Project Initial Report
Phase 2

CS550 Project: Adaptive Wait-k Policy for Simultaneous Text-to-Text Machine Translation Based on RL

Group Members:

Tanmay Kumar Shrivastava (12241870)

Darsh Mahajan (12240500)

Shavaneeth Gourav (12241100)

Under the Guidance of:

Dr. Rajesh Kumar Mundotiya

Github: https://github.com/Tanmay-IITDSAI/MLProject

Index

Introduction to the Problem Statement

Base Model:

- The project starts by implementing a Mixture-of-Experts (MoE) Wait-k policy as the backbone model.
- The Wait-k policy enables each head in the multi-head attention mechanism to handle different levels of latency during translation.
- This policy allows the model to balance between waiting for more input tokens and translating simultaneously, essential for simultaneous machine translation tasks.

Reinforcement Learning (RL) Integration:

- **State:** In the RL framework, the state includes the source tokens already processed and the target tokens generated so far.
- **Action:** The action space defines selecting a value of k, determining how many source tokens to wait before generating a translation.
- **Reward Function:** The reward combines the BLEU score (for translation quality) and Average Lagging (for latency).
- The model is penalized for high latency or low-quality translations.
- We will also test the ROUGE score, which assesses coverage and informativeness, critical for capturing the core meaning of the input content.
- **Agent:** A policy-based RL agent will be employed to predict the optimal value of k. The policy is learned using policy gradient methods.

Abstract

The project aims to design a simultaneous translation model using the Mixture-of-Experts (MoE) Wait-k policy for efficient handling of varying latency levels. By integrating reinforcement learning (RL), the model dynamically selects the optimal number of source tokens to wait before generating target tokens. The reward function balances translation quality and latency, with BLEU and ROUGE scores guiding the training. This hybrid approach ensures high-quality, low-latency translation output. *Future improvements will include fixing dimensionality errors and fine-tuning the reward function.*

Workflow

1. Dataset Preparation

- We created a custom dataset class for simultaneous translation, which processes both source and target sentences.
- Source sentences are tokenized and padded to a fixed maximum length, and decoder input is shifted to align with target sentences.
- AutoTokenizer from the T5 model was used to handle input tokenization.

2. Mixture-of-Experts (MoE) Wait-k Policy

- The **`MOEWaitKPolicy`** class implements the Mixture-of-Experts Wait-k mechanism, allowing the model to leverage multiple experts for different heads of attention.
- The **`AdaptiveWaitKMode`**`l` class integrates the encoder and decoder components, incorporating the adaptive wait-k policy into the translation process.
- The **`SimultaneousTranslationModel`** class encapsulates the entire model architecture, facilitating the forward pass through the network.

3. Model Architecture

- A combined model was built that integrates the Adaptive MoE Wait-k policy with a reinforcement learning policy.
- The model is trained over multiple epochs using the Adam optimizer and Cross-Entropy loss. The training loop involves passing batches through the model, calculating the loss, and updating the weights.

4. Training Process

- The training loop includes cross-entropy loss to optimize the model, with backpropagation handled using the Adam optimizer.
- BLEU and ROUGE scores are calculated during the evaluation phase to monitor both translation quality and coverage.
- Training metrics are visualized using BLEU and ROUGE scores plotted over epochs to observe improvements.
- During training, the model is evaluated using BLEU and ROUGE scores to assess translation quality and performance. Average scores are computed and plotted over the epochs to visualize the model's learning curve.

Inferences

The implementation demonstrates a structured approach to building a simultaneous translation model using modern machine learning techniques. The integration of the Mixture-of-Experts mechanism and reinforcement learning principles shows promise for optimizing translation quality and reducing latency. However, various errors were encountered during implementation.

Identified Errors

- **Tensor Size Mismatch:** The most critical error is a size mismatch between tensors in the `forward` method of the `MOEWaitKPolicy` class. Specifically, when attempting to combine expert outputs with the attention mask, the sizes of the tensors do not align, causing a runtime error.
- **Syntax Errors:** There are minor syntax issues, such as misplaced commas and missing parentheses, which would prevent the code from executing properly.