Case 3. Prompt Engineering for Enhanced Medical Image Classification

Applications of Neural Networks in Medicine

The following instructions are developed using prompt engineering with Metropolia's M653 cloud-based Copilot (Accessed 19.11.2024).

Type

Teamwork, 15 p

Aim

The aim of this assignment is to leverage large language models (LLMs), such as ChatGPT or Copilot, and prompt engineering to improve the performance of the CNN models developed in the 2nd assignment.

Task

First, review the 2nd assignment. Revisit the CNN models created for the pneumonia X-ray image classification task. Identify the best-performin model based on sensitivity and specificity.

Then, use LLMs to get guidance and tips on improving the code and performance of the CNN models. Experiment with different prompts to obtain suggestions for optimizing the model architecture, hyperparameters, data augmentation techniques, etc.

Document prompts and outcomes. Keep a record of the prompts used and the corresponding responses from the LLM. Implement the suggestions provided by the LLM and evaluate their impact on the model's performance.

Assess the quality of the prompts based on their relevance and usefulness of the responses. Analyze how the prompts could be refined to get more targeted and effective guidance.

Compile a report documenting the entire process, including the prompts, responses, code changes, and performance improvements. Discuss the effectiveness of using LLMs and prompt engineering in this context.

Return

Submit a Word or PDF document containing the documented process and the improved model. Ensure the report is well organized, with clear explanations and comments. Consider the following structure:

1. **Introduction**
   - Briefly explain the report's purpose and the role of prompt engineering in your project's context.
2. **Methodology**
   - Describe the process of prompt engineering, including the tools used (e.g., ChatGPT, Copilot). Explain how prompts were designed and refined throughout the project.
3. **Prompts and Responses**
   - Provide a table or list of the prompts used and the corresponding responses from the language model. Highlight any significant suggestions or insights gained from the responses.
4. **Implementation**
   - Detail how the suggestions from the language model were implemented in the code. Include code snippets before and after applying the changes, if applicable.
5. **Evaluation**
   - Compare the model's performance before and after applying the changes suggested by the language model. Use metrics such as accuracy, sensitivity, specificity, and F1-score to quantify the improvement.
6. **Analysis**
   - Discuss the relevance and usefulness of the prompts. Analyze the impact of prompt engineering on the code quality, development time, and computational resources.
7. **User Experience**
   - Include feedback from team members on their experience with prompt engineering. Discuss whether it facilitated learning and understanding of the subject matter.
8. **Conclusion**

    - Summarize the key findings and the overall impact of prompt engineering on the project. Provide recommendations for future use of prompt engineering in similar projects.

9. **Appendices**
    - Include any additional information, such as the full list of prompts and responses, or extended code snippets.

Remember to keep the documentation clear and concise, focusing on how prompt engineering contributed to the project's outcomes. Use visual aids like tables and charts to present data effectively. Good luck with your report! 📄 ✨

Evaluation

- Organization
    - The document follows a clear structure
- Clarity
    - The document is clear, polished, and easy to understand
- Contents
    - The contents are reasonable and contain clear conclusions aligned with results.

Max. 15 points. Late submission reduces the maximum achievable points.

Evaluation

The following categories are used for evaluation:

- Organization
  - The code is sequential, and the code cells (parts of scripts) are in proper order.
  - The document follows a clear structure.
- Clarity
  - The document (and embedded code) is clear, polished, and easy to understand.
  - The code follows good coding practices and contains sufficient comments.
  - The document parts support the code.
- Contents
  - The background and data preprocessing are well explained.
  - The models are validated.
  - The results are reasonable.
  - The conclusions are clearly stated and in line with the results.

Max. 15 points. Late submission reduces the maximum achievable points.

Materials

- Deep Learning with Python.
  - Chapter 6. Deep learning for text and sequences.
- Deep learning for text
- GitHub - fchollet/deep-learning-with-python-notebooks samples
  - See the first edition folder
    - 6.1. One-hot encoding of words or characters
    - 6.1. Using word embeddings
    - 6.2. Understanding recurrent neural networks
- Tutorials | TensorFlow Core
  - Beginner: Load and Preprocessing data > Text
  - Advanced: Text