Data Analysis and Presentation – Final Project

Submission Instructions

1. **Submission Deadlines**: There are two possible submission deadlines for the project to accommodate both students who wish to complete their degree early and those who prefer to take more time to delve into their projects. The available submission dates are September 23, 2024, and November 14, 2024. You must inform us in advance of your intended submission date through the meeting schedule file. You can change your mind later, but you must notify us if you decide to do so.

2. **Group Submissions**: Projects must be submitted according the same assignments groups

3. **Submission File Format**: Each team should submit a zip file named `<ID1_ID2_ID3>`, where `<ID<i>>` represents the ID number of each of the 3 (or 4) team members. The zip file must include the following files:

   - `<ID1_ID2_ID3>.pdf`
   - `<ID1_ID2_ID3>.MP4`

   where the pdf file is the final project report and the MP4 file is a the project video presentation (both detailed next).

4. **Report Requirements**:

   - The report should include a link to a GitHub repository, as detailed in the instructions provided later.
   - The report (PDF file) must be written in English.

5. **Questions**:

   - Any general questions about the project should be posted in the Moodle forum.
   - Specific questions can be asked through the course email `dav094295@gmail.com`. Please include an indicative title, mention who are the team members and write your questions clearly in a numbered list.

6. **GPU Server Usage**: You can use the GPU server assigned for you. Please follow the instructions provided in the file `Students - How to work with your VM.pdf`.

Introduction

The goal of this project is to gain a hands-on experience with one or more topics covered in the course. you are expected to demonstrate a deep understanding of the selected task, define a related problem, suggest a solution, compare it with other potential solutions, write a detailed and comprehensive report, implement your work with a clear and reproducible code and present the project in a video. This document details the guidelines and requirements for the project.
We next provide a recommended outline for the project structure according to the topic you chose to focus on. Your project should focus on at least of these, though you may also choose to combine them. Following this, we will explain the submission instructions and detail the grade structure.

1 Approximate Nearest Neighbor

Approximate Nearest Neighbor (ANN) search is a fundamental component in vector DB, allowing for a faster search with a compromise in the accuracy of the results. The focus of a project in this topic should be on a significant experimental setup and solution space exploration. However, if you manage to suggest a genuine algorithmic approach it will be well-appreciated. A general outline for such a project can include:

- **Problem Definition and Literature Review:**
  - Clearly define the problem you aim to solve or settings you want to explore. Specifically, focus on interesting and non-trivial settings in which traditional solutions tend to fail.
  - Conduct a short literature review to understand existing solutions and identify gaps.
- **Data Collection and Preprocessing:**
  - Collect relevant datasets for your project.
  - Perform necessary preprocessing steps such as data cleaning (if needed).
  - Embed your data. Which model did you use?
  - Consider generating a synthetic dataset suitable for your settings of the problem
- **Experimentation and Evaluation:**
  - Suggest several baselines to explore.
  - Design and conduct experiments to evaluate the performance of your algorithms. Think carefully what are the evaluation metrics that should be used.
  - Present the results using various means (tables, plots, *etc.*).
  - Discuss any trade-offs derived from the experiments, as well as the limitations of your experimental setup.
  - Highlight the main observations and insights.

2 Retrieval Augmented Generation

Retrieval-Augmented Generation (RAG) is a technique that utilizes vector database to retrieve relevant information, and then generate a response based on the retrieved information. By relying on relevant information from the database, LLMs can generate more informative and coherent responses and (hopefully) reduce the generation of incorrect or irrelevant information. As numerous interesting projects are available in the internet, if you choose to focus on this topic it shall be something genuine. It can be either designing a complete pipeline including user interface or diving deeper into the components of RAG (indexing, retrieval or generation). A general outline for such a project can include:

- **Use Case Suggestion:**
  - Clearly define the problem you aim to solve with RAG. What is the motivation for using RAG instead of just chatting with a LLM?
  - Is the RAG part of a bigger pipeline (*e.g., a chatbot*)? If so, please provide an outline of the pipeline.
- **Data Collection and Preprocessing:**
  - Collect relevant datasets for your use case and explain about it. You may choose more than a single dataset.
  - Perform necessary preprocessing, including text cleaning and chunking of documents into manageable sizes.
- **Indexing and Retrieval**
  - Select embedding model and insert your data into a vector DB.
  - Implement a retrieval system to fetch relevant documents based on the query embeddings. Did you make any modifications in standard approach to cope with the requirements of your system? Did you use advanced methods such as rerankers?
  - Explain how you verified the quality of your retrieval phase.
  - Document your prompting methods in the Github repository and discuss them later in the experimental section.
- **Experimentation and Evaluation:**
  - Suggest several baselines to compare your algorithm with.
  - Design and conduct experiments to evaluate the performance of your RAG system.
  - Compare your solution with the baselines. Which evaluation metrics did you use? Did you use LLMs as part of the evaluation pipeline?
  - Discuss any trade-offs between retrieval accuracy and generative quality.
  - Highlight the main observations and insights

3 Active Learning

Active learning is a machine learning approach where the model selectively queries the most informative data points from a pool of unlabeled data. This technique aims to improve learning efficiency and performance by focusing on the most relevant examples, reducing the amount of labeled data needed. Active learning is particularly useful in scenarios where labeled data is scarce or expensive to obtain. A project in this topic should consider the following aspects:

- **Use Case Suggestion:**
  - Develop an active learning system for a real-world application.
- **Problem Definition and Literature Review:**
  - Clearly define the problem you aim to solve with active learning.
  - Conduct a literature review to understand existing active learning strategies and their applications.
- **Data Collection and Preprocessing:**
  - Collect relevant datasets for your use case.
  - Perform necessary preprocessing.
- **Algorithm Selection and Implementation:**
  - Select appropriate active learning strategies (*e.g.*, random, uncertainty sampling, query-by-committee) to compare with.
  - Suggest a genuine approach, suitable for your settings.
  - Implement these strategies and integrate them with a machine learning model.
- **Experimentation and Evaluation:**
  - Suggest several baselines to compare your algorithm with.
  - Design and conduct experiments to evaluate the performance of your active learning system.
  - Which evaluation metrics did you use?
  - Discuss any trade-offs between labeling effort and model performance.
  - Did you incorporate LLMs as part of the pipeline? How?
  - Present the results using various means (tables, plots, *etc.*).

4 Graph Neural Networks

Graph Neural Networks (GNNs) are a class of neural networks designed to process and analyze graph-structured data.GNNs can effectively handle complex relationships and dependencies between data objects, making them extremely powerful for tasks involving social networks, biological networks, recommendation systems, and more. Just like RAG, the internet is full with projects in this field. Hence, you are expected to include a genuine part in your work, and perform a thorough experimental analysis. A structure of the project may be:

- **Use Case Suggestion:**
  - Develop a GNN model for an interesting use case. Explain why a GNN may be useful for this case.
  - Conduct a literature survey to understand existing solutions for this type of data.
- **Data Collection and Preprocessing:**
  - Collect relevant datasets for your use case.
  - Perform necessary preprocessing, including converting the data into graph structures where nodes represent entities (e.g., users, items) and edges represent relationships or interactions. Explain using formal notation how your data is modeled by the graph structure. How edges are formed? Are the weighted? Do you have multiple node types?
  - Explore different graph construction techniques
  - Explain how did you generate the inital feature vectors.
- **Algorithm Selection and Implementation:**
  - Select appropriate GNN architectures (e.g., Graph Convolutional Networks, Graph Attention Networks).
  - Did you use additional deep learning techniques (*e.g.,* batch normalization) to improve performance?
- **Experimentation and Evaluation:**
  - Suggest several baselines (both graphs and non-graph approach) to compare your solution with.
  - Design and conduct experiments to evaluate the performance of your GNN model. Which evaluation metrics did you use?
  - Use relevant metrics such as accuracy, precision, recall, and F1-score to assess the model's effectiveness.
  - Analyze how various architectures impact the model's performance and ability to learn from graph-structured data.

Project Meetings and Project Proposal

Each team should attend a single 20-minute meeting with the course staff. Schedule your meeting according to the instructions provided on Moodle. Within a week after the meeting, submit a one-page project proposal that outlines your project, including the project title and the names and emails of your team members.

Submission

Your final submission should include a PDF report structured similarly to a short academic paper. The report should clearly convey the entire process you went through during your project and summarize all aspects of your work. The report must be a maximum of 6 pages, excluding the appendix. The appendix should list the main papers, videos, and blog posts (no need to add a formal bibliography). By and large, your report should be structured as follows:

- **Title and Author Information:** The name of your work, and the names and emails of the team members.
- **Abstract:** A brief summary of the project, including the problem addressed, the methods used, and the main findings.
- **Introduction:** Introduce the problem and its significance. Provide an overview of the existing solutions and point at existing gaps.
- **Methodology:** Describe the dataset and preprocessing steps, and detail the algorithms and models used.
- **Experiments:** Explain the experimental setup and evaluation metrics. Conduct experiments to demonstrate the key points conveyed in your work. Include tables, graphs, and other visual aids to support your findings. Discuss the performance of your approach compared to baselines.
- **Short Discussion:** Provide insights and potential improvements for future work. Summarize the key findings of your project.
- **Appendix:** List the main papers, videos, and blog posts you relied on. Include a link to your Github repository containing all code and related materials.

Code

Your GitHub repository is a major component of your project submission. It should demonstrate not only the functionality of your code but also the quality of your collaboration and documentation. The repository must adhere to the following requirements:

- **Code Quality and Structure:** Ensure that your code is well-structured and organized into meaningful files (and if needed, directories).
- **README file:** Include a `README` file that provides an overview of the project, setup instructions, and how to run your code.
- **Documentation:** Document all significant parts of your code using comments and docstrings.
- **Collaboration Evidence:** Use Git effectively to show the collaboration between team members. Ensure that the commit history reflects contributions from multiple users. Include meaningful commit messages that describe the changes made.
- **Reproducibility:** Ensure that your code can be easily set up and run by others. This includes providing clear setup instructions and listing any dependencies.

Video Presentation

The video presentation should be attached to you submission. It should allow you to concisely communicate the key aspects of your project and demonstrate your work visually. The video length must be at most 7 minutes, in which you present the problem, motivation, methodology, experiments and insights of your work.

Grading

Your final project will be evaluated based on three main components: the written report, the code submission, and the video presentation. The grading distribution is as follows:

- **Report (60%):**
  - **Content and Clarity:** Your report should thoroughly explain the problem, methodology, experiments, results, and conclusions.
  - **Novelty:** Your work cannot be a copy-paste of existing works. Your novelty should be well-stressed in the report.
  - **Effort:** The depth of your analysis, the comprehensiveness of your experiments, and the overall effort invested in the project will be assessed.
  - **Documentation:** Clear documentation of your process, including the rationale behind your choices and detailed explanations of your methods.
- **Code (30%):**
  - **Functionality:** Your code should run correctly and produce the results described in your report.
  - **Organization:** The code should be well-organized into meaningful directories and files.
  - **Documentation:** Code should be well-commented, with clear explanations of algorithms, functions, and any non-trivial steps.
  - **Collaboration:** Effective use of Git, reflecting contributions from all team members, with meaningful commit messages.
- **Video Presentation (10%):**
  - **Clarity:** The presentation should be clear and easy to follow, highlighting the key aspects of your work.
  - **Demonstration:** If you implemented an entire system, briefly demonstrate it in the video.