6998 Project Proposal

Title: From-Scratch Transformer for Extractive Question Answering

Participant: Wong Tsz Yat (tw3047)

Objective: This project aims to implement a custom Transformer architecture inspired by "Attention Is All You Need" specifically for an **extractive question-answering (QA)** task. By building the core components from the ground up—multi-head attention, positional embeddings, and feed-forward layers—this project will provide a hands-on understanding of modern deep learning techniques for natural language processing.

1. **Deep Learning Fluency**: Demonstrate an end-to-end workflow, from data processing to model design and evaluation, applying concepts learned throughout the course.
2. **Transformer Understanding**: Gain in-depth knowledge of how attention-based models function by coding them from scratch rather than using an off-the-shelf library.
3. **QA Performance**: Achieve competitive results (Exact Match (EM) and F1 scores) on SQuAD dataset, .
4. **Modest Success Projections**: A realistic EM/F1 of *around 60–70%* for a first implementation from scratch. With further tuning or partial pre-trained embeddings, performance could improve closer to *70–80%.*

Plan:

Data & Preprocessing

- Dataset: The primary dataset will be SQuAD (Stanford Question Answering Dataset). It is publicly available and contains context paragraphs, questions, and labeled answer spans.
- Parse JSON to extract relevant triplets.
- Tokenize (e.g., subword or byte-pair) and split into train/validation sets.

Framework

- PyTorch was chosen for its flexibility and strong Transformer support.

Network Architecture

- Embeddings: Learn from scratch or use pre-trained (e.g., GloVe).
- Positional Encoding: Sine/cosine is the same as in the original Transformer.
- Transformer Encoder Blocks:
    - Multi-head attention (scaled dot-product).
    - Two-layer feed-forward network (ReLU/GELU).
    - Residual connections and layer normalization.
    - Start with 2–4 layers and scale if resources allow.
- QA Output Layer: A linear head for predicting start and end token positions.

Training

- Loss: Sum of cross-entropies for start/end positions.

- Optimizer: Adam or AdamW with optional scheduling/warmup.
- Batch Size: 8–32, subject to GPU limits.
- Metrics: Exact Match (EM) and F1.