT often outperforms traditional methods in terms of accuracy**, especially when there's a substantial amount of labeled training data available. It excels in capturing semantic relationships and contextual information.

Data Size: BERT's performance is highly data-dependent. With a limited labeled dataset, traditional methods may be more robust, while BERT requires larger training datasets for optimal performance.

Fine-Tuning Overhead: Fine-tuning BERT can be computationally intensive, making it less practical for small-scale projects.

Resource Requirements: Traditional methods are often more memory-efficient and faster to train compared to BERT, which requires substantial computational resources.

Conclusion

The paper "Comparing BERT against Traditional Machine Learning Text Classification" offers a comprehensive analysis of the advantages and limitations of using BERT for text classification. It underscores that BERT is a powerful tool for tasks where large labeled datasets and high accuracy are critical, thanks to its ability to capture semantic relationships and context. However, traditional machine learning methods remain valuable, especially in scenarios with limited labeled data, limited computational resources, or the need for efficiency and simplicity.

In our case however, with the soon to come access to better computation power and a substantial amount of data to train the model on, BERT might be the way to move forward.