# Pushing Aside the Bench for the Mark

So, you need to choose an LLM for your product but where do you start? It is an expansive topic, but this article starts at the very basics. By example of choosing between five Anthropic models, it will prove that you cannot simply depend on benchmarks or marketing. On reaching the end, you will understand why it is crucial to measure and compare LLMs against your own use case.

## Benchmarks Are General

When you go to any of the popular benchmark leaderboards it is like you are look at a list of “here are the fastest and most capable athletes.” A general athlete is not what you need, your use case needs a swimmer. Of course they must be capable, but in what exactly: the quality of their stroke, resilience under pressure, or something else? They need to be fast too but fast in what exactly, 100m backstroke or getting to the pool? It turns out you need an open water swimmer capable of swimming the English Channel, fast.

This is the essence as to why you cannot simply make your choice by popular benchmarks. They rank in general, but your use case is never general. Added to this, it is possible to game benchmarks during model training. The waters could be murky.

The section below will setup a fictious (and very simple) example of having to choose between the five Anthropic models. The purpose is to set a way of thinking first and avoid getting lost.

## Choosing An Anthropic Model

Assume that for my use case I need to decide between the following Anthropic models:



Figure 1: The Anthropic Model Families and the models (or snapshots) we will be testing (blue).

Looking at the above Claude Sonnet seems to be the easiest choice. And of the two snapshots available, I am sure (Oct 2024) must be even better than (Jun 2024). It looks like the best choice, the most popular leaderboards point to this too, but is it?

It depends entirely on your use case. The way to test this is by evaluating or “marking” prompt responses. Not just generic prompts but a set of prompts with representative coverage of your use case. The more coverage you have and the more relevant the prompts, the more confident you will be in choosing a model or justifying a move to the latest version (snapshot).

For this explanation, I will make evaluating my use case (unrealistically) simple by “marking” model responses on just two prompts:

Speed Test (prompt 1):

* **Explain photosynthesis in a concise paragraph**. (max\_tokens: 400)

Capability Test (prompt 2):

* **What is the geometric monthly fecal coliform mean of a distribution system with the following FC counts: 24, 15, 7, 16, 31 and 23? The result will be inputted into a NPDES DMR, therefore, round to the nearest whole number.**

 As LLMs deal in probabilities, the same prompt can have different responses and of course speed can vary too. For each model, I chose to run each prompt 50 times sequentially for each model (5 x 2 x 50 = 500 runs).

The results presented were gathered from a full test run done on the 4th of February 2025 at 15:02 UTC. The 500 prompts and their responses finished 16 minutes later.

The sections that follow visually step through these test results for speed, capability, and cost.

## Speed Test Results

Like measuring typing speed in words per minute, model speed is measured in tokens per second. This measures the total time from sending your prompt request until receiving the response - including both the model's "thinking" time (inference) and the internet journey time. This is the speed your product will experience. See Appendix 1 for the code snippet used.

Looking at Figure 1 (above), perhaps you are assuming the same as I did:

* Haiku models will be the fastest
* Haiku 3.5 will be faster than Haiku 3.0 (newer is better?)

But the results proved my assumptions wrong:



[Figure 2: Haiku 3 is fastest, but Sonnet 3.5 is faster than Haiku 3.5]

Figure 2 above shows “speed over time” for the speed test prompt that was sent 50 times to each model, sequentially.

What is most surprising is that the newer Haiku 3.5 was slower than Sonnet 3.5. Also, compared to Opus how unstable the model speeds were.

Looking at the same result data but in an easier-to-compare format:



[Figure 3: Speed Individual Dots plot]

Figure 3 makes the speed differences more apparent. Haiku 3 was the fastest but also the most unstable (notice the vertical spread). Here Haiku 3.5 does appear slower than Sonnet 3.5 (but may not be statistically significant).



[Figure 4: Speed Summary Table]

Figure 4 is a full summary of the speed test results. I have chosen the median to compare models because of outliers.

Now that you understand the easiest, testing and comparing speed, we will move on to testing model capability.

## Capability Test Results

Capability is easy to measure when a definitive response is expected, like a math solution or classification. Most frequently you will need to compare textual responses. In this case there are methods, both manual and automated, that can be used to evaluate or “mark” textual responses for comparison. In my opinion, the most promising is using an independent LLM to judge the responses; it can be automated and is more consistent than most people.

The capability test prompt I used (Figure x below) has an exact solution of “18”. You can see that even Opus “the most intelligent model” got 6% of its answers wrong:



Figure: Summary of Capability Results (including all incorrect answers)

Figure: Unexpectedly the older “Haiku 3” model outperforms its newer model and Sonnet!

Like speed, the results for my “use case” are in contradiction to the benchmarks. Sonnet should be outperforming Haiku. And like with speed, both Haiku and Sonnet degraded.

It is now time to look at what appears to be easier than testing speed, comparing costs.

## Cost Calculations

I used to work in manufacturing whose product was sold to mines, shipping companies etc. Buyers unconnected to operations on the ground would opt for cheaper suppliers. The cheaper products would fail sooner or require more maintenance. The client would suffer far more costly repercussions but uncaptured by a quarterly financial report. Nobody can blame the buyers as everyone likes the look of a low price.

In the LLM world, most providers price models in “cost per token”. However, it is a bit more complex than simply comparing token costs because a token is not standardised. For example, Anthropic may tokenise a given sentence as “12 anthropic tokens”, where Open AI may tokenize it as “14 Open AI tokens”.

The trick is to first look beyond the quoted price and delve into deeper cost questions. Do you get accurate responses efficiently or does it require multiple back and forth? If the model is hosted, does it get congested and what will this cost your product? Is the API easy and robust to use? Is there solid documentation on prompt engineering practices for the model? Privacy, how much is that going to cost if it goes wrong? Only then take the time to standardise and compare token costs.

Anthropic prices their models in dollars per million tokens ($/MTOK) with input tokens costing more than output tokens. You can see where this is going, which price is more important to your use-case. And, is it possible to cache a prompt and do you pay for it?

Current Anthropic pricing:

Figure: Current Anthropic pricing per million tokens ($US).

It cost $1.85 to run in the entire test set of 500 prompt requests ( 2 prompts x 5 models x 50 runs per model). In the detailed breakdown below, the speed tests cost more because a “concise paragraph” was asked for. Whereas the capability prompt instructed “respond only with a number”.



Figure: Cost breakdown per model for running all speed and capability tests.

It is obvious now that whilst developing your product at scale, prompts can be optimised for cost efficiency. But first, ensure you look at the whole cost picture in choosing your model.

## Results Summary

The below figure summarises all the results into one diagram. On the x-axis are the median speed test results (running prompt 1 against each model 50 times). On the y-axis are the capability test results (running prompt 2 against each model 50 times). The size of each circle is the total cost, for both speed and capability tests combined.



Figure: All Results: Capability vs Speed vs Total Cost

Comparing the benchmarks and Anthropic’s model positioning, when measured against my fictitious use case are the biggest contradictions and remarkable surprises:

Speed

* "Haiku 3.5 our fastest model" — it was not the fastest, Sonnet 3.5 appeared faster. But were dwarfed by the older Haiku 3.
* All newer models and snapshots degraded in speed.

Capability:

* “Sonnet 3.5 our best combination of performance and speed” — it was not for me, Haiku 3 outperformed Sonnet 3.5.
* All newer models and snapshots degraded in capability.

Cost — most surprising of all:

* Haiku 3 is the fastest with the closest capability to Opus — yet it is the cheapest!

Again though, please keep in mind that two little test prompts are not enough to draw a conclusion, but neither are general benchmarks.

What these results prove is that you can expect to find surprises: you need to evaluate against your own use case by testing with use case relevant prompts.

## Future Proofing

You certainly will choose the wrong model. For no other reason that things change rapidly: think DeepSeek-R1 that arrived in January. If the benchmarks are to be believed, it is comparable to Claude Sonnet but at $2.19/MTOK instead of $15! That would make your December decision wrong (ceteris paribus).

The perfect choice does not matter nearly as much as your ability to adjust from a “wrong” decision. What matters is your ability to evaluate a new model (or model update) against your use case, then to make it easy to switch, confidently.

So deeply consider (A) and (B) below:

### (A) For evaluating new models:

As your product grows, so you add more prompt evaluations for the prompts used by your product. Now not only do you have the evaluations you started off with in choosing your model, but a growing evaluation set representative of your use case! Strive for these tests to be automated. With a little work, you can run this evaluation set against other models for comparison.

At DeepSeek-R1’s $2.19/MTOK, it is almost too difficult to believe that it could possibly be as good as Claude 3.5 Sonnet. You don’t know until you test against your own use case.

### (B) Making a model switch easier:

From the start, at the very least allow your Engineers to invest in an abstraction layer. Get into a room and explain future proofing, explain the story of DeepSeek-R1 where the cost is 1/7th of Sonnet. Go beyond price, as a team what will you do when a new model comes out that everyone and the leaderboards are saying is better? What will it take to confidently and easily switch? How are we going to easily run our prompt evaluation suite against a new model?

The answer you are likely to get is at the very least, is an abstraction layer that sits between your application’s code and LLM API’s. You do not need a perfect interchangeable model solution, but there are things that if done with discipline from the start will keep the product rapidly flexible.

## Conclusion

In the real world you cannot use only two prompts to compare models you must strive for good coverage. I think of my product in major segments of LLM tasks. I emphasise more segments than others, but within each I strive for coverage by creating relevant test prompts that inject real messy data. When it comes to costs, take the time to research the entire cost picture.

Once the model is chosen the journey has only started. As your product grows and evolves so too must your prompt evaluation test suite. Today’s decision is likely to be wrong tomorrow.

Benchmarks and marketing are general, your use case is not. Take the time to automate your prompt evaluations. It is an investment that surpasses a quarterly report. You may need to defend this investment but persist. You are giving the best chance of having your product flourish in a rapidly changing landscape, and to perform well along the way.

And yes, even the product manager must develop a solid understanding of prompting and statistical analysis.

## Appendix



Notebook that runs the test cases: