Technical Report: Ethos ML Reasoning Challenge

This document outlines the design, approach, and performance of a machine learning system developed for the **Ethos ML** logic and reasoning challenge. The final implementation uses a finetuned Transformer model to solve problems by treating the task as a multiple-choice classification problem.

Pipeline Design :

1. **Load data**
    - From **train.csv** and **test.csv**
2. **Create text pairs for training**
    - Combine:
      topic + problem_statement + answer_option_i
    - Use that as the input text for embeddings
    - Label = **1** if option i is correct, else **0**
    - → Gives a classification dataset (option relevance)
3. **Generate embeddings**
    - Using **SentenceTransformer("paraphrase-MiniLM-L6-v2")** (fast, good balance)
    - Create one embedding per text pair.
4. **Train a model**
    - **RandomForest** or **LogisticRegression**
    - Input: **embedding**
    - Target: label (1 / 0)
5. **Predict on test.csv**
    - For each test question:
        - Encode 5 (**problem + option_i**) combinations
        - Predict probability that it's the correct option
        - Pick highest-probability option → **correct_option_number**
        - Also choose solution = **answer_option_**{predicted}
6. **Save output.csv**
    - topic, problem_statement, solution, correct option

1. System Design and Architecture

The overall framework is a supervised learning pipeline built using **PyTorch** and the **Hugging Face transformers** library. The core of the system is a pre-trained language model that is finetuned to select the correct answer from a list of five options.

Key Components:

• **Model Architecture:** The system is built upon **roberta-large**, a powerful, pre-trained Transformer model with **355 million parameters**. This model was chosen for its strong performance on a wide range of **Natural Language Understanding (NLU)** tasks, providing a robust foundation for learning the nuances of the problem set.

• **Model Head:** We specifically use the **RobertaForMultipleChoice architecture**. This specialized head is placed on top of the **base RoBERTa** model. Its design is optimal for this task, as it is built to process a single question paired with multiple potential answers, and output a single classification score indicating which answer is the most plausible.

• **Tokenizer:** A **RobertaTokenizer** is used to preprocess the text data. It converts the questions and answer options into the numerical token ID format that the **RoBERTa model** requires. It also handles padding and truncating sequences to a uniform length of **256 tokens**.

• **Training Framework:** The entire pipeline is implemented in **PyTorch**. An **AdamW optimizer** was used to finetune the model's weights over three epochs with a learning rate of **1e-5**.

2. Problem Decomposition & Reasoning Approach

While an initial **'Agentic AI'** approach with multi-step reasoning was considered, a more direct and reliable classification-based strategy was implemented for the final solution.

Problem Formulation

Instead of generating an answer from scratch, the problem was reframed from a generative task to a discriminative task: **'Given a problem and five options, which is the correct one?'** This simplifies the problem space and leverages the powerful pattern-recognition capabilities of Transformer models.

Input Representation

The **'decomposition'** of the problem happens at the data-processing stage. For each question in the dataset, the system creates five distinct input sequences. Each sequence is a combination of the shared context **(the topic and problem statement)** and one of the unique answer options.

[Topic + Problem Statement] </s> [Answer Option 1]
[Topic + Problem Statement] </s> [Answer Option 2]
...and so on for all five options.

Reasoning as Plausibility Scoring

The **'reasoning'** is implicitly learned by the model during the finetuning process. By training on train.csv, the model learns the deep semantic relationships that make a given **[Question + Answer]** pair plausible or **'correct'**. During inference, the **RobertaForMultipleChoice** model processes all five input sequences simultaneously and outputs a single logit for each, representing a score. A **softmax function** is then applied to these logics, and the option with the highest resulting probability is chosen as the final answer. The model is therefore trained to be an expert at scoring the plausibility of each answer in the context of the problem.

3. Results and Evaluation

Performance Metrics

The primary metric for evaluating the model's performance is Accuracy, defined as the percentage of questions for which the model predicts the correct option number. The training process was monitored by observing the cross-entropy Loss, which consistently decreased over the three epochs, indicating that the model was successfully learning from the training data.

Evaluation Results

Final Accuracy on Test Set: **76.66%**

Inference Time and Computation Power

Inference Time: The finetuned model is highly efficient during prediction.

- Inference Speed: During testing, the model achieved an average inference speed of approximately 14 predictions per second.
- Total Inference Time: For the full test set, the total time to generate all predictions is estimated to be approximately 3 minutes. This demonstrates that the final solution is not only accurate but also computationally efficient at producing results.

Computation Power: The model was trained and evaluated on a standard **Google Colab** instance, utilizing a single **NVIDIA T4 GPU** with approximately **15GB** of VRAM. **roberta-large** is a substantial model (**355 million parameters**), which required a relatively small training batch size of 4 to fit within the GPU's memory constraints. It took around 40 to 45 minutes to train the model.

(LR = 1e-5 and EPOCHS: 10)

Analysis and Key Insights

• **Model Choice:** The decision to upgrade from **bert-base-uncased** to **roberta-large** was a key factor. The larger model capacity and improved pre-training objective of **RoBERTa** provide a stronger foundation for learning the complex patterns in the dataset.

• **Batch Size:** Training a large model like **roberta-large** on a standard Colab GPU required reducing the batch size from 8 to 4 to avoid **OutOfMemory errors**. This is a common and necessary trade-off between model size and hardware limitations.

• **Approach Limitations:** The classification approach is highly efficient and robust. However, it is fundamentally limited by the provided answer choices. It cannot generate an answer that is not in the options and can be fooled by cleverly worded but incorrect **'distractor'** options.

Future Improvements

• **Hyperparameter Tuning:** Further experimentation with the learning rate, batch size, and number of training epochs could yield incremental performance gains.

• **Model Experimentation:** Given the **13B parameter** limit, testing even more modern and capable encoder models like **deberta-v3-large** could provide another boost in accuracy.

• **Cross-Validation:** To get a more robust measure of the model's performance and to tune hyperparameters more effectively, a **k-fold cross-validation** strategy could be implemented on the training dataset.