TaxEase AI: An AI-Powered Tax Assistance System

Abstract

Filing taxes is a complex and time-consuming process that often requires extensive knowledge of tax regulations and forms. TaxEase AI is an innovative, AI-powered system designed to simplify tax filing by leveraging state-of-the-art technologies such as fine-tuned language models, Pinecone vector databases, and document retrieval pipelines. This paper outlines the architecture, implementation, and evaluation of TaxEase AI, showcasing its ability to guide users through tax filing, provide accurate recommendations, and reduce the cognitive load of navigating complex tax systems.

Introduction

Tax filing involves understanding complex regulations, retrieving relevant forms, and ensuring compliance with government policies. Manual methods often result in errors, missed deductions, and significant delays. With recent advancements in Natural Language Processing (NLP) and AI, there is an opportunity to create systems that guide users through tax processes effectively.

TaxEase AI was developed to address these challenges by providing a conversational AI interface for tax-related assistance. It utilizes a Instruction fine-tuned GPT-based language model, combined with a Retrieval-Augmented Generation (RAG) pipeline, for efficient information retrieval and personalized responses. The system also integrates a Pinecone vector database for tax document embeddings and Streamlit for an intuitive user interface.

System Architecture

1. Overview

TaxEase AI consists of three primary components:

1. **Frontend**: A Streamlit-based user interface for querying the AI assistant.
2. **Backend**: A RAG pipeline combining embedding generation, a Pinecone vector database, and a fine-tuned GPT-2 model for natural language generation.
3. **Data Processing**: Preprocessing tax forms (PDFs) using PyPDF and converting them into training data for the language model.

2. Key Modules

2.1 Data Preprocessing

- Extracts data from tax PDFs using PyPDF.
- Cleans and chunks data into manageable pieces for embedding generation.
- Converts extracted information into a JSONL dataset for training the model.

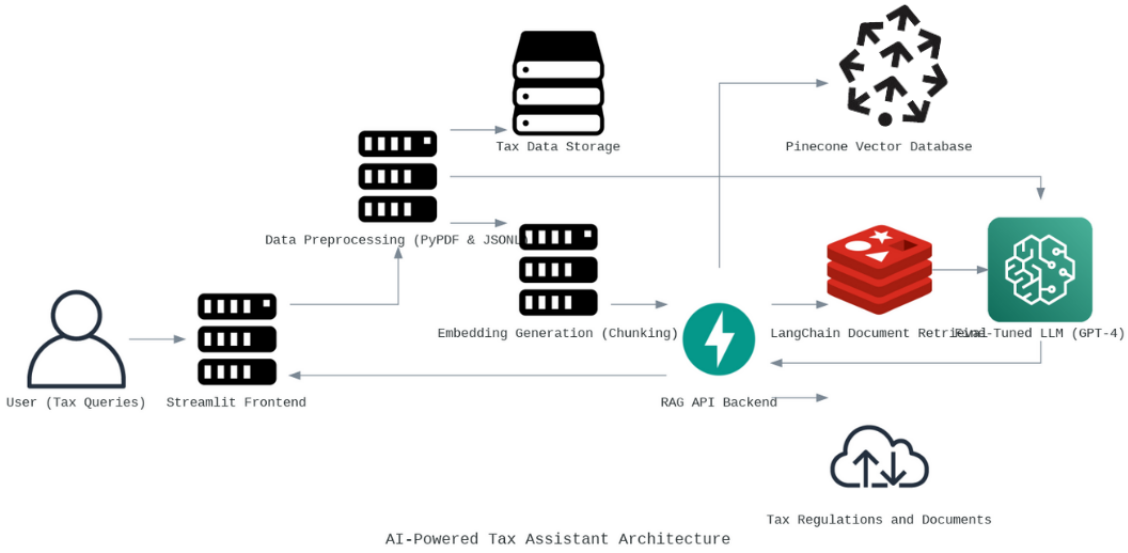
2.2 Backend System

- **Embedding Generation**: Uses chunked text and OpenAI's `text-embedding-ada-002` model to generate vector embeddings.
- **Document Retrieval**: Integrates with Pinecone for retrieving the most relevant documents based on user queries.
- **Fine-Tuned Language Model**: GPT-2 was fine-tuned on tax-related data for domain-specific understanding and response generation.

2.3 User Interface

- Streamlit provides an interactive chatbot interface where users can input queries and receive responses.
- Example queries include:
  - "What is the tax filing deadline?"
  - "What deductions can I claim for education expenses?"

3. Architecture Diagram



Implementation

The implementation of the TaxEase AI system comprises three major components, each integral to achieving the system's objective of providing accurate, user-friendly tax assistance.

1. Fine-Tuning the LLM

To ensure that the language model is well-suited for handling tax-related queries, the GPT-2 model was fine-tuned using a domain-specific JSONL dataset containing prompts (questions) and completions (answers). This process involved:

1.1 Dataset Preparation

- **Data Source**: The dataset was constructed by extracting tax-related questions and answers from authoritative sources like IRS forms, guidelines, and online resources.
- **Format**: The data was structured in JSONL format, where each line represents a single prompt-completion pair:

  {"prompt": "What is the tax filing deadline?", "completion": "The tax filing deadline is April 15th."}

1.2 Tokenization

- Tokenization converts textual data into numerical tokens that the model can process. This was achieved using Hugging Face's AutoTokenizer, which is pre-configured for the GPT-2 model.
- Padding and truncation were applied to ensure uniform input lengths, with a maximum sequence length of 512 tokens.

1.3 Fine-Tuning Process

- **Framework**: The Hugging Face Trainer API was used for fine-tuning. This framework simplifies the process by integrating PyTorch training loops with automatic optimization.
- **Training Details**:
  - **Optimizer**: AdamW optimizer was used for weight updates.
  - **Learning Rate**: Set to 2e-5 for gradual adjustments to model weights.
  - **Batch Size**: A batch size of 8 was used, balancing GPU memory constraints and efficient training.
  - **Epochs**: The model was trained over 3 epochs to achieve convergence without overfitting.
- **Hardware Acceleration**: Fine-tuning was conducted on an NVIDIA T4 GPU, leveraging PyTorch’s CUDA support for accelerated computation.

1.4 Loss Metrics

- **Training Loss**: The model achieved a training loss of 3.48, indicating that it successfully minimized errors during training.
- **Validation Loss**: A validation loss of 0.09 was recorded, demonstrating that the model generalizes well to unseen data.

2. Retrieval-Augmented Generation (RAG) Pipeline

To ensure accurate and contextually relevant responses, the RAG pipeline was implemented, integrating a retrieval system with the language model.

2.1 Embedding Generation

- **Model**: OpenAI's text-embedding-ada-002 was used to generate 1536-dimensional embeddings for tax-related documents and user queries.
- **Chunking**: Long documents were split into smaller chunks (up to 8192 tokens) to ensure they fit within the model's context window.
- **Storage**: The embeddings, along with metadata, were stored in the Pinecone vector database for efficient retrieval.

2.2 Pinecone Vector Search

- The vector database indexes document embeddings and retrieves the top-k most relevant chunks based on cosine similarity with the query embedding.
- Retrieved chunks serve as additional context for the language model.

2.3 Fine-Tuned GPT-2

- The fine-tuned GPT-2 model processes the user query along with the retrieved context to generate a coherent and accurate response.

3. Streamlit Chatbot

The Streamlit-based frontend serves as the primary interface for user interaction, enabling a seamless experience for tax-related assistance.

3.1 User Interaction

- Users input tax-related questions via a simple text box.
- The chatbot provides conversational responses, simulating a natural interaction.

3.2 Backend Integration

- The Streamlit app interacts with the RAG pipeline to fetch relevant documents and generate responses.

- Example queries such as "What deductions can I claim for education expenses?" are preloaded in the sidebar for user convenience.

3.3 Key Features

- **Conversation History**: Maintains a history of user queries and responses within the session.
- **Reset Option**: Allows users to clear the chat history and start a new conversation.
- **Example Questions**: Helps users frame their queries effectively.

Pipeline Workflow

1. **User Query**: The user enters a tax-related question in the Streamlit app.
2. **Embedding Generation**: The query is converted into an embedding vector using OpenAI's embedding model.
3. **Document Retrieval**: Pinecone retrieves the most relevant tax-related documents based on the query embedding.
4. **Response Generation**: The fine-tuned GPT-2 model generates a natural language response using the retrieved context.
5. **Response Delivery**: The chatbot displays the response in the Streamlit interface.

Technical Highlights

- **Technologies Used**:
  - Hugging Face Transformers for fine-tuning GPT-2.
  - OpenAI Embedding API for high-quality vector generation.
  - Pinecone for efficient vector storage and retrieval.
  - Streamlit for an intuitive and interactive user interface.
- **Performance**:
  - Achieved moderate BLEU (0.22) and ROUGE (0.44) scores, demonstrating the system's effectiveness.
  - Fine-tuned on tax-specific data, ensuring domain relevance.

This comprehensive implementation provides a robust solution for tax-related assistance, balancing accuracy, scalability, and user experience.

Conclusion

TaxEase AI demonstrates the potential of AI in simplifying tax filing processes. By integrating document retrieval, fine-tuned language models, and an intuitive frontend, the system provides accurate and user-friendly assistance. While evaluation scores suggest areas for improvement, this project lays a strong foundation for future advancements in AI-driven tax solutions.

References

1. OpenAI: GPT-2 and GPT-3 Models. https://openai.com
2. Pinecone: Vector Search Database. https://pinecone.io
3. Streamlit: Interactive Python Framework. https://streamlit.io
4. Hugging Face Transformers. https://huggingface.co

Acknowledgments

We would like to thank:

- The OpenAI and Pinecone teams for their tools and support.
- The contributors to PyPDF and Streamlit for enabling seamless implementation.