Word Template in CEURART for One Column

1. Approach

This work involves the fine-tuning of pre-trained language models using the Hugging Face Transformers library to address a classification task. The models were evaluated in both fine-tuned and zero-shot settings, enabling a comparative study of their performance. The dataset used contained columns `text` and `label`, representing the input data and corresponding labels, respectively. A stratified data split ensured balanced distribution across training and testing subsets.

Github Repository Link: https://github.com/2fundun07/ceng463_hw2

2. Data Preparation

The dataset was initially loaded in CSV format from TSV format leveraging **load_dataset** function from **datasets**. Then, data is converted into a **pandas Dataframe** for manipulation. As the assignment paper suggested, a stratified split was performed in order to preserve class distribution percentages between training and test datasets. The data is partitioned into ninety to ten percentages as training and test data respectively. Key transformations included

- Tokenization using a pre-trained tokenizer.
- Creation of Hugging Face **DatasetDict** objects for streamlined pipeline integration.
- Setting random state for reproducibility and consistency in experimentation.

The columns **text** (original language) and **text_en** (translated English text) were both leveraged, enabling cross-linguistic evaluation of model performance.

2.1. Dataset Statistics

Table 1
Italian Orientation Dataset

Class Label	Count (Training Set)	Count (Test Set)
0	1159	208
1	1871	129
Total	3030	337

The ideology dataset has a noticeable class imbalance because Class 1 has **61.7%** of both the training and test instances, while Class 0 only has **38.3%**. This dataset is moderately imbalanced, but the ratio is consistent between the training and test sets.

The imbalance might affect the model's performance if Class 1 is the minority class. The imbalance problem can be addressed by using class weights or resampling (i.e. over-sampling the minority class).

Table 2
Italian Power Dataset

Class Label	Count (Training Set)	Count (Test Set)
0	4415	491
1	2648	294
Total	7063	785

The power dataset has a similar class imbalance to ideology dataset because Class 1 has **62.5%** of both the training and test instances, while Class 0 only has **37.5%**. This dataset is moderately imbalanced, but the ratio is consistent between the training and test sets.

The imbalance might still present challenges if the models used are not designed to handle imbalanced data. The imbalance problem can be addressed by balanced accuracy in addition to ones that are suggested for the ideology dataset.

3. Experimental Setup

3.1 Model Loading

The model for the fine-tuning is an **XLM-RoBERTa** based **FacebookAI/xlm-roberta-base** which is a multilingual transformer model suitable for text classification tasks. The model is pre-trained on vast multilingual corpora, hence it is ideal for fine-tuning on a custom dataset such as these datasets are. However it is a large model, which makes it computationally expensive and causes challenges in Colab environment since model requires a significant memory itself.

In the tokenization part, padding is set to maximum length and truncation is set to true to ensure that the input data fits the model's input size requirements.

The model used for inference is **facebook/bart-large-mnli** which is a pre-trained model for **zero-shot-classification** tasks. Its multi-langual capabilities make it a good fit for task involving multiple languages.

3.2 Setup

The training arguments for fine-tuning task are configured using Hugging Face's **TrainingArguments** from **transformers** library.

- **Learning Rate :** Set to **2e-5**, which is typical value for fine-tuning transformer models. Because it helps the model to converge more gradually.
- **Evaluation Strategy :** Set to **epochs** which means that the model will be evaluated at the end of each training epoch instead of each **step**.
- **Epoch Number :** Set to **3**. The larger values typically lead to better model performance unless the model is overfitting where both the accuracy and validation loss are increasing. However, the issue I had was beyond this, larger values caused Colab memory to crash.
- **Batch Size :** Set to **16** which is the number of samples that will be process at once during training for each GPU or the computation device. The larger values caused memory crash.
- **Weight Decay :** Set to **0.01**, it's a regularization technique to avoid overfitting by penalizing the large weights the model.
- **FP16 :** Set to **True**, it enables mixed-precision training using 16-bit floating-point numbers instead of the standard 32-bit. It speeds up training and reduce memory usage on GPUs without sacrificing too much accuracy.

The inference setup leverages **pipeline** from Hugging Face's **transformers** library. The pipeline is set to **zero-shot-classification** which automatically tokenizes inputs, processes them through the model, and decodes the predictions.

- **Batch Inference:** set to **16**, processing data in batches to improve computational efficiency when running inference on the GPU instead of single run.

Finally, the **classification reports** are included to analyse the performance of models in greater detail beyond just accuracy. It provides insights into model behaviour for each class, helping to identify areas for improvements.

4. Results

Table 3
Italian Orientation Dataset Results

Model	Dataset	Accuracy	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Class 0)	F1- Score (Class 1)
Fine-Tuned	English	69.73	0.74	0.69	0.33	0.93	0.45	0.79
Inference	English	62.00	0.52	0.64	0.22	0.88	0.31	0.74
Inference	Original	58.00	0.41	0.62	0.22	0.80	0.29	0.70

The f1-score for Class 1 (0.79) is significantly higher, indicating that the fine-tuned model handles the positive class (label 1) more effectively. However, the recall for Class 0 (0.33) is relatively poor, showing that the model struggles to identify instances of the negative class (label 0).

The model performs relatively well on Class 1 (f1-score: 0.74) but has low precision (0.52) and very poor recall (0.22) for Class 0, indicating a strong bias toward predicting Class 1.

The precision, recall, and f1-scores for Class 0 are even lower than in the English text, suggesting that the zero-shot model is less effective on non-English data, possibly due to less robust multilingual handling or semantic mismatches in the original text.

The fine-tuned model consistently outperformed the zero-shot model, as expected, due to being trained on task-specific data. The zero-shot model shows language dependence: performance is better on translated English ("text_en") compared to the original text ("text").

Conclusion is;

- **Fine-tuning** is essential for achieving higher accuracy and balanced performance across classes, especially when labeled task-specific data is available.
- **Zero-shot models** are a viable alternative when fine-tuning is not possible, but they exhibit **biases** and **language dependencies** that should be carefully evaluated before deployment.

Table 4
Italian Power Dataset Results

Model	Dataset	Accuracy	Precision (Class 0)	Precision (Class 1)	Recall (Class 0)	Recall (Class 1)	F1-Score (Class 0)	F1- Score (Class 1)
Fine-Tuned	Original	77.07	0.80	0.84	0.84	0.65	0.82	0.68
Inference	English	41.00	0.64	0.38	0.13	0.88	0.21	0.53
Inference	Original	44.00	0.64	0.38	0.22	0.79	0.33	0.51

The fine-tuned model clearly outperforms the zero-shot model on all metrics, highlighting the importance of task-specific fine-tuning. The zero-shot model struggles with Class 0 recall, leading to poor overall performance, especially in English text. On the original text, the zero-shot model shows slight improvement, but its overall performance remains inadequate compared to the fine-tuned model.

Conclusion is;

- The **fine-tuned model** is the clear choice for tasks requiring high accuracy and balanced performance across classes. It achieves **77.07% accuracy**, significantly higher than the zero-shot model.
- The **zero-shot model's limitations** are evident in both English and original texts, with low accuracy (41%-44%) and severe struggles in identifying the negative class.
- If labeled data is unavailable, the zero-shot model can provide a quick baseline, but its performance issues should be carefully considered before deployment.