Executive Summary:

As the demand for AI-powered applications continues to grow, the trend towards developing smaller versions of large language models (LLMs) like *llama.cpp* is gaining momentum. My analysis revolving around *llama.cpp* revealed several key insights:

Analyzing multiple posts on *r/LocalLLaMA* as well as discussions and issues under the *llama.cpp* repository, I found that *llama.cpp* is sufficient for most developers and hobbyists, with both size and speed being adequate, albeit speed slightly less emphasized than size. The problem arises when using *llama.cpp* for multi-user services for production (i.e., hosting inference for a large audience), due to weaker throughput and precision of output. As a result, individuals prefer to use either *vLLM* or *TensorRT-LLM* for a large user-base. Nevertheless, the focus of the llama team should be to make the model's size even smaller while retaining as much accuracy as possible. This is because the size is a prime factor of consideration by users when opting for *llama.cpp*. So, to remain the top choice, the size should always be worked on.

Moreover, my analysis revealed that among speed, size, and accuracy issues, speed issues are the most explored, followed by accuracy issues, with size issues being the least explored. Similarly, speed issues had the highest number of unique users addressing them, followed by accuracy and size. Although this is a good sign that the model is being improved in the areas it needs work on, developers are encouraged to continue working on size-related issues to further optimize the model's performance and ensure its efficiency across various use cases.

Thousands of developers prefer to use *llama.cpp* for use cases related to managing or modifying confidential data. Although larger models have proven to be better in such cases, they cannot be used due to privacy constraints (e.g. of use cases: production code not reaching the internet, medical records, legal or unpublished documents). Thus, using local models' features is crucial for many organizations and affects millions of people worldwide. Additionally, thousands of developers also use it for use cases ranging from uninterrupted service from internet outages to unbiased conversations pertaining to specific domains. This is not possible with large models as they are hosted on the cloud and influenced by blackboxed algorithms, respectively. Thus, there exist various use cases of locally run models where larger models cannot be used.

Analyzing 1212 issues under the *llama.cpp* repository, I found that contributors tend to encounter as well as address bugs (confirmed and unconfirmed) and enhancements the most out of all other types of issues. Bugs are most commonly encountered in the following sequence (most to least): build-related bugs, performance bugs, server/backend bugs, model-related bugs, input/output bugs. For enhancements, the sequence is as follows (most to least): model architecture, performance, server functionality, build processes, compatibility & miscellaneous features. Focusing on addressing these areas in this specific order can significantly enhance the model's functionality, performance, and user experience to meet evolving standards and expectations.

For more quantitative details regarding the analysis as well as the raw data, please refer to the accompanying Google Sheets document.