Blake Green

Professor Xihao Xie

CS 5393.004

10 April 2025

Exam 2: Comprehensive Report

Overview of Models Chosen

I carefully weighed architectural variety, resource needs, and overall capability when choosing models for this program. Ultimately, I selected three open-source language models available via Ollama: Microsoft's Phi, Google's Gemma:7b, and Meta's Llama2:13b. By picking one model from each company, I ensured a broad mix of design philosophies and technical approaches.

Phi is the smallest model, having 1.3 billion parameters. I chose it for its efficiency in handling routine language tasks while using minimal resources. This potentially makes Phi a reliable candidate for tasks like answering questions, summarizing text, and even translation work, especially when fast responses and low resource use are top priorities.

Gemma:7b holds 7 billion parameters and balances efficiency and performance well. It handles simple code generation and creative writing well, making it ideal for systems with moderate hardware. Its intermediate size also allows for detailed comparisons and practical testing.

I initially considered llama3.3 for its advanced text generation. However, its heavy computational requirements forced me to switch to llama2:13b. As the largest model in my selection (with 13 billion parameters), llama2:13b delivers strong language generation for

complex tasks while still being manageable on local machines. This choice helped me ensure high-quality results without overloading my hardware.

In short, the program was established with a diverse toolset by selecting phi, gemma:7b, and llama2:13b from different companies. This variety supports a detailed analysis of essential language tasks while keeping resource use practical and applicable in real-world scenarios.

Detailed Results from Basic Exploration and Focused Experimentation

My hands-on tests with Gemma:7b, Llama2:13b, and Phi produced valuable insights across tasks such as general knowledge questioning, text summarization, code generation, creative writing, and language translation.

Gemma:7b and Llama2:13b for general knowledge questions delivered short, accurate answers to all four queries. Phi also performed reasonably well but notably missed the mark on Task 3 ("Who wrote '1984'?") by providing an off-topic answer. In addition, phi's extra wordiness—meant to add context—sometimes led to mistakes (for example, referring to "Lake Paris" instead of the correct term). In contrast, Gemma:7b used markdown formatting to highlight important details, while Llama2:13b kept its language simple.

In the text summarization task, each model had its style. All three identified the key points in the Reuters news excerpt and a Gutenberg short story. Gemma:7b delivered short, structured summaries that condensed the essential information without extra fluff. Phi's summaries balanced being short with enough detail, staying factual and neutral. Meanwhile, Llama2:13b offered thorough narratives but sometimes added unnecessary information, like random mentions of copper prices, that could distract from the main summary. These differences show that each model has strengths depending on the reader's needs.

For code generation, Gemma:7b produced clear, well-documented Python functions complete with docstrings, usage examples, and even breakdowns of complexity. Llama2:13b's code was also effective, but more to the point, lacking extra commentary. On the other hand, Phi added creative twists to standard programming prompts, as seen when it compared programming languages during a factorial calculation. While these creative touches were engaging, they sometimes strayed from the pure coding task. Notably, phi's slight edge in efficiency—for example, smartly skipping even numbers when checking for primes—showed a practical performance benefit.

When tackling creative writing, the models again revealed their unique voices. Phi offered detailed storytelling with emotional depth and even added unexpected logic puzzles after the main poem. Gemma:7b relied on rich poetic imagery, effectively capturing sensory details. Llama2:13b produced clear, easy-to-read narratives and poems that stayed directly on topic. Although Phi's approach showed great flexibility, its tendency to wander occasionally impacted the focus of the task.

The differences were clear during language translation. Gemma:7b consistently produced fluent and mostly accurate translations with only minor word slips. Llama2:13b also performed well, though it occasionally showed stylistic inconsistencies and some code-switching—especially in English-to-French tasks. Phi's performance was the most erratic, often adding unrelated content (like logic puzzles) that hurt the clarity and usefulness of its translations, especially noticeable in its English-to-German attempts.

Overall, Gemma:7b emerged as the most consistent in delivering precise, clear answers, making it ideal for tasks that require high accuracy and short responses. Phi's performance was more variable, while Llama2:13b balanced accuracy and engaging narrative detail.

Analysis of Models' Strengths and Weaknesses.

Gemma:7b

Strengths:

Gemma:7b consistently delivered precise and clear results across the board, excelling in tasks focused on accuracy. When answering general questions, it provided short, direct responses enhanced by well-structured markdown formatting. While short, its summaries still captured all the important details—perfect for scenarios requiring quick information extraction. Additionally, its code generation outputs featured well-documented Python functions with thorough explanations and practical examples. Gemma also excelled at creative writing by offering vivid, sensory-rich narratives and maintaining fluent and reliable translations with only minor errors.

Weaknesses:

The main drawback of Gemma:7b is its slower execution time. Tasks such as summarization and creative writing take longer to process, which might be problematic when real-time responses are needed. Although there were a few minor translation errors, these were generally less significant than the delays in other tasks.

Llama2:13b

Strengths:

Llama2:13b struck a good balance between clarity and detail. Its responses to general questions were clear and straight to the point, without extra elaboration. The model excelled in summarization by providing context-rich narratives for readers looking for in-depth information,

even if it sometimes included extra details. Code generation produced short solutions that were easy to implement. Its creative writing aligned well with the prompts, and its translation tasks effectively preserved meaning, with fewer significant errors.

Weaknesses:

Despite its overall solid performance, Llama2:13b sometimes suffered from longer processing times, especially in summarization and creative writing, which could be problematic when speed is crucial. Its habit of adding extra details in summaries occasionally detracts from the content's relevance, and minor stylistic inconsistencies in translation suggest that there is still room for improvement in handling multiple languages.

Phi

Strengths:

Phi is notable for its creative and thorough explanations that often go beyond the basic requirements. Though wordy, its responses to general questions offer extra context that can be educational. Phi excels in creative writing, crafting detailed narratives with emotional depth, adding additional elements like puzzles for an inventive twist. Code generation shows clever optimizations, such as efficiently skipping unnecessary steps.

Weaknesses:

However, Phi's tendency to be very wordy sometimes undermines accuracy and focus. This overexplanation leads to errors—for example, imprecise geographic references—and occasionally derailed its summarization responses by introducing off-topic details. Its

translations were also affected by unrelated additions like logic puzzles, which hurt the clarity and overall usefulness for practical applications.

Performance Efficiency

Gemma:7b and Llama2:13b experienced longer execution times and higher CPU usage, especially during creative writing and summarization. These factors make them less suitable for real-time applications or situations with limited computational resources. In contrast, phi generally processed simpler tasks more quickly, though it encountered significant computational overhead when handling extended and complex queries.

In essence, the choice of model depends on your specific needs: choose Gemma if accuracy and short responses are most important, opt for Llama2 if you need detailed, narrative-driven content, and turn to Phi when the situation calls for creative elaboration—even if it sometimes affects accuracy.

Insights Gained about Open-Source LLMs

My experimentation with different open-source LLMs has highlighted their diverse strengths, limitations, and the scenarios where they work best. I noticed that the underlying architecture greatly influences a model's performance on specific tasks. For example, Gemma:7b's excellent performance in language translation might be linked to Google's strong background in translation technology. This suggests that understanding a model's basic design and training can be key to choosing the right tool for a given task.

I also observed that newer, larger models like llama3.3 require significantly more computational resources while offering superior language generation and understanding. This

makes them less practical for everyday use on local hardware, prompting the use of alternatives such as llama2:13b, which more effectively balance power and resource use.

On the other hand, smaller models like Phi, though more resource-efficient, can struggle with precision and prompt understanding. Phi's tendency to include extra creative information—helpful in some educational or artistic settings—can lead to inaccuracies in tasks where exact information is critical.

A clear trend emerged: larger, more recent models generally perform better than their smaller, older counterparts because their higher number of parameters helps them understand context and deliver richer responses. However, these advantages come with the cost of higher computational overhead. Therefore, selecting the right open-source LLM involves balancing model size, available resources, task specifics, and each model's strengths and weaknesses. Gemma:7b is excellent for accuracy and clear communication, phi is ideal for creative tasks (despite occasional precision issues), and although models like llama3.3 are competent, they may not be practical for widespread use.

Reflections on Practical Implications of Findings

My detailed exploration of phi, gemma:7b, and llama2:13b has important real-world implications. The key takeaway is that model selection must match the specific task and the available computational resources.

For example, Gemma:7b's clear, concise, and reliable outputs make it especially useful in professional settings—whether legal, medical, or academic—where quick access to accurate information is crucial. Its performance also lends itself well to automated reporting, professional translations, and educational tools where clarity is essential.

Llama2:13b, with its richer narrative capabilities, is well-suited for content that benefits from detailed context such as journalism, content marketing, and storytelling. Its ability to deliver engaging, detailed content can significantly boost user engagement, even though its moderate resource demands might require use on more advanced hardware or via cloud services.

Despite its occasional lapses in accuracy due to being overly wordy, Phi shows how a tailored model can excel in creative tasks. With further fine-tuning, Phi could be optimized for niches like storytelling, interactive education, or rapid code prototyping—areas where creative depth is valued more than exact accuracy.

Moreover, my analysis suggests that using a mix of models in hybrid or ensemble systems could take advantage of each model's strengths (Gemma's clarity, Llama2's narrative detail, and Phi's creative flair), resulting in more versatile and robust applications across different domains.

Challenges Faced and Resolutions

During development, several challenges emerged that required thoughtful solutions to ensure optimal performance and accurate resource monitoring.

The first significant obstacle involved using llama3.3. Despite its advanced text generation skills, it was too demanding for my laptop's hardware. Its heavy computational needs led to very slow execution times, so I switched to llama2:13b. This change offered a more balanced approach, delivering strong performance without overloading my system.

Another significant challenge was accurately measuring memory usage for each model and task. My initial methods produced inexplicable negative values in megabytes, signaling flaws in my approach. To overcome this, I implemented a threading strategy. I set up a dedicated

communication thread (comm_thread) to handle subprocess communications in the background, while the main thread continuously tracked resource usage.

By wrapping the subprocess in a psutil.Process object, I could monitor detailed memory and CPU usage, including that of any child processes. I also added a warm-up phase for the CPU usage counter to create a reliable baseline before starting the measurements. This approach ensured accurate reporting of maximum resource consumption during each model's operation, successfully resolving my earlier issues.

Conclusion

This report has highlighted the strengths and limitations of three open-source large language models—phi, gemma:7b, and llama2:13b—across various language processing tasks. Gemma:7b consistently delivered precise, clear, and concise outputs, making it especially suitable for translation and code generation tasks. Llama2:13b provided a balanced mix of detailed narrative and clear responses, ideal for content-rich applications such as journalism and storytelling. While phi excelled in creative elaboration and efficiency for simpler tasks, its wordiness sometimes affected accuracy in more complex applications.

The practical insights gained from my work emphasize the importance of matching model capabilities with application needs and available hardware. By addressing resource management and performance tracking challenges with careful model choices and improved monitoring techniques, I am now better positioned to make informed decisions for deployment, fine-tuning, or even combining models to maximize efficiency and effectiveness in real-world use.