Project Report: Anime Recommender System

# Overview

This project aims to create an anime recommendation system using NLP techniques, fine-tuning embedding models, and leveraging open-source large language models (LLMs). The system uses user queries to recommend anime based on an anime synopsis and genre dataset, providing personalized suggestions.

## Goals and Objectives

The main goal of this project was to develop an efficient anime recommendation system by using:

1. Fine-tuned embeddings to capture relevant context from anime descriptions.
2. A local LLM to generate coherent and relevant responses.
3. A retrieval-based QA approach to retrieve the most appropriate anime recommendations based on user queries.

This report presents the implementation steps and explains the techniques employed.

# Environment Setup

**Cell 1** - Setup and Installations

In this step, we set up the environment by disabling logging from Weights & Biases (WANDB) and silencing warnings that may clutter the output.

We also installed the necessary Python libraries, such as:

1. **Transformers** and **Sentence Transformers** for pre-trained models.
2. **LangChain** and **LangChain-HuggingFace** for efficient chaining of LLMs.
3. **ChromaDB** for managing vector stores and **datasets** to manage data.

A screen shot of a computer

Description automatically generated**Explanation**:

1. **Weights & Biases (WANDB)** is a tool often used for tracking experiments. Since it's not required for this implementation, we disabled it.
2. **Transformers** and **Sentence Transformers** are used for model fine-tuning and text embeddings. **LangChain** is a robust framework for connecting LLMs with various downstream tasks.

# Data Preparation

**Cell 2 & Cell 3** - Loading Libraries and Importing Data

A screenshot of a computer program

Description automatically generated

We loaded the necessary libraries and the dataset containing anime titles, synopses, and genres.

**Cell 4 & Cell 5** - Preprocessing the Data

A screen shot of a computer program

Description automatically generated**Explanation**:

1. We corrected a typo in the data and removed entries with incomplete or uninformative descriptions.
2. We combined the title, synopsis, and genres into a unified "combined\_info" field, which provides a richer context to the model for embedding generation.

# Fine-Tuning Embedding Model

**Cell 8** - Fine-Tuning the Embedding Model

A computer screen shot of a program

Description automatically generated

**Explanation**:

1. **SentenceTransformer** is used to load a pre-trained embedding model, which we fine-tuned with the SimCSE (Simple Contrastive Learning of Sentence Embeddings) approach.
2. **Fine-tuning** ensures that the embeddings generated are more suited to the anime context, improving recommendation accuracy.
3. **MultipleNegativesRankingLoss** was used to train the model to generate better contextual embeddings, minimizing the loss for relevant recommendations.

# Creating Embeddings and Vector Store

**Cell 9** - Creating a Vector Store for Retrieval

A computer screen shot of a program code

Description automatically generated**Explanation**:

1. We used **HuggingFaceEmbeddings** with the fine-tuned model to generate embeddings for each dataset chunk.
2. **CharacterTextSplitter** splits documents into manageable chunks to facilitate efficient vector searches.
3. **Chroma** is a vector store that enables fast retrieval based on cosine similarity between the query and anime embeddings.

# LLM and Retrieval Chain Setup

**Cell 10** - Defining the Language Model

A computer screen shot of a program

Description automatically generated**Explanation**:

1. We used **Flan-T5**, an open-source LLM, which we loaded using HuggingFace's AutoTokenizer and AutoModelForSeq2SeqLM.
2. The **pipeline** configuration controls the generation characteristics, such as **temperature** (controls randomness), **top\_p** (sampling diversity), and **repetition\_penalty** (to avoid repetitive responses).
3. The model was wrapped in a LangChain LLM wrapper for our retrieval-based QA chain.

# RetrievalQA Chain Setup

**Cell 11** - Setting Up RetrievalQA Chain

A computer screen with text on it

Description automatically generated**Explanation**:

1. **PromptTemplate** defines how queries are formatted before being sent to the LLM.
2. The **retriever** retrieves the most relevant documents from the vector store (k=10 implies we retrieve 10 documents for each query).
3. **RetrievalQA** uses these retrieved documents as context for generating coherent anime recommendations.

# Testing the Model and Personalized Prompts

**Cells 12-19** - Querying the System

A screenshot of a computer program

Description automatically generatedWe used different user queries to evaluate the recommender system's output. For personalization, additional user information (e.g., age and gender) was included in the prompts to generate more customized recommendations.

A screenshot of a computer program

Description automatically generated

**Explanation**:

1. We iterated through multiple user queries and obtained relevant anime recommendations, verifying that the model can have diverse and meaningful suggestions.

**Summary**: The project integrates several techniques, from embedding-based document retrieval to leveraging an LLM for coherent responses. It uses open-source models and state-of-the-art techniques to generate contextually relevant anime recommendations.