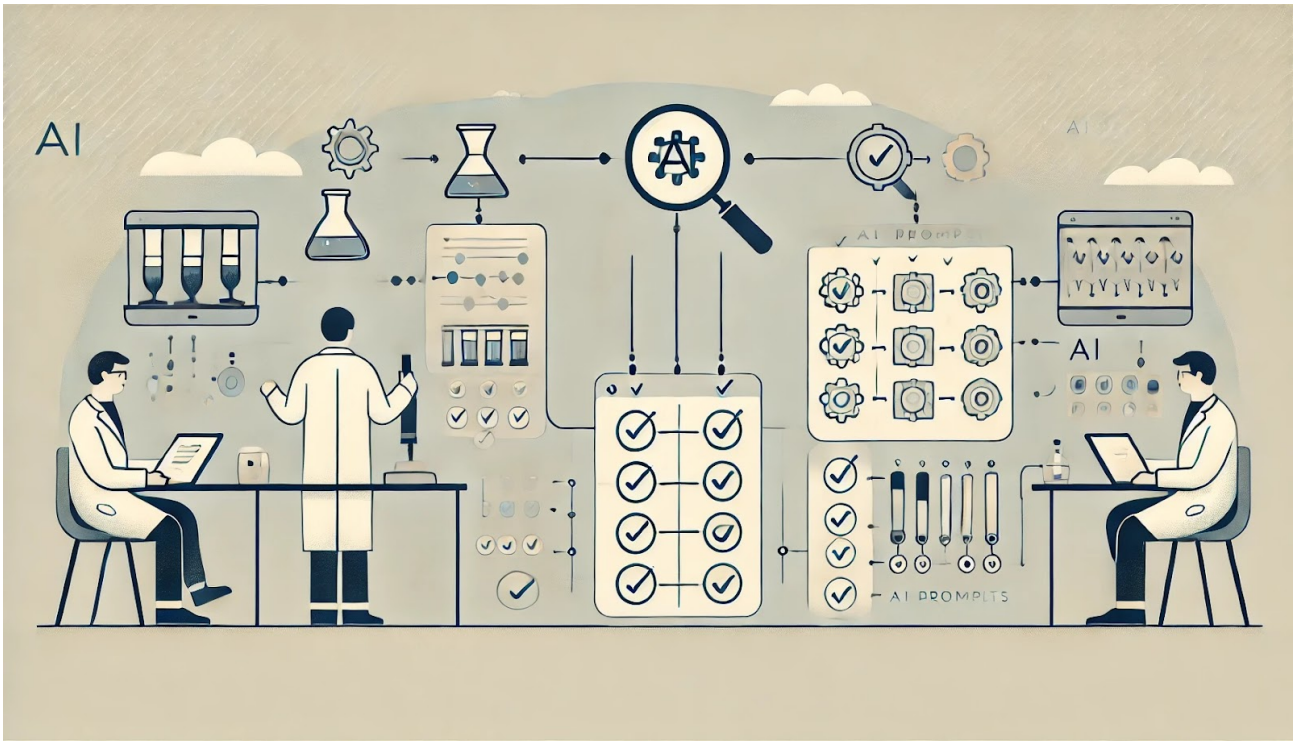
Mastering Prompt Engineering with Functional Testing: A Systematic Guide to Reliable LLM Outputs

Balancing precision and consistency in prompt optimisation

Ugo Pradere • 15 min read • 2025-03-15

s://towardsdatascience.com/mastering-prompt-engineering-with-functional-testing-a-systematic-guide-to-reliable-llm-output



How prompt evaluation with a systematic approach composed of algorithmic testing with input/output data fixtures can make prompt engineering for complex AI tasks more reliable.

Image created with DALL-E by the author

Creating efficient prompts for large language models often starts as a simple task... but it doesn't always stay that way. Initially, following basic best practices seems sufficient: adopt the persona of a specialist, write clear instructions, require a specific response format, and include a few relevant examples. But as requirements multiply, contradictions emerge, and even minor modifications can introduce unexpected failures. What was working perfectly in one prompt version suddenly breaks in another.

If you have ever felt trapped in an endless loop of trial and error, adjusting one rule only to see another one fail, you're not alone! The reality is that traditional prompt optimisation is clearly missing a structured, more scientific approach that will help to ensure reliability.

That's where functional testing for prompt engineering comes in! This approach, inspired by methodologies of experimental science, leverages automated input-output testing with multiple iterations and algorithmic scoring to turn prompt engineering into a measurable, data-driven

process.

No more guesswork. No more tedious manual validation. Just precise and repeatable results that allow you to fine-tune prompts efficiently and confidently.

In this article, we will explore a systematic approach for mastering prompt engineering, which ensures your LLM outputs will be efficient and reliable even for the most complex AI tasks.

Adding a large set of rules to a prompt can introduce partial contradictions between rules and lead to unexpected behaviors. This is especially true when following a pattern of starting with a general rule and following it with multiple exceptions or specific contradictory use cases. Adding specific rules and exceptions can cause conflict with the primary instruction and, potentially, with each other.

What might seem like a minor modification can unexpectedly impact other aspects of a prompt. This is not only true when adding a new rule but also when adding more detail to an existing rule, like changing the order of the set of instructions or even simply rewording it. These minor modifications can unintentionally change the way the model interprets and prioritizes the set of instructions.

The more details you add to a prompt, the greater the risk of unintended side effects. By trying to give too many details to every aspect of your task, you increase as well the risk of getting unexpected or deformed results. It is, therefore, essential to find the right balance between clarity and a high level of specification to maximise the relevance and consistency of the response. At a certain point, fixing one requirement can break two others, creating the frustrating feeling of taking one step forward and two steps backward in the optimization process.

Testing each change manually becomes quickly overwhelming. This is especially true when one needs to optimize prompts that must follow numerous competing specifications in a complex AI task. The process cannot simply be about modifying the prompt for one requirement after the other, hoping the previous instruction remains unaffected. It also can't be a system of selecting examples and checking them by hand. A better process with a more scientific approach should focus on ensuring repeatability and reliability in prompt optimization.

From laboratory to AI: Why testing LLM responses requires multiple iterations

Science teaches us to use replicates to ensure reproducibility and build confidence in an experiment's results. I have been working in academic research in chemistry and biology for more than a decade. In those fields, experimental results can be influenced by a multitude of factors that can lead to significant variability. To ensure the reliability and reproducibility of experimental results, scientists mostly employ a method known as triplicates. This approach involves conducting the same experiment three times under identical conditions, allowing the experimental variations to be of minor importance in the result. Statistical analysis (standard mean and deviation) conducted on the results, mostly in biology, allows the author of an experiment to determine the consistency of the results and strengthens confidence in the findings.

Just like in biology and chemistry, this approach can be used with LLMs to achieve reliable responses. With LLMs, the generation of responses is non-deterministic, meaning that the same

input can lead to different outputs due to the probabilistic nature of the models. This variability is challenging when evaluating the reliability and consistency of LLM outputs.

In the same way that biological/chemical experiments require triplicates to ensure reproducibility, testing LLMs should need multiple iterations to measure reproducibility. A single test by use case is, therefore, not sufficient because it does not represent the inherent variability in LLM responses. At least five iterations per use case allow for a better assessment. By analyzing the consistency of the responses across these iterations, one can better evaluate the reliability of the model and identify any potential issues or variation. It ensures that the output of the model is correctly controlled.

Multiply this across 10 to 15 different prompt requirements, and one can easily understand how, without a structured testing approach, we end up spending time in trial-and-error testing with no efficient way to assess quality.

A systematic approach: Functional testing for prompt optimization

To address these challenges, a structured evaluation methodology can be used to ease and accelerate the testing process and enhance the reliability of LLM outputs. This approach has several key components:

- **Data fixtures:** The approach's core center is the data fixtures, which are composed of predefined input-output pairs specifically created for prompt testing. These fixtures serve as controlled scenarios that represent the various requirements and edge cases the LLM must handle. By using a diverse set of fixtures, the performance of the prompt can be evaluated efficiently across different conditions.
- **Automated test validation:** This approach automates the validation of the requirements on a set of data fixtures by comparison between the expected outputs defined in the fixtures and the LLM response. This automated comparison ensures consistency and reduces the potential for human error or bias in the evaluation process. It allows for quick identification of discrepancies, enabling fine and efficient prompt adjustments.
- **Multiple iterations:** To assess the inherent variability of the LLM responses, this method runs multiple iterations for each test case. This iterative approach mimics the triplicate method used in biological/chemical experiments, providing a more robust dataset for analysis. By observing the consistency of responses across iterations, we can better assess the stability and reliability of the prompt.
- **Algorithmic scoring:** The results of each test case are scored algorithmically, reducing the need for long and laborious << human >> evaluation. This scoring system is designed to be objective and quantitative, providing clear metrics for assessing the performance of the prompt. And by focusing on measurable outcomes, we can make data-driven decisions to optimize the prompt effectively.

Step 1: Defining test data fixtures

Selecting or creating compatible test data fixtures is the most challenging step of our systematic approach because it requires careful thought. A fixture is not only any input-output pair; it must

be crafted meticulously to evaluate the most accurate as possible performance of the LLM for a specific requirement. This process requires:

1. A deep understanding of the task and the behavior of the model to make sure the selected examples effectively test the expected output while minimizing ambiguity or bias.

2. Foresight into how the evaluation will be conducted algorithmically during the test.

The quality of a fixture, therefore, depends not only on the good representativeness of the example but also on ensuring it can be efficiently tested algorithmically.

A fixture consists of:

• **Input example:** This is the data that will be given to the LLM for processing. It should represent a typical or edge-case scenario that the LLM is expected to handle. The input should be designed to cover a wide range of possible variations that the LLM might have to deal with in production.

• **Expected output:** This is the expected result that the LLM should produce with the provided input example. It is used for comparison with the actual LLM response output during validation.

Step 2: Running automated tests

Once the test data fixtures are defined, the next step involves the execution of automated tests to systematically evaluate the performance of the LLM response on the selected use cases. As previously stated, this process makes sure that the prompt is thoroughly tested against various scenarios, providing a reliable evaluation of its efficiency.

Execution process

1. **Multiple iterations:** For each test use case, the same input is provided to the LLM multiple times. A simple for loop in nb_iter with nb_iter = 5 and voila!

2. **Response comparison:** After each iteration, the LLM response is compared to the expected output of the fixture. This comparison checks whether the LLM has correctly processed the input according to the specified requirements.

3. **Scoring mechanism:** Each comparison results in a score:

%Pass (1): The response matches the expected output, indicating that the LLM has correctly handled the input.

%Fail (0): The response does not match the expected output, signaling a discrepancy that needs to be fixed.

4. **Final score calculation:** The scores from all iterations are aggregated to calculate the overall final score. This score represents the proportion of successful responses out of the total number of iterations. A high score, of course, indicates high prompt performance and reliability.

Example: Removing author signatures from an article

Let’s consider a simple scenario where an AI task is to remove author signatures from an article. To efficiently test this functionality, we need a set of fixtures that represent the various signature styles.

A dataset for this example could be:

Validation process:

- **Signature removal check:** The validation function checks if the signature is absent from the rewritten text. This is easily done programmatically by searching for the signature needle in the haystack output text.
- **Test failure criteria:** If the signature is still in the output, the test fails. This indicates that the LLM did not correctly remove the signature and that further adjustments to the prompt are required. If it is not, the test is passed.

The test evaluation provides a final score that allows a data-driven assessment of the prompt efficiency. If it scores perfectly, there is no need for further optimization. However, in most cases, you will not get a perfect score because either the consistency of the LLM response to a case is low (for example, 3 out of 5 iterations scored positive) or there are edge cases that the model struggles with (0 out of 5 iterations).

The feedback clearly indicates that there is still room for further improvements and it guides you to reexamine your prompt for ambiguous phrasing, conflicting rules, or edge cases. By continuously monitoring your score alongside your prompt modifications, you can incrementally reduce side effects, achieve greater efficiency and consistency, and approach an optimal and reliable output.

A perfect score is, however, not always achievable with the selected model. Changing the model might just fix the situation. If it doesn't, you know the limitations of your system and can take this fact into account in your workflow. With luck, this situation might just be solved in the near future with a simple model update.

Benefits of this method

- **Reliability of the result**: Running five to ten iterations provides reliable statistics on the performance of the prompt. A single test run may succeed once but not twice, and consistent success for multiple iterations indicates a robust and well-optimized prompt.
- **Efficiency of the process:** Unlike traditional scientific experiments that may take weeks or months to replicate, automated testing of LLMs can be carried out quickly. By setting a high number of iterations and waiting for a few minutes, we can obtain a high-quality, reproducible evaluation of the prompt efficiency.
- **Data-driven optimization:** The score obtained from these tests provides a data-driven assessment of the prompt's ability to meet requirements, allowing targeted improvements.
- **Side-by-side evaluation:** Structured testing allows for an easy assessment of prompt versions. By comparing the test results, one can identify the most effective set of parameters for the instructions (phrasing, order of instructions) to achieve the desired results.
- **Quick iterative improvement:** The ability to quickly test and iterate prompts is a real advantage to carefully construct the prompt ensuring that the previously validated requirements remain as the prompt increases in complexity and length.

By adopting this automated testing approach, we can systematically evaluate and enhance prompt performance, ensuring consistent and reliable outputs with the desired requirements. This

method saves time and provides a robust analytical tool for continuous prompt optimization.

Systematic prompt testing: Beyond prompt optimization

Implementing a systematic prompt testing approach offers more advantages than just the initial prompt optimization. This methodology is valuable for other aspects of AI tasks:

1. **Model comparison:**

- Provider evaluation: This approach allows the efficient comparison of different LLM providers, such as ChatGPT, Claude, Gemini, Mistral, etc., on the same tasks. It becomes easy to evaluate which model performs the best for their specific needs.
- Model version: State-of-the-art model versions are not always necessary when a prompt is well-optimized, even for complex AI tasks. A lightweight, faster version can provide the same results with a faster response. This approach allows a side-by-side comparison of the different versions of a model, such as Gemini 1.5 flash vs. 1.5 pro vs. 2.0 flash or ChatGPT 3.5 vs. 4o mini vs. 4o, and allows the data-driven selection of the model version.

2. **Version upgrades:**

- Compatibility verification: When a new model version is released, systematic prompt testing helps validate if the upgrade maintains or improves the prompt performance. This is crucial for ensuring that updates do not unintentionally break the functionality.
- Seamless Transitions: By identifying key requirements and testing them, this method can facilitate better transitions to new model versions, allowing fast adjustment when necessary in order to maintain high-quality outputs.

3. **Cost optimization:**

- Performance-to-cost ratio: Systematic prompt testing helps in choosing the best cost-effective model based on the performance-to-cost ratio. We can efficiently identify the most efficient option between performance and operational costs to get the best return on LLM costs.

Overcoming the challenges

The biggest challenge of this approach is the preparation of the set of test data fixtures, but the effort invested in this process will pay off significantly as time passes. Well-prepared fixtures save considerable debugging time and enhance model efficiency and reliability by providing a robust foundation for evaluating the LLM response. The initial investment is quickly returned by improved efficiency and effectiveness in LLM development and deployment.

Quick pros and cons

Key advantages:

- **Continuous improvement:** The ability to add more requirements over time while ensuring existing functionality stays intact is a significant advantage. This allows for the evolution of the AI task in response to new requirements, ensuring that the system remains up-to-date and efficient.
- **Better maintenance:** This approach enables the easy validation of prompt performance with

LLM updates. This is crucial for maintaining high standards of quality and reliability, as updates can sometimes introduce unintended changes in behavior.

- **More flexibility**: With a set of quality control tests, switching LLM providers becomes more straightforward. This flexibility allows us to adapt to changes in the market or technological advancements, ensuring we can always use the best tool for the job.
- **Cost optimization:** Data-driven evaluations enable better decisions on performance-to-cost ratio. By understanding the performance gains of different models, we can choose the most cost-effective solution that meets the needs.
- **Time savings:** Systematic evaluations provide quick feedback, reducing the need for manual testing. This efficiency allows to quickly iterate on prompt improvement and optimization, accelerating the development process.

Challenges

- **Initial time investment:** Creating test fixtures and evaluation functions can require a significant investment of time.
- **Defining measurable validation criteria:** Not all AI tasks have clear pass/fail conditions. Defining measurable criteria for validation can sometimes be challenging, especially for tasks that involve subjective or nuanced outputs. This requires careful consideration and may involve a difficult selection of the evaluation metrics.
- **Cost associated with multiple tests:** Multiple test use cases associated with 5 to 10 iterations can generate a high number of LLM requests for a single test automation. But if the cost of a single LLM call is neglectable, as it is in most cases for text input/output calls, the overall cost of a test remains minimal.

Conclusion: When should you implement this approach?

Implementing this systematic testing approach is, of course, not always necessary, especially for simple tasks. However, for complex AI workflows in which precision and reliability are critical, this approach becomes highly valuable by offering a systematic way to assess and optimize prompt performance, preventing endless cycles of trial and error.

By incorporating functional testing principles into prompt engineering, we transform a traditionally subjective and fragile process into one that is measurable, scalable, and robust. Not only does it enhance the reliability of LLM outputs, it helps achieve continuous improvement and efficient resource allocation.

The decision to implement systematic prompt testing should be based on the complexity of your project. For scenarios demanding high precision and consistency, investing the time to set up this methodology can significantly improve outcomes and speed up the development processes. However, for simpler tasks, a more classical, lightweight approach may be sufficient. The key is to balance the need for rigor with practical considerations, ensuring that your testing strategy aligns with your goals and constraints.

Thanks for reading!

Written By

Share This Article

- Share on Facebook
- Share on LinkedIn
- Share on X

Towards Data Science is a community publication. Submit your insights to reach our global audience and earn through the TDS Author Payment Program.

Other mentions by Author

- towardsdatascience.com | Related Articles
- towardsdatascience.com | Deep Dive into LlaMA 3 by Hand ✍️
- towardsdatascience.com | Deep Dive into Transformers by Hand ✍️
- towardsdatascience.com | Deep Dive into Sora's Diffusion Transformer (DiT) by Hand ✍️
- towardsdatascience.com | UniFliXsg: AI-Powered Undergraduate Program Recommendations for Singapore Universities
- towardsdatascience.com | Beware of Unreliable Data in Model Evaluation: A LLM Prompt Selection case study with Flan-T5
- towardsdatascience.com | Semantic Search Engine for Emojis in 50+ Languages Using AI 😊🌍🚀