Project TLDR: Standalone Desktop application for Question-Answering and Summary using resource efficient LLMs

Manu Hegde

A Capstone Project Proposal
submitted in partial fulfillment of the
requirements of the degree of

Master of Science in Computer Science & Software Engineering

University of Washington
19 July 2024

Project Committee:
Prof. Erika Parsons, Committee Chair
Prof. Michael Stiber, Committee Member
Prof. Shane Steinert-Threlkeld, Committee Member

I. Introduction

This document outlines the proposal to develop a standalone desktop application that enables ChatGPT like Question-Answering and Summarization on top of a corpus of documents on user's device. The application shall embody a resource efficient implementation of a chosen Large Language Model (LLM), targeting Apple's M1/M2 hardware platform.

II. Project Goals and Vision

A. Goals of Project

The primary goal of the project is to develop a standalone desktop application that enables Question and Answering, Summarization on top of a repository of documents. Additionally, the resource usage of this application should be limited and predictable to allow for comfortable multi-tasking on the user's device. Furthermore, the goal also includes the ability to re-phrase large and complex text to present information in a simplified, appealing and engaging manner that captures the reader's attention, by leveraging storytelling and narrating techniques that are already known and understood.

B. Identification of Problem

There is a significant need for tools that can summarize and interpret academic materials due to their often complex and dense nature. Furthermore, there is a need to handle numerous sets of documents in a single context for holistic understanding of a subject or topic. While it is possible to achieve these results currently, it requires repeated and significant manual interventions with tools like ChatGPT, especially when more than one document is involved. Furthermore, in case of online tools like ChatGPT, Gemini or Claude. of all relevant pieces of information needs to be shared with the 3rd party. Instances of leakage of data shared with ChatGPT [1,2] raise concerns of data safety and confidentiality. Also, in case of research, including information from unknown sources can also impact the integrity of the information obtained.

While desktop-based solutions like Ollama and LLamaFile do exist, their goals are intended towards enabling large number of open-source models to run on diverse set of hardware. This narrows their base of users to only those skilled technically. Furthermore, it limits their scope to optimize for performance as well as streamline usability for specific use cases. Hence, an application that could enable Question-Answering and Summarization on top of a set of documents like academic papers in Zotero collection, Google Drive, folders on the computer or even in Notes (like Apple Notes) could serve to be very useful and beneficial to students and researchers.

D. Stakeholders and Beneficiaries

The applications main beneficiaries will be students and researchers whose work involves reading and understanding numerous papers and books while avoiding information from unattributable sources from the internet.

III. Criteria

A. Levels of Success

1. Minimum: The application can perform summarization, question and answering on content from a collection of documents like academic papers and books on user's device, without requiring internet access.
2. Expected: The application delivers context-specific information while utilizing not more than 50% of system's resources on a device like M1 MacBook Air.

3. Aspirational: Ability to take multi-modal input and ability to rephrase the content in very engaging and interesting manner, following the examples of books like HeadFirst Java that has engaging story telling for complex technical topics.

B. Quality and Associated Measurement Metrics

Quality metrics include functional & qualitative, as well as non-functional metrics. While the functional & qualitative metrics determine the quality of results, non-functional metrics help ascertain performance from resource consumption and speed of execution of the software while achieving the functional goals.

Functional Metrics:

1. Text Corpus Size in GB: The amount of text that can be handled by the implemented LLM to successfully deliver relevant results
2. BertScore [5] comparison against ChatGPT: Score evaluating the closeness of the application's output with that of ChatGPT when provided with the same input. Range of Bertscore is [-1,1] with -1 being extremely dissimilar and 1 being identical.
3. AI feedback ratings: To measure on a scale of 1 to 10, the relevance of the application's output as measured by another AI (such as ChatGPT, Gemini, Claude)
4. User feedback ratings: To measure on a scale of 1 to 10 the satisfaction of output from a user's perspective.

Non-Functional Metrics: Non-functional metrics are critical to determine the usability of the application for in the expected target environment.

1. Resource usage metrics:
   - Main Memory (RAM) used (in GB)
   - Storage used by application (in GB)
   - CPU usage (in percentage x clock rate x time)
   - GPU Usage (in percentage x clock rate x time; for Apple Neural Engine)
   - Data throughput (in MB): Amount of input text passed as part of RAG to yield a given result
2. User interaction metrics:
   i. Turnaround Time (in Seconds): Total Time from user input to the start of output generation
   ii. Total Response Time (in Seconds): Total Time from user input to end of output generation

IV. Positioning of the project

This project aims to fill the gap in the current set of tools for researching information by leveraging the latest advancements in Natural Language Processing. The value proposition of this project can be understood by understanding the current landscape of tools for such use cases.

Current existing solutions:

1. Web-based/Cloud server-based solutions like ChatGPT and Gemini:
   <u>Advantages:</u>
   a) They offer fastest, most up-to-date and most accurate models that have vast amount of knowledge gathered from all text available on the internet and beyond
   b) They offer multi-modal inputs i.e. text as well as images, enabling a holistic understanding of input document especially when containing diagrams and visuals
   <u>Disadvantages:</u>
   a) They offer at most single document-based question answering.

b) Even so, having been trained with vast amount of data from internet, they may still result in unreliable information or replicate copyrighted content without attribution.
c) Data being shared as input to these models can be stored and used by the solution provider, for proprietary purposes.
d) Outputs generated by the model could also have partial license whereby the solution provider may still retain their right to store and use the model's output for feedback purposes and thereby cause confidentiality and false positive plagiarism issues.

2. Web based document Q&A solutions like chatpdf.com
   Advantages: These solutions offer mostly single document Q&A which is fast and accurate.
   Disadvantages: They use models like ChatGPT or Gemini to power their use cases and hence continue to retain their disadvantages as well.

3. Desktop based solutions like Ollama and LLamaFile
   Advantages: Contains implementation of large number of open-source models for all major types of hardware and all major operating systems.
   Disadvantages:
   a) They focus on accessibility of models on variety of hardware and do not optimize for any particular use case or optimize performance for any specific hardware.
   b) They do not provide ability to perform Q&A over multiple documents
   c) They are intended for technically savvy users and require understanding of the various models and their nuances.

Value proposition:
Build an application that can process multi-modal data (text and visual input) from a collection of documents and perform Question and answering on them; Enabled by a single LLM (Large Language Model) implementation for single hardware platform (Apple M1/M2) and operating system (MacOS). This project aims to borrow some common implementational learnings for Large Language Models through projects like Ollama and LLamaFile and their common backend library, The *llama.cpp* project. This project aims to leverage fundamental concepts in hardware performance optimizations like CPU pinning, cache line optimizations from domains such as High-performance computing and Parallel Programming in Grid & Cloud.

V. Project Plan

Autumn 2024	
Week 1	Shortlist 10 open-source models whose code and weights are available
Week 2-4	a) Quantize or obtain quantized FP16 and Int8 weights for each model considered. Perform any additional model size reduction techniques as appropriate for the model b) Evaluate and pick one model based on resource usage and output quality tradeoffs
Week 5-8	Implement the chosen Language Model for inference, using Apple CoreML API to take advantage of ANE (Apple Neural Engine)
Week 9-11	Implement Retrieval Augmented Generation workflow, optimizing for ANE as well as M1 CPU
Winter 2025	

Week 1-4	Implement the required graphical user interface for the desktop application
Week 5-8	Build features to integrate with services such as Apple notes, Zotero and Google Drive to fetch and process documents in multimodal fashion
Week 8-11	Perform testing and bug fixing to ensure robustness and usability of the application
Spring 2025	
Week 1-2	Perform experiments to obtain the pre-prompting required for simplified rephrasing of complex topics
Week 3-4	Collect user feedback and perform supervised finetuning, on the original unquantized model weights and re-quantize
Week 5-6	Collate results and prepare for presentation and defense
Week 7-9	Final project defense

VI. Constraints, Risks, Resources

A. Key Constraints

1. Technical Constraints
    a) LLM size in memory: Resource constraints on the target device. Even a small LLM can take many hundreds of MBs or even GBs of space and hence reduction of this resource constraint is a key aspect for the project.
    b) Quality of output: Although the major model size reduction techniques result only in marginal reduction in output quality, when multiple such techniques are applied, they may result in significant loss of output quality and hence the tradeoffs need to be considered cautiously.
2. Implementational Constraints:
    a) Time and complexity of project: Since this project forays into a highly evolving field of Technology i.e. Natural Language Processing, the techniques and methodologies considered may not be completely mature and could have undiscovered side effects causing additional complexity. Hence time and effort estimations could be inaccurate.
    b) Reliable Performance measurement: Measuring and quantifying performance in a multi-tasking environment with a graphical user interface could be challenging and may require order of magnitude more measurements to reliably estimate performance. Furthermore, replicating the workflow with online models like ChatGPT to yield output quality comparisons could require additional effort than estimated.

B. Resources Needed for Success

1. Reliable user hardware for development and testing
2. Open-source licensed Large Language Models with their code and weights
3. Compute and storage to perform quantization and other model size reduction techniques
4. Documentation on workings of Apple Neural Engine and the CoreML API
5. Access to high quality academic materials for testing and refinement
6. Access to online models like ChatGPT for output quality comparison and feedback

C. Anticipated Risks

1. Technical challenges in model optimization – Optimizing the model for specific resource constraints can be challenging and further risks limiting the performance and quality
2. Performance optimization challenges: Since the RAG workflow involves number of steps including parsing and storing significant amount of multimodal data, there may be challenges in

leveraging the acceleration hardware and result in portions of workflow that are cpu-only and potentially bottlenecked.

VII. Scope and Design

A. Application Workflow Design

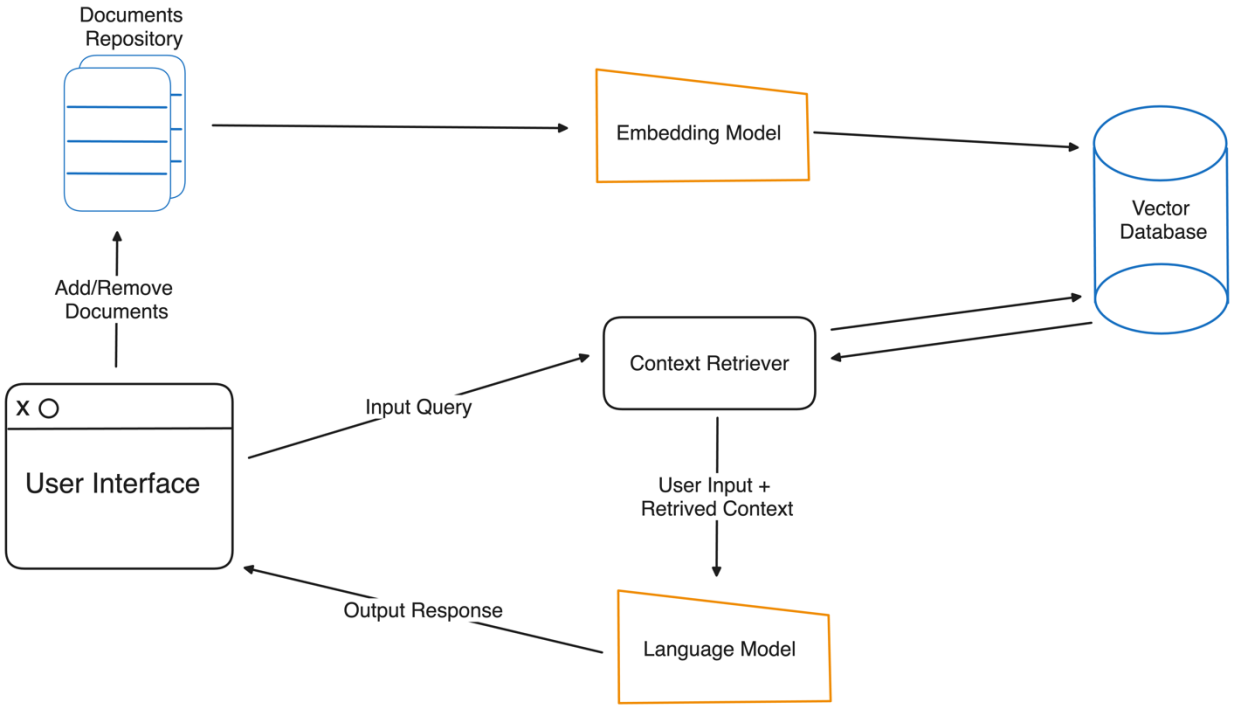

Figure 1: Workflow of TLDR desktop application

The application workflow is as follows

a) Workflow 1:
   1. User selects documents to add to the Document Library
   2. Documents are tokenized and converted to embeddings using the embedding model for the given LM (language model)
   3. The obtained embeddings are stored in the vector database

b) Workflow 2:
   1. User provides input query
   2. Vector DB performs a scan to obtain all portions of text that are contextually relevant to the user's query
   3. LLM obtains the relevant context from vector database along with user input and generates output text
   4. Output is displayed to the user via Graphical User Interface

B. Application Design and Scope

Technical Design:

- The application will be designed and developed for MacOS.

- Architecture for development and deployment is arm-v8 architecture of M1/M2 chips.
- Application will be developed using Swift and C-Sharp Languages for both User Interface as well as for the LLM implementation.
- Bash scripting, Python and C++ programming languages could also be involved for additional functionalities.

VIII. References

1. Mattern et al. (2023) - Membership Inference Attacks against Language Models via Neighbourhood Comparison
2. Nasr et al. (2023) - Scalable Extraction of Training Data from (Production) Language Models
3. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference, Jacob et. al, 2017
4. LoRA - Low Rank Adaptations of Large Language Models, Hu et. Al, 2021
5. Zhang et al. (2020) - BERTScore: Evaluating Text Generation with BERT

IX. Appendix

1. TLDR – Abbreviation for 'Too Long Didn't Read' a common term in online forums like Reddit to denote a summary or gist of a lengthy text
2. Model Inference - Obtaining predictions or output from a machine learning model after the completion of its training phase
3. Large Language Model (LLM) - family of Language models like Transformers (Vaswani et. al, 2017) that have hundreds of millions or billions on parameters
4. Prompt - Instructions given by the user to a Large Language Model specifying the task to be performed
5. Pre-Prompting - Providing instructions to a Large Language Model that dictates and guides model's behavior while responding to a user's prompt. This is especially done for prevention of abuse and misinformation
6. Multimodal Language Model - A Language model that can take multiple types of input, generally text and images
7. Retrieval Augmented Generation (RAG) - Generating output from a language model by providing additional contextual data from a database, based on the given input query/prompt