最终项目作业

发布：第9周（11月6日）截止：
第13周（12月7日）小组人数：
1-3名学生

1. 引言

这个最终毕业设计项目是您本学期工作的结晶。它旨在综合您在我们课程所有三个主要部分中培养的技能：

- **第一部分：Python基础：** 您将运用数据结构、函数、面向对象编程和错误处理的知识来构建一个鲁棒的、可复现的代码库。
- **第二部分：用于机器学习的Python：** 您将应用您对数据管道（NumPy, Pandas）、可视化（Matplotlib）、模型评估和验证的理解。
- **第三部分：用于深度学习和大型语言模型的Python：** 您将直接接触第9-12周的概念，使用PyTorch、Transformers、微调和AI智能体模式。

该项目具有挑战性，但旨在实现。它将为您的作品集提供一个重要的、与行业相关的作品。您的目标是选择**以下三个项目选项之一**并交付一个完整、文档齐全且功能性的解决方案。

2. 项目选项

请仔细审阅这三个选项。每个项目都包括**核心目标**（构成项目的基础）和**高级方向**（为获得更高分数而探索的机会）。

项目A：微调小型语言模型以完成实际任务

主题： 对紧凑模型进行参数高效微调（PEFT），以完成真实世界任务（分类或生成）。

为什么有效： 该项目让您具体接触到 PyTorch、Transformers 库、数据管道、训练循环、评估指标和过拟合控制。它以适度的计算要求实现高影响力。

🌟 现实世界动机

想象一下您在一家快速发展的初创公司工作。客户支持团队的电子邮件应接不暇。您的目标是建立一个能帮助他们的模型。

- **分类路径:** 构建一个分类器，自动将收到的电子邮件标记为错误报告、功能请求、计费问题或一般咨询，以便将其路由给合适的人员。

Final Project Assignment

Issued: Week 9 (November 6th)
Due: Week 13 (December 7th)
Group Size: 1-3 students

1. Introduction

This final capstone project is the culmination of your work this semester. It is designed to synthesize the skills you have developed across all three major parts of our course:

- **Part 1: Python Fundamentals:** You will use your knowledge of data structures, functions, OOP, and error handling to build a robust, reproducible codebase.
- **Part 2: Python for Machine Learning:** You will apply your understanding of data pipelines (NumPy, Pandas), visualization (Matplotlib), model evaluation, and validation.
- **Part 3: Python for Deep Learning and LLMs:** You will directly engage with the concepts from Weeks 9-12, working with PyTorch, Transformers, fine-tuning, and AI agentic patterns.

The project is challenging but designed to be achievable. It will provide you with a significant, industry-relevant piece for your portfolio. Your goal is to choose one of the three project options below and deliver a complete, well-documented, and functional solution.

2. Project Options

Please review the three options carefully. Each project includes **Core Objectives** (which form the baseline for the project) and **Advanced Directions** (opportunities to explore for a higher grade).

Project A: Fine-Tune a Small Language Model for a Practical Task

Theme: Parameter-efficient fine-tuning (PEFT) on a compact model for a real-world task (classification or generation).

Why this works: This project provides concrete exposure to PyTorch, the Transformers library, data pipelines, training loops, evaluation metrics, and overfitting control. It delivers high impact with modest compute requirements.

🌟 Real-World Motivation

Imagine you work at a fast-growing startup. The customer support team is overwhelmed with emails. Your goal is to build a model that helps them.

- **Classification Path:** Build a classifier that automatically tags incoming emails as Bug Report, Feature Request, Billing Issue, or General Inquiry so they can be routed to the right person.

- **生成路径:** 构建一个生成模型，起草简短、礼貌、专业的“谢谢”或“我们正在处理”的常见问题回复。

建议模型（根据硬件选择一个）：

- **CPU / Colab（免费层级）：** distilbert-base-uncased（文本分类）、distilgpt2（短文本生成）
- **单GPU (T4/V100)：** TinyLlama 或 GPT2-small (使用 LoRA/QLoRA)

候选数据集（小型、真实且安全）：

- **分类：** SST-2（情感）、AG News（主题）、IMDb。或者在线寻找一个小型“客户支持意图”数据集。
- **生成：** DailyDialog、产品评论摘要。
- 如果原始数据集过大，您可以对**子集**进行抽样。
- **您的选择：**您可以查找并提出自己的数据集（例如，来自Kaggle、公共API或另一门课程），只要它具有可管理的大小并适合适度的计算预算即可。

💡 提示与入门

- 我们建议使用Hugging Face数据集库以便于数据加载和标记化。

- 首先，让一个基线模型（例如，distilbert-base-uncased）通过Trainer API运行一个简单的分类任务。
- 一旦基线模型工作正常，就可以使用PEFT库集成PEFT（如LoRA）。这通常只需几行代码即可修改您的模型配置。
- 务必密切关注你的数据收集器，尤其是针对生成任务，确保填充和掩码得到正确处理。

🎯 核心目标

你的目标是展示一个完整的微调工作流。这里的成功意味着：

1. **构建一个完整的管道：** 这包括加载和清理数据集、标记化、训练/验证分割、PEFT (LoRA) 微调，以及实现早期停止和基本检查点等机制。

2. **报告标准指标：**
   - **对于分类：** 你应该报告准确率/F1分数，并考虑置信区间（通过自举法或多重随机种子）来展示你的结果的稳定性。
   - **对于生成任务：**您应该报告ROUGE-L分数。为了捕捉定性差异，您必须通过实现一个轻量级的“LLM作为评判者”评估器来补充这一点。您将使用一个独立的大语言模型（例如，GPT-3.5或其他API）根据您设计的明确的评分标准，对您的模型生成输出的质量（例如，礼貌性、相关性）（至少20个样本）进行评分。

3. **分析过拟合控制：** 你应该尝试和比较不同的超参数，例如学习率扫描或权重衰减，并使用Matplotlib损失/指标曲线可视化其影响。

- **Generation Path:** Build a generative model that drafts short, polite, professional "thank you" or "we're looking into it" replies to common questions.

Suggested models (choose one based on hardware):

- **CPU / Colab (free tier):** distilbert-base-uncased (text classification), distilgpt2 (short-form generation)
- **Single GPU (T4/V100):** TinyLlama or GPT2-small with LoRA/QLoRA

Candidate datasets (small, realistic, and safe):

- **Classification:** SST-2 (sentiment), AG News (topic), IMDb. Or find a small "customer support intent" dataset online.
- **Generation:** DailyDialog, product review summarization.
- You can sample a **subset** if the original dataset is too large.
- **Your Choice:** You may find and propose your own dataset (e.g., from Kaggle, a public API, or another course), as long as it is of a manageable size and suitable for the modest compute budget.

💡 Hints & Getting Started

- We recommend using the Hugging Face datasets library for easy data loading and tokenization.
- Start by getting a baseline model (e.g., distilbert-base-uncased) running with the Trainer API for a simple classification task.
- Once the baseline works, focus on integrating PEFT (like LoRA) using the peft library. This often involves just a few lines of code to modify your model config.
- Pay close attention to your data collator, especially for generation tasks, to ensure padding and masking are handled correctly.

🎯 Core Objectives

Your goal is to demonstrate a full fine-tuning workflow. Success here means:

1. **Building a full pipeline:** This includes loading and cleaning the dataset, tokenization, train/val splits, PEFT (LoRA) fine-tuning, and implementing mechanisms like early stopping and basic checkpointing.
2. **Reporting standard metrics:**
   - **For Classification:** You should report Accuracy/F1 and consider confidence intervals (via bootstrap or multiple seeds) to show the stability of your results.
   - **For Generation:** You should report ROUGE-L scores. To capture qualitative differences, you must supplement this by implementing a lightweight "LLM-as-Judge" evaluator. You will use a separate LLM (e.g., GPT-3.5 or another API) to score the quality (e.g., politeness, relevance) of your model's generated outputs (minimum 20 samples) based on a clear rubric you design.
3. **Analyzing overfitting control:** You should experiment with and compare different hyperparameters, such as a learning-rate sweep or weight decay, and visualize the impact with Matplotlib loss/metric curves.

🚀 高级方向（探索至少两个。你也可以提出自己的方向。）

● **以数据为中心的变体：** 数据本身如何影响模型？尝试实现主动抽样或简单数据增强，并比较你的结果（理想情况下使用统计检验）。

● **泛化变体:** 你的模型学习效果如何？进行零样本、少样本与微调比较。你也可以分析随着训练样本数量增加，验证性能的扩展情况。

● **效率变体:** 存在哪些权衡？比较 QLoRA (4/8位) 与全精度微调（如果你的 GPU 允许），并报告训练时间和内存使用方面的差异。

● **安全性/鲁棒性检查:** 你的模型有多鲁棒？实施简单的毒性或提示注入探测，并记录潜在的缓解措施（例如：过滤、拒绝模板).
● **你自己的变体:** 提出一个复杂度相当或更高，替代高级方向。**这必须与讲师讨论并获得批准。**

📦 可交付成果 (项目A)

1. 可复现代码（理想情况下通过配置文件实现一键运行）。详细的README.md 文件。
2. 一份6–8页的报告：问题、方法、消融、结果、失败模式、局限性等。3. 一个简短的（≤ 3 分钟）预录屏演示你的项目。4. 第13周的最终演示。

📊 评分标准（项目A）

● 工作：60 %（包括重要性、原创性、结果与分析、难度与工作量、代码质量、报告清晰度等）
● 演示：40 %（教师、助教和同行评审，包括总体印象、表达清晰度、组织结构等）

项目B：采用ReAct模式的使用工具的LLM代理

主题： 一个LLM驱动的智能体，能够逐步推理、查阅工具，并返回有依据的、可审计的答案。

工作原理： 智能体是一个高度相关的行业话题。这个项目让你练习函数调用、错误处理、控制流以及推理可靠性的评估。

🌟 现实世界动机

设想你正在构建一个“个人旅行助手”。一个用户想问：“从我的城市出发，有什么好的三天周末旅行推荐？酒店费用大概会是多少？”仅凭大语言模型会产生幻觉。你的智能体可以通过以下方式解决这个问题：

1. 使用搜索工具查找“[user_city]附近的周末旅行”。（例如，观察：找到城市A、B、C。）

🚀 Advanced Directions (Explore at least two. You may also propose your own.)

- **Data-centric variant:** How does the data itself impact the model? Try implementing active sampling or simple data augmentation and compare your results (ideally with statistical tests).
- **Generalization variant:** How well does your model learn? Conduct a zero-shot vs. few-shot vs. fine-tuned comparison. You could also analyze the scaling of validation performance as you increase the number of training examples.
- **Efficiency variant:** What are the trade-offs? Compare QLoRA (4/8-bit) vs. full-precision fine-tuning (if your GPU permits) and report on differences in training time and memory usage.
- **Safety/robustness check:** How robust is your model? Implement a simple toxicity or prompt-injection probe and document potential mitigations (e.g., filtering, refusal templates).
- **Your Own Variant:** Propose an alternative advanced direction of equivalent or greater complexity. **This must be discussed and approved by the lecturer.**

📦 Deliverables (Project A)

1. Reproducible code (ideally a one-click run with a config file). Detailed README.md file.
2. A 6–8 page report: problem, method, ablations, results, failure modes, limitations, etc.
3. A short (≤ 3 min) pre-recorded screencast demoing your project.
4. A Final Presentation at Week 13.

📊 Grading Rubric (Project A)

- Work: 60 % (including Significance, Originality, Results & Analysis, Difficulty & Workload, Code Quality, Report Clarity, etc.)
- Presentation: 40 % (Teacher&TA&Peer review, including Overall Impression, Clarity of Speech, Organization, etc.)

Project B: Tool-Using LLM Agent with the ReAct Pattern

Theme: An LLM-powered agent that can reason step-by-step, consult tools, and return grounded, auditable answers.

Why this works: Agents are a highly relevant industry topic. This project allows you to practice function-calling, error handling, control flow, and the evaluation of reasoning reliability.

🌟 Real-World Motivation

Imagine you are building a "Personal Travel Assistant." A user wants to ask: "What's a good 3-day weekend trip from my city, and how much would a hotel cost?" An LLM alone would hallucinate. Your agent can solve this by:

1. Using a search tool to find "weekend trips near [user_city]." (e.g., Observation: Found cities A, B, C.)

2. 再次使用搜索工具查找“A市的酒店”。（例如，观察：找到酒店X、Y、Z。）3. 使用计算器工具“将[hotel_price] 乘以3。” 4. 最后，将这些信息整合为一个有依据的答案。

建议的工具集（选择 ≥2）：

● **搜索/文档：** 一个本地语料库或维基百科API子集；通过本地向量存储（例如FAISS）进行简单检索。
● **计算：** 一个安全数学/评估工具（Python评估器，沙盒化为基本算术）。
● **规划器：** 一个确定性规划器，将任务分解为子步骤（您自己的或一个轻量级计划骨架）。

💡 提示与入门

● 从简单入手！创建一个工具，例如计算器 (def add(a, b): return a + b)，并手动编写一个ReAct提示。您的首要目标应该是解析大语言模型的输出（例如，行动：add(2, 3)），调用您的Python函数，并将观察：5反馈回循环中。
● 大力侧重提示工程。你的提示词必须清楚地解释思想 ->行动 -> 观察 循环，并提供一些示例（小样本学习）。
● 错误处理至关重要。如果大语言模型调用了一个不存在的工具，或者提供了格式错误的JSON，会发生什么？你的智能体需要优雅地处理这种情况并报告一个错误观察。

🎯 核心目标

你的目标是构建一个可靠的代理，使其能够成功使用工具来回答问题。这里的成功意味着：

1. **实现最小ReAct循环：** 你的智能体应遵循 思想 → 行动（工具调用）→ 观察 → 最终答案 循环。这需要定义结构化工具模式（JSON输入/输出）并解析大语言模型的响应。 2. **添加基本防护措施：** 可靠的代理需要安全特性。你应该实施基本但关键的防护措施，例如 **最大步数**（以防止无限循环）和 **工具调用超时**。 3. **构建小型评估集：** 创建一个包含**~20-30个查询**的评估集，并附带“黄金”最终答案，以测试你的智能体正确使用工具的能力。 4. **启用基本日志记录进行调试：** 为了使智能体有用，它必须是可调试的。你应该将每个思想、行动和观察记录到文件或控制台，以创建代理的推理步骤的“跟踪”。

🚀 高级方向 (探索至少两个。您也可以提出自己的建议。)

● **自洽性或多数投票解码：** 如何提高可靠性？尝试运行K次追踪（多次智能体运行），并通过验证模型或简单的启发式方法选择最佳答案。

● **幻觉控制:** 您如何确保事实准确性？整合来自您的搜索工具的检索置信度分数，并添加引用要求。如果未找到高置信度证据，您的智能体可能会自动失败。

2. Using the search tool again for "hotels in city A." (e.g., Observation: Found hotels X, Y, Z.)
3. Using a calculator tool to "multiply [hotel_price] by 3."
4. Finally, synthesizing this information into a grounded answer.

Suggested toolset (select ≥2):

- **Search/Docs:** A local corpus or Wikipedia API subset; simple retrieval via a local vector store (e.g., FAISS).
- **Computation:** A safe math/eval tool (Python evaluator sandboxed to basic arithmetic).
- **Planner:** A deterministic planner that breaks tasks into substeps (your own or a lightweight plan skeleton).

💡 Hints & Getting Started

- Start simple! Create one tool, like a calculator (def add(a, b): return a + b), and write a ReAct prompt by hand. Your first goal should be to parse the LLM's output (e.g., Action: add(2, 3)), call your Python function, and feed the Observation: 5 back into the loop.
- Focus heavily on prompt engineering. Your prompt must clearly explain the Thought -> Action -> Observation cycle and provide a few examples (few-shot learning).
- Error handling is critical. What happens if the LLM calls a tool that doesn't exist or provides malformed JSON? Your agent needs to handle this gracefully and report an error observation.

🎯 Core Objectives

Your goal is to build a reliable agent that can successfully use tools to answer questions. Success here means:

1. **Implementing a minimal ReAct loop:** Your agent should follow the *thought → action (tool call) → observation → final answer* cycle. This requires defining structured tool schemas (JSON in/out) and parsing the LLM's responses.
2. **Adding essential guardrails:** A reliable agent needs safety features. You should implement basic but critical guardrails, such as a **maximum number of steps** (to prevent infinite loops) and **timeouts** for tool calls.
3. **Building a small evaluation set:** Create an evaluation set of **~20-30 queries** with "gold" final answers to test your agent's ability to use its tools correctly.
4. **Enabling basic logging for debugging:** For an agent to be useful, it must be debuggable. You should log each Thought, Action, and Observation to a file or the console to create a "trace" of the agent's reasoning steps.

🚀 Advanced Directions (Explore at least two. You may also propose your own.)

- **Self-consistency or majority-vote decoding:** How can you improve reliability? Try running K traces (multiple agent runs) and pick the best answer via a verifier model or a simple heuristic.
- **Hallucination control:** How do you ensure factual accuracy? Integrate retrieval confidence scores from your search tool and add a citation requirement. Your agent could auto-fail if no high-confidence evidence is found.

● **评判模型:** 您能否自动化评估？使用轻量级评估模型（例如，微调分类器）来评估您的代理轨迹的事实性和工具使用效率。
● **安全过滤器：** 你如何防御攻击？使用启发式模式实现一个提示注入检测器，并演示攻击和你的缓解措施。
● **你自己的变体：** 提出一个具有同等或更高复杂性的替代高级方向。**这必须与讲师讨论并获得批准。**

📦 可交付成果 (项目B)

1. 代理框架代码（最好是带配置文件的+ 一键运行）。工具规范（文档齐全）。详细的README.md 文件。 2. 一个评估集（20-30+ 条查询，附带标准答案）和一个评分脚本。
3. 一份6-8页的报告：设计选择、故障分析、消融（例如，有/无检索，有/无验证器）等。
4. 一个简短的（≤ 3 分钟）预录屏演示你的项目。 5. 第13周的最终演示。

📊 评分标准 (项目B)

● 工作：60%（包括重要性、原创性、结果与分析、难度与工作量、代码质量、报告清晰度等）
● 演示: 40 % (教师、助教和同行评审，包括总体印象、表达清晰度、组织结构等)

项目C: 检索增强生成 (RAG) 与端到端评估

主题: 构建一个领域特定助手，利用 RAG 从提供的语料库中回答问题，重点关注索引、分块和评估。

为何有效: RAG 是当今最普遍的企业LLM模式。本项目提供了数据工程、向量搜索、提示设计、批处理、缓存和客观评估方面的动手实践经验。

🌟 现实世界动机

设想您想为某个特定领域构建一个“问答机器人”。

● **技术方向：** 构建一个机器人，通过阅读 Pandas 或 Scikit-learn 等库的官方文档，回答高度具体的问题。再也不用为了一个答案在 10 个不同的网页上搜索了。

● **娱乐方向：** 构建一个针对《艾尔登法环》或《赛博朋克2077》等电子游戏的“游戏背景知识专家”。你给它输入游戏的维基，它就能回答关于角色、地点和历史的深度问题，并附带引用。

提供的资产（选择一个领域语料库）：

- **Judge model:** Can you automate evaluation? Use a lightweight evaluator model (e.g., a fine-tuned classifier) to score the factuality and tool-use efficiency of your agent's traces.
- **Safety filter:** How do you defend against attacks? Implement a prompt-injection detector using heuristic patterns and demonstrate both an attack and your mitigation.
- **Your Own Variant:** Propose an alternative advanced direction of equivalent or greater complexity. **This must be discussed and approved by the lecturer.**

📦 Deliverables (Project B)

1. Agent framework code (ideally a one-click run with a config file) + tool specifications (well-documented). Detailed README.md file.
2. An evaluation set (20-30+ queries with gold answers) and a scoring script.
3. A 6–8 page report: design choices, failure analysis, ablations (e.g., with/without retrieval, with/without verifier), etc.
4. A short (≤ 3 min) pre-recorded screencast demoing your project.
5. A Final Presentation at Week 13.

📊 Grading Rubric (Project B)

- Work: 60 % (including Significance, Originality, Results & Analysis, Difficulty & Workload, Code Quality, Report Clarity, etc.)
- Presentation: 40 % (Teacher&TA&Peer review, including Overall Impression, Clarity of Speech, Organization, etc.)

Project C: Retrieval-Augmented Generation (RAG) with End-to-End Evaluation

Theme: Build a domain-specific assistant that answers questions from a provided corpus using RAG, emphasizing indexing, chunking, and evaluation.

Why this works: RAG is the most ubiquitous enterprise LLM pattern today. This project provides hands-on experience with data engineering, vector search, prompt design, batching, caching, and objective evaluation.

🌟 Real-World Motivation

Imagine you want to build a "Q&A Bot" for a specific domain.

- **For Tech:** Build a bot that can answer highly specific questions about a library like pandas or scikit-learn by reading its official documentation. No more hunting through 10 different web pages for an answer.
- **For Fun:** Build a "Game Lore Expert" for a video game like *Elden Ring* or *Cyberpunk 2077*. You feed it the game's wiki, and it can answer deep questions about characters, locations, and history, complete with citations.

Provided assets (pick one domain corpus):

● **技术文档：**（例如，Python、NumPy、PyTorch 或 Pandas 摘录）
● **政策/手册：**（例如，学术政策，住房规定）
● **小型研究语料库：**（10-20 份调查/教程论文；摘要+ 部分）
● **您的选择：**您自己的语料库（例如，游戏维基、一套规章制度），大小类似。

💡 提示与入门

● 您的首要任务应该是数据管道。首先编写一个脚本来加载、清理和分块您的文档。一个简单的固定大小分块器是一个很好的起点。
● 使用像 sentence-transformers 这样的库来生成嵌入，并使用 faiss-cpu 来构建一个简单、快速的向量索引。
● 尽早构建你的评估集。在提供的初始集基础上进行扩展，将使你在试验不同的分块器、检索器和提示时，更容易客观地衡量你的进展。

● 你的提示是关键。一个好的RAG提示会清楚地指示大语言模型只使用提供的上下文来回答问题，并且如果答案不存在，则说“我不知道”。

🎯 核心目标

你的目标是构建和评估一个完整的RAG系统，重点关注检索和生成两方面的质量。成功意味着：

1. 构建完整的RAG管道：这包括数据清洗、实施分块策略（例如，固定大小与语义分割）、生成嵌入、创建向量索引、构建检索器，并将检索到的上下文传递给生成器（大语言模型）。
2. 创建强大的评估：你必须实施“闭卷”（无RAG）与RAG评估。你的任务是从零开始创建一个强大的评估集，至少包含50对总计的（带有真实答案和来源引用），以测试你的系统。 3. 运行对比实验：你应该比较至少两个检索器（例如，BM25等稀疏检索器与密集检索器；或两种不同的密集模型）和至少两个提示（例如，基础提示与更专业的指令微调提示）。 4. 报告综合指标：你需要报告检索指标（Recall@k、MRR）和答案质量指标（完全匹配/F1分数或ROUGE）。这应该通过实施一个“LLM作为评判者”评估器（至少30个样本）来补充，以评估答案的定性方面（例如，忠实性、相关性）。

🚀 高级 方向（至少探索两个。您也可以提出自己的方向） .)

● **查询重写：**如果用户的查询不好怎么办？实施像HyDE（假设文档嵌入）这样的技术，或者“精简问题”步骤（使用大语言模型重写查询），并评估其增益。

● **重排序：**您的检索器找到了10个文档，但前3个是最好的吗？添加一个轻量级的交叉编码器重排序器来重新排序最初检索到的文档，并执行消融研究。

● **延迟与内存分析：**您的系统效率如何？分析批处理大小、最大令牌长度和缓存等因素。您能生成一个吞吐量与质量图表吗？
● **安全与隐私：** 如果您的文档包含敏感信息怎么办？实施一个步骤来匿名化

- **Tech docs:** (e.g., Python, NumPy, PyTorch, or pandas excerpts)
- **Policy/handbook:** (e.g., academic policies, housing rules)
- **Mini research corpus:** (10–20 survey/tutorial papers; abstracts + sections)
- **Your choice:** A corpus of your own (e.g., a game wiki, a set of regulations) of similar size.

💡 Hints & Getting Started

- Your first priority should be the data pipeline. Start by writing a script to load, clean, and chunk your documents. A simple fixed-size chunker is a great starting point.
- Use a library like sentence-transformers to generate embeddings and faiss-cpu for a simple, fast vector index.
- Build your evaluation set *early*. Expanding on the provided starter set will make it much easier to objectively measure your progress as you experiment with different chunkers, retrievers, and prompts.
- Your prompt is key. A good RAG prompt clearly instructs the LLM to *only* use the provided context to answer the question and to say "I don't know" if the answer isn't present.

🎯 Core Objectives

Your goal is to build and evaluate a full RAG system, focusing on the quality of both retrieval and generation. Success here means:

1. **Building the full RAG pipeline:** This involves data cleaning, implementing a chunking strategy (e.g., fixed-size vs. semantic split), generating embeddings, creating a vector index, building a retriever, and passing the retrieved context to a generator (LLM).
2. **Creating a strong evaluation:** You must implement a "closed-book" (no RAG) vs. RAG evaluation. Your job is to create from scratch a strong evaluation set of at least 50 total pairs (with ground-truth answers and source citations) to test your system.
3. **Running comparison experiments:** You should compare at least two retrievers (e.g., a sparse retriever like BM25 vs. a dense retriever; or two different dense models) and at least two prompts (e.g., a vanilla prompt vs. a more specialized instruction-tuned prompt).
4. **Reporting comprehensive metrics:** You need to report retrieval metrics (Recall@k, MRR) and answer quality metrics (Exact Match / F1 or ROUGE). This should be supplemented by implementing an "LLM-as-Judge" evaluator (for at least 30 samples) to assess the qualitative aspects of your answers (e.g., faithfulness, relevance).

🚀 Advanced Directions (Explore at least two. You may also propose your own.)

- **Query rewriting:** What if the user's query is bad? Implement a technique like HyDE (Hypothetical Document Embeddings) or a "condense question" step (using an LLM to rewrite the query) and evaluate the gains.
- **Re-ranking:** Your retriever finds 10 documents, but are the top 3 the best? Add a lightweight cross-encoder re-ranker to re-order the initial retrieved documents and perform an ablation study.
- **Latency & memory profiling:** How efficient is your system? Profile factors like batch size, max token length, and caching. Can you produce a throughput vs. quality chart?
- **Safety & privacy:** What if your documents contain sensitive info? Implement a step to redact

语料库中类似PII的模式，并衡量这对其可回答性的影响。
● **你自己的变体：** 提出一个具有同等或更高复杂度的替代高级方向。 **这必须与讲师讨论并获得批准。**

📦 可交付成果 (项目C)

1. 可复现的代码库（最好是带配置文件的“一键运行”）。详细的 README.md 文件。 2. 你的评估集（50+ 问答对）和你的评分脚本。 3. 一份 6–8 页的报告：你的设计、实验、成本/延迟-质量权衡等。 4. 一个简短的（≤ 3 分钟）预录屏演示，演示你的项目。 5. 第13周的最终演示。

📊 评分标准 (项目C)

● 工作：60%（包括重要性、原创性、结果与分析、难度与工作量、代码质量、报告清晰度等）
● 演示：40%（教师、助教和同行评审，包括总体印象、表达清晰度、组织结构等）

PII-like patterns in the corpus and measure the impact this has on answerability.
- **Your Own Variant:** Propose an alternative advanced direction of equivalent or greater complexity. **This must be discussed and approved by the lecturer.**

📦 Deliverables (Project C)

1. Reproducible repository (ideally a one-click run with a config file). Detailed README.md file.
2. Your evaluation set (50+ QA pairs) and your scorer script.
3. A 6–8 page report: your design, experiments, cost/latency-quality trade-off, etc.
4. A short (≤ 3 min) pre-recorded screencast demoing your project.
5. A Final Presentation at Week 13.

📊 Grading Rubric (Project C)

- Work: 60 % (including Significance, Originality, Results & Analysis, Difficulty & Workload, Code Quality, Report Clarity, etc.)
- Presentation: 40 % (Teacher&TA&Peer review, including Overall Impression, Clarity of Speech, Organization, etc.)

3. 共享物流和要求

计算与环境

- **目标：** 所有项目均适用于 CPU 或单个 T4/V100/A10 图形处理器，运行于 GoogleColab/Kaggle (8–12GB 显存)。
- **可复现性:** 您必须强制使用确定性种子并提供一个 requirements.txt 文件。
- **约束:** （针对项目A的）训练运行应限制在T4 GPU上≤ **2 小时**的实际运行时间内。

整体评分理念

您的成绩将基于对您的工作的全面评估，并根据项目专用评分标准进行加权。除了技术组件之外，我们还关注：

- **技术深度与正确性:** 您的项目是否有效，并且是否建立在对概念的扎实理解之上？
- **实验设计与可复现性:** 您是否设计了公平的实验来检验您的想法？其他人能否运行您的代码并获得相同的结果？
- **结果质量与分析:** 您是否超越了单纯报告数字，并提供了对您的系统为何如此运行（包括故障分析）的深刻分析？
- **报告质量与沟通:** 您的报告、演示和代码是否清晰、专业且易于理解？您是否审慎分析了**局限性**和伦理影响？
- **演示文稿/演示:** 您是否在规定时间内有效地传达了您项目的目标、方法和结果？

学术诚信

您可以使用开源库和预训练检查点。但是，您**必须**在报告中提供一份“材料清单”，列出所有使用的资产、库和检查点，并附上适当的引用。需要提交团队贡献声明。请准备好接受对您的提交内容进行的随机口头检查。

3. Shared Logistics and Requirements

Compute & Environment

- **Target:** All projects are scoped for CPU or a single T4/V100/A10 GPU on Google Colab/Kaggle (8–12GB VRAM).
- **Reproducibility:** You must enforce deterministic seeds and provide a requirements.txt file.
- **Constraint:** Training runs (for Project A) should be limited to ≤ **2 hours** of wall-clock time on a T4 GPU.

Overall Grading Philosophy

Your grade will be based on a holistic assessment of your work, weighted according to the project-specific rubrics. Beyond the technical components, we are looking for:

- **Technical depth & correctness:** Does your project work, and is it built on a solid understanding of the concepts?
- **Experimental design & reproducibility:** Did you design fair experiments to test your ideas? Can someone else run your code and get the same results?
- **Results quality & analysis:** Did you go beyond just reporting numbers and provide insightful analysis of *why* your system behaves the way it does (including failure analysis)?
- **Report quality & communication:** Is your report, demo, and code clear, professional, and easy to understand? Did you thoughtfully analyze the **limitations** and ethical implications?
- **Presentation/demo:** Did you effectively communicate your project's goals, methods, and results in the allotted time?

Academic Integrity

You are permitted to use open-source libraries and pretrained checkpoints. However, you **must** require a “Bill of Materials” in your report, listing all assets, libraries, and checkpoints used, with proper citations. Team contribution statements are required. Be prepared for random oral checks on your submission.