Final Project: Transformer-Based Question Answering System

Yasin Ceran

November 24, 2025

1 Overview

For your final project in this NLP class, you will focus on Transformers by building a hands-on question answering (QA) system that extracts answers from passages and generates explanatory responses. Since we've emphasized Transformers as the state-of-the-art in NLP, this project will let you apply them to real-world tasks, creating a pipeline that demonstrates their power in handling context, attention, extraction, and generation. Everyone will follow the same structure and use the same dataset, but you can experiment with hyperparameters, prompts, or model variants for personalization.

This project is designed to be portfolio-ready—something you'll be proud to share, showcasing how Transformers enable advanced NLP applications like smart search assistants or knowledge retrieval tools. Implement in a Jupyter notebook with code, explanations, visualizations, and a reflective report.

2 Objectives

- Fine-tune Transformers for extractive QA and response generation tasks.
- Evaluate and compare model performance across variants (e.g., BERT for extraction vs. GPT-style for generation).
- Explore Transformer-specific features like attention visualization.
- Build a deployable demo (e.g., Gradio interface) to interact with your models.
- Reflect on why Transformers outperform earlier models, fostering pride in mastering cutting-edge NLP.

3 Dataset

To ensure fair grading and comparison, everyone must use the **SQuAD 2.0** dataset, available via Hugging Face Datasets (`datasets.load_dataset("squad_v2")`). This dataset contains over 100,000 question-answer pairs from Wikipedia passages, including unanswerable questions for realism. It includes 87,000 training and 11,000 validation samples—use these for your splits.

4 Project Structure and Requirements

Use Hugging Face Transformers and PyTorch. Divide your Jupyter notebook into clearly labeled sections with code cells and markdown explanations. The notebook should include:

1. **Data Preparation (15%)**:
    - Load the SQuAD 2.0 dataset and explore it (e.g., print 5 sample questions/passages/answers, compute distribution of answerable vs. unanswerable questions, average passage length).

- Preprocess: Use a Transformer tokenizer (e.g., from BERT or GPT-2) to tokenize questions and passages; handle variable lengths with truncation/padding (max length 384 for passages, 32 for questions).
- For unanswerable questions, ensure proper handling (e.g., no-answer labels).
- Use the provided train/validation splits; subsample if needed for compute limits.
- Visualize: Plot passage length histogram and word cloud for questions.

2. **Extractive QA with Transformers (25%)**:
   - Fine-tune BERT (or DistilBERT) for extractive QA (predict start/end spans in passages, including no-answer detection).
   - Use the Hugging Face `Trainer` API or a custom training loop with appropriate loss.
   - Train for 3-5 epochs; plot training/validation loss and metrics curves.
   - Evaluate on the validation set: Report Exact Match (EM), F1-score, and handling of unanswerable questions.
3. **Response Generation with Transformers (25%)**:
   - Fine-tune a generative model (e.g., GPT-2 or DistilGPT2) to generate explanatory responses based on extracted answers (e.g., prompts like "Explain the answer to [question] from [passage]: [answer]").
   - Prepare data by creating prompt-response pairs (e.g., prompt with question/passage/answer, response as a natural explanation).
   - Train with causal language modeling objective for 3-5 epochs.
   - Generate 5 sample explanations for different QA pairs; evaluate qualitatively (clarity, accuracy) and with perplexity on held-out data.
4. **Advanced Exploration (15%)**:
   - Visualize attention heads from the fine-tuned BERT model on 2-3 sample QA pairs (use bertviz or similar library).
   - Compare zero-shot performance (using pre-trained model without fine-tuning) vs. fine-tuned on a subset of the validation set.
   - Experiment: Try LoRA for parameter-efficient fine-tuning on one model, or compare tiny/base model sizes; report efficiency (training time, parameters) and performance trade-offs.
5. **Integrated Demo (10%)**:
   - Build a simple Gradio or Hugging Face Spaces interface: Input a passage and question, output extracted answer and a generated explanation.
   - Deploy it (free tier on Hugging Face Spaces) and provide a working link in your notebook.
   - Include code to run the demo locally if needed.
6. **Analysis and Reflection (10%)**:
   - Compare models: Create a table summarizing metrics (e.g., EM/F1 for QA, perplexity for generation).
   - Discuss: Advantages of Transformers (e.g., parallelization, long-range dependencies) over RNNs/CNNs; challenges faced (e.g., handling long passages or unanswerable questions).
   - Reflect: Write 1-2 paragraphs on what you learned, any surprises, and why you're proud of the project (e.g., "Building a QA system showed me NLP's role in intelligent search.").

5 Technical Guidelines

- Use Hugging Face Transformers and PyTorch for implementation.
- Handle GPU acceleration if available (`torch.cuda.is_available()`).
- Set random seed to 42 for reproducibility (`torch.manual_seed(42)`).
- Experiment with hyperparameters (e.g., learning rate 2e-5, batch size 16) and document your choices.
- No plagiarism: Implement and explain your code; cite any external resources or code snippets.
- Notebook should be clean, well-commented, and runnable end-to-end.

6 Submission and Grading

- **Due Date**: December 12, 2025. Submit via the class portal as a Jupyter notebook (.ipynb), PDF export, and a link to your deployed demo.
- **Grading Rubric**: Based on completeness (all sections implemented), correctness (models train and evaluate without errors), performance (reasonable metrics, e.g., ¿70 EM for QA), depth of analysis, and creativity (e.g., extra experiments or visualizations).
- The uniform dataset and structure allow for easy comparison and grading, while experiments enable personalization.
- Peer discussions are encouraged, but code must be your own.

This project will equip you with a Transformer-based QA tool—be proud of advancing to the forefront of AI!