GPT-2 Chatbot Streamlit Application

Project Documentation and Technical Report

Project Overview

Objective: Developed an interactive conversational AI application using GPT-2 and Streamlit, enabling real-time text generation and user interaction.

Technical Architecture

- **Framework:** Streamlit
- **Model:** OpenAI's GPT-2
- **Libraries:**
    - Torch
    - Transformers
    - Streamlit

Key Implementation Components

1. Model Loading Strategy

- Utilized `@st.cache_resource` decorator for efficient model caching
- Implemented fallback for padding token to ensure model compatibility
- Selected base GPT-2 model for generative capabilities

2. Response Generation Parameters

Initial Configuration

max_length=150
no_repeat_ngram_size=2
temperature=0.5
top_k=50
top_p=0.95
do_sample=True

Parameter Tuning and Rationale

Temperature (0.5)

- **Purpose:** Controls randomness in response generation
- **Outcome:** Balanced between creativity and coherence
- **Adjustment Impact:** Moderate predictability with slight randomness

Top-K Sampling (50)

- **Purpose:** Limits token selection to top 50 most probable tokens
- **Outcome:** Reduced nonsensical generations
- **Adjustment Impact:** Improved response quality and relevance

Top-P Sampling (0.95)

- **Purpose:** Nucleus sampling to dynamically select token pool
- **Outcome:** More diverse and contextually appropriate responses
- **Adjustment Impact:** Enhanced response variety while maintaining coherence

3. User Interface Design

- Implemented chat-style interface
- Added session state management for conversation history
- Created sidebar with application information
- Error handling for response generation

Performance Characteristics

Strengths

- Real-time AI text generation
- Lightweight and easy-to-deploy application
- Flexible conversation management
- Minimal computational requirements

Limitations

- Base GPT-2 model has limited domain-specific knowledge
- Potential for generating inconsistent or irrelevant responses
- No persistent memory between sessions

Potential Improvements

1. Implement fine-tuning on specific domain datasets
2. Add advanced context management
3. Integrate more sophisticated sampling techniques
4. Implement conversation filtering and safety mechanisms

Technical Challenges Addressed

- Token generation and sampling
- Session state management
- Model loading and caching
- Error handling in generative AI

Deployment Environment

- Recommended environments:
  - Local development
  - Streamlit Cloud

Conclusion

The GPT-2 Chatbot Streamlit application demonstrates a practical implementation of generative AI with an intuitive user interface, showcasing the potential of transformer-based models in conversational applications.
