📝

Proposal

Problem Statement and Task Definition

Our system is a chatbot which can answer questions based on one or more user provided files.

Input:

1. A directory containing several PDF or plain text files.
2. Questions you want to ask.

Output:

1. Summarize the content of a file.
2. Answer specific questions based on all the files provided.
3. Address the source/referenced file of the answer.
4. If there are different answers for a question from different files, analyze and elaborate on the conflict.

When searching for information of a certain topic. The result is often overwhelming and requires a lot of time to digest and understand. And sometimes there might be information from different sources telling a different story. Our project aims to assist the process. You can simply download or copy all the reference files and ask your questions. Our chatbot summarizes its answer and provide of which files it is from if you want to dive further into.

Description of the challenges

challenges:

1. PDF parsing
2. How to understand natural language
3. contextual understanding
4. answer generation

topics to address those challenges:

1. developing tool such as Python PyMuPDF (FitZ) or PDFMiner packages
2. Using NLU algorithm to enhance how machines comprehend context, sentiment, and the nuances of language for applications like chatbots, voice assistants, and content analysis.
3. Using NLP to interpret the meaning of words, phrases, or sentences within their surrounding textual environment.
4. Using NLP to sequence-to-sequence models or transformers, to understand a query and produce a relevant, coherent response based on a given context or knowledge base.

Input/Output Behavior with Concrete Examples

Input 1: Provide a single PDF file. Type "What should I do?" into the chatbot interface.
Output 1:

```
Here are the simple bullet points of what you should do for the assignment:

Task A: Traditional Machine Learning Models

Part 1: Load and prepare the dataset
    - Implement a function to load images and labels from a dataset folder.
    - Convert images to tuples of numpy arrays and labels.
Part 2: Build and train models
    - Use scikit-learn to build and train KNN, Random Forest, and AdaBoost classifiers.
    - Test the models and calculate performance metrics like accuracy, precision, and recall.
```

```
Part 3: Perform additional experiments
    - Change hyperparameters of the classifiers and compare performance.
    - Find the best performing classifier and its optimal hyperparameters.
Part 4: Detect car
    - Implement a function to detect cars in images from the DETECT folder.
    - Load video frames and parking space coordinates to detect cars using the classifier.
Part 5: Draw line graphs
    - Plot graphs of parking slots occupation and accuracy of the models.

Task B: Deep Learning Model with YoloV7

Part 1: Apply pre-trained model to image
    - Load a pre-trained YoloV7 model and apply it to a .png image to detect objects.
Part 2: Fine-tune YoloV7 model
    - Input training instructions for YoloV7 and aim for 90% accuracy.
    - Provide a screenshot of the accuracy and training curve.
Part 3: Try other state-of-the-art methods (bonus)
    - Implement other state-of-the-art methods for vehicle detection.
    - Compare them with Adaboost and discuss findings.

Report

- Write a report including code screenshots, explanations, graphs, and discussions on different
- Describe challenges faced and how you overcame them.

Submission

- Submit a zip file containing Python code files and the report PDF.
- Follow the naming and format requirements for submission.
```

Input 2:
Provide multiple files. Type "狗的學名是甚麼" into the chatbot interface.

Output 2:

```
狗的學名是Canis lupus familiaris
來源: dog.txt
```

Related works

Use GPT-4 & LangChain to create a ChatGPT Chatbot for your PDF Files: link

Methodology

1. Use PyMuPDF to extract data from PDF files. PyMuPDF provides easy and efficient text extraction and PDF manipulations, but it may require a learning curve due to its complex API and limited documentation for advanced use case.
2. Use RAG architecture to embed the data into LLM. This enhance the question-answering process by improving relevance, efficiency in information extraction, contextual understanding, combination of retrieval and generation method, etc.
3. Fraction multiple or large PDF files into fragments and store in a 15vector database.
4. Use Langchain to implement RAG. This enhances natural language understanding, semantic reasoning, integration with generation models, etc.

Evaluation Metrics

Establish several test cases, create several inputs along with predefined outputs. Embed both the predefined outputs and the actual test outputs into vectors, then perform similarity comparison to set it as precision.

Baselines

Baseline:

1. For one given PDF and the questions, give the required answer.
2. Summarize the PDF.

Oracle:

1. The bot should be able to collect the answer from multiple given PDF.
2. The bot should answer the question while providing the way or where it find those answers in the PDF.
3. It should answer the question with higher accuracy.
4. summarize the PDF based on more customized requests.

Work Plan

Time schedule

- **Project Planning & Research**
  - Setting our project goals and objectives
  - Review existing LLM applications and case studies
- **Learning and Initial Development**
  - Learning RAG
  - Learning LangChain framework to implement RAG
  - Learning OpenAI API
- **Application Development**
  - Design frontend user interface
  - Design backend that take users' prompt and generate a specific answer based on the PDF provided
- **Documentation and Deployment**
  - Testing different LLM models and comparing with each other
  - Deploy the service online

Discussion

We decided to open our discussion channel on **Notion**, which is very helpful to do progress tracking and schedule our tasks.

🤖 AI Final Project - Team 10

Repo

https://github.com/ChuEating1005/Homework-Helper.git