

Short answer: 78 version pairs. That is the useful headline—but it needs a qualifier. The count describes version pairs with measurements in the index, not every model release announced by a lab, every model available to users, or every possible comparison between two versions.

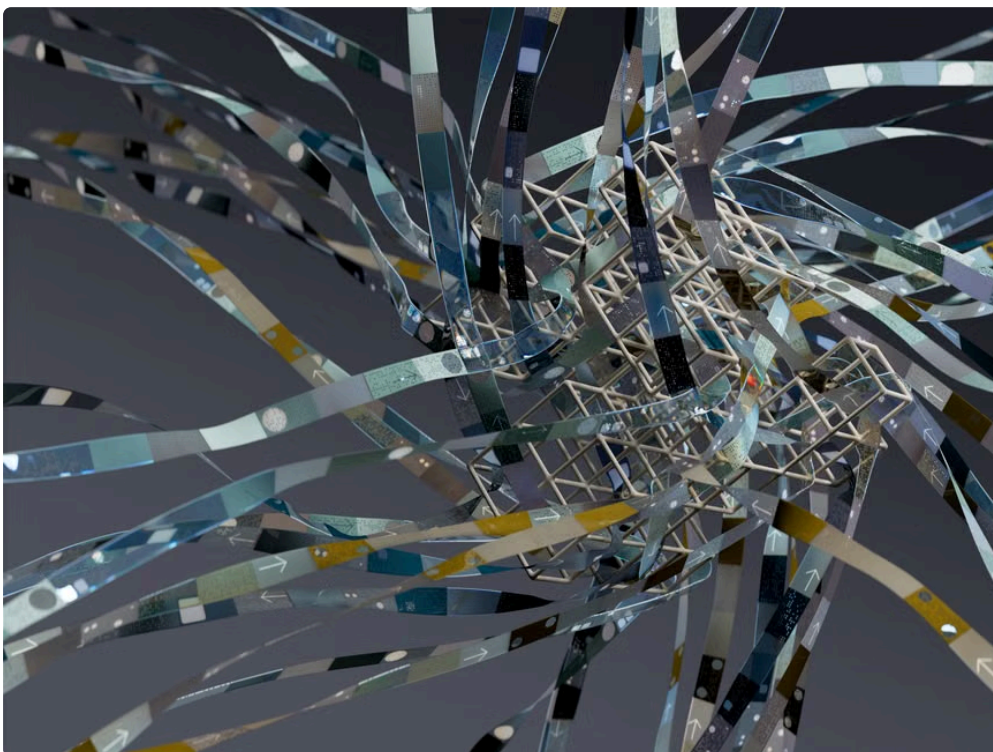
That distinction matters when teams use an index to decide whether to upgrade. A release can be announced without appearing in the underlying measurements. And even when two versions are measured, a score does not automatically tell you whether the newer one is better for your prompts, your budget, or your users.

This post explains what the 78-pair count can—and cannot—tell you, why verified release dates matter, and how to use blind-vote preference data alongside launch benchmarks. It also covers practical checks for regressions and the release cadence that makes version tracking harder than it looks.

What “78 version pairs” means

A version pair is a comparison between two identifiable releases—for example, an earlier model version and a later one. In this index, **78 version pairs** have measurements available for comparison. That is a count of measured pairs, not a count of all models, all releases, or all possible pairings.

The difference is easy to miss. If a lab has released five versions, there are ten possible pairings among them. But an index may include only some of those comparisons: perhaps adjacent releases, versions with enough evaluation data, or pairs that can be matched consistently across a selected source. The total depends on the index’s inclusion rules.



So read “78” as a coverage statement: these are the pairs the index can evaluate under its stated method. It should not be read as proof that all 78 were tested in the same way, on the same date, or for the same tasks unless the index explicitly says so.

What the count refers to What it does not automatically mean Version pairs with measurements represented in the index Every release announced by every lab Comparisons available under the index’s inclusion rules Every

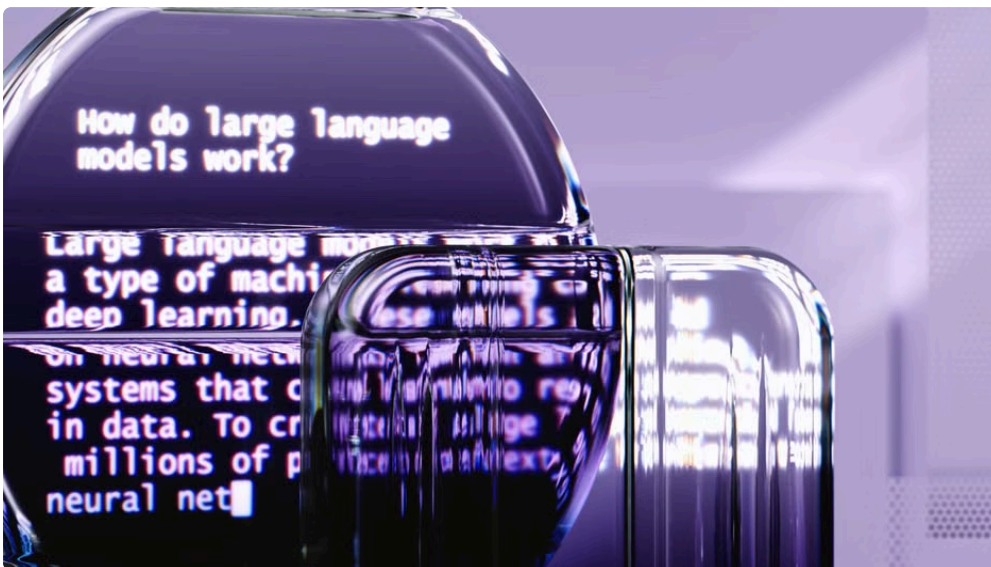
possible pairing of versions A starting point for comparing measured releases A guarantee that newer versions are better for every workload

For Suprmind Data Lab, or any team publishing a version index, the practical next step is to make the counting rule visible. Does a pair require both versions to appear in one leaderboard? Are measurements taken from a single snapshot? Are models grouped by API name, public alias, or dated version identifier? Those details determine what the number means.

Verified release dates are not announcement dates

Release tracking has at least two clocks. The first is the **announced date**: when a company says a model is coming, previews it, or opens a waitlist. The second is the **verified release date**: when there is evidence that the model was actually available in the relevant product or API.

Those dates can differ. A model may be announced in a blog post, rolled out gradually, and become broadly accessible later. An API model may have a dated identifier before a consumer product exposes it. A benchmark may test a preview build that is not the same version customers later use.



For an index, the safest practice is to record both dates and label them clearly. “Announced” describes what the lab communicated. “Verified” describes what the evidence supports. The evidence might be a dated API reference, release note, model card, product availability notice, or a dataset observation. If the source only supports an announcement, do not silently turn it into a confirmed release date.

- **Announcement**: the lab publicly describes or previews the model.
- **Verified availability**: a dated source shows that users could access the model in the relevant channel.
- **Measurement date**: the date the benchmark or preference data was collected or recorded.

Keeping these separate helps explain why a release can be real but not yet measured in an index. It also prevents a common timeline mistake: comparing a model’s launch announcement with an older measurement for its predecessor and implying that the comparison reflects a like-for-like test.

Blind-vote preference is not a launch benchmark

One useful source for preference comparisons is the LMArena text leaderboard, formerly LMSYS Chatbot Arena. In a blind-vote setting, people compare model responses without being told which model produced each one, then

select the response they prefer. The leaderboard can help answer a specific question: *Which answer did voters prefer in the comparisons represented by this data?*

That is different from a launch benchmark. A launch benchmark may test a fixed set of questions, a particular capability, or a lab-selected evaluation suite. It can offer useful evidence, but it is not the same signal as human preference from blinded comparisons. The two methods answer different questions and have different sources of bias.

Blind voting is not a universal quality test. Results can shift with the prompt mix, the voters, the number of comparisons, model availability, and changes to the voting interface or sampling process. A preference score also does not directly measure factual accuracy, latency, tool reliability, refusal behavior, or performance on your private data.

The LMArena dataset is available on Hugging Face as `lmarena-ai/leaderboard-dataset`. When using it to check a version pair, inspect the dataset documentation and fields rather than relying on a chart alone. Confirm which model identifiers are present, what observations the comparison uses, and whether the relevant versions can be distinguished cleanly.

A practical blind-vote check

1. Confirm the exact model identifiers. Similar display names may refer to different versions or aliases.
2. Check the data snapshot and observation counts for both sides of the comparison.
3. Review the leaderboard's settings, including whether style control is applied.
4. Read the result as a preference signal, not as a complete measure of model quality.
5. Compare the result with your own representative prompts before making an upgrade decision.

Style control deserves particular attention. Response length, formatting, tone, and willingness to explain can influence preferences. A style-controlled view can help separate some presentation effects from other differences, but it does not remove every source of bias. If a model wins mainly because it is more verbose or polished, **latest AI models released** that may or may not be valuable in your product.

Use the index to make upgrade decisions—not to outsource them

The 78 version pairs are most useful as a shortlist for investigation. They show where the index has evidence. They do not tell you that every measured change will matter in production, or that a model with a higher preference result will be cheaper, faster, or more reliable.

Before upgrading, compare the new version with the one you actually run. Use the same prompts, tools, system instructions, output limits, and scoring rules. Include routine requests as well as edge cases. If your system depends on structured output or function calls, test those directly; a general leaderboard result cannot stand in for them.

Track at least four outcomes: task success, failure or regression rate, latency, and cost. For a customer-facing assistant, add measures that reflect your product, such as escalation rate or user correction. Keep the previous version available long enough to roll back if the new one regresses on a critical path.

Look for regressions, not just wins

A version can improve on one task and get worse on another. It might follow instructions better but produce longer answers. It might be stronger at coding while becoming less reliable at a constrained JSON format. A single blended score can hide those trade-offs.

- Test the top tasks by volume, not just the most impressive demo prompt.
- Include known failure cases and prompts that have caused regressions before.
- Compare outputs blind where practical, so evaluators are less influenced by model names.
- Record the exact model ID, test date, prompt set, and scoring method.
- Repeat tests when the provider’s model alias may point to a changing backend.

This is where measured releases are more valuable than release headlines. A release note can tell you what a lab intends to change. A repeatable evaluation tells you what changed in your daily use.

Price claims need a label and a source

Price is part of the upgrade decision, but comparisons need the same care as quality claims. For example, a secondary source has reported that GPT-5.2 costs **about 40% more than GPT-5.1**. Treat that as a **lab claim reported by a secondary source**, not as a universal cost difference verified for every usage pattern.

Before applying that figure, check the source’s publication date, the pricing units, and which token categories or product tiers it compares. Actual bills also depend on input and output volume, caching, batch discounts, and the model mix in your application. A percentage increase in a listed rate does not necessarily translate into the same percentage increase in your total spend.

For a fair comparison, run a representative workload through both versions and calculate cost per successfully completed task. That puts price next to quality and failure rate, where it belongs. A cheaper model that needs retries may cost more in practice; a more expensive model may be worth it for a high-value workflow, but only if your tests show a meaningful gain.

Release cadence across labs changes the measurement problem

OpenAI, Anthropic, and other labs do not release models on one shared schedule. They may ship major models, smaller variants, previews, API aliases, and product-specific updates at different intervals. Some updates are easy to identify by a dated model name. Others may change behind a stable alias or roll out gradually.

That uneven cadence makes a simple chronological leaderboard tempting—and sometimes misleading. A fast release cycle can produce several announced versions before there is enough comparable data for a measured pair. A slower cycle can leave a model with more observations simply because it has been available longer. More observations can improve confidence, but they do not guarantee that a result transfers to your use case.

For each release, keep a compact record: announced date, verified availability date, model identifier, measurement source, measurement date, and any known limitations. Then distinguish “not measured yet” from “measured and unchanged.” Those are different findings. The first is missing evidence; the second is evidence of no observed difference under a particular method.

What to take away from the 78-pair count

78 version pairs is a meaningful coverage figure because it tells readers how many comparisons have measurements in the index. It is not a universal count of model improvements, and it does not tell you how many releases the labs announced. The value of the number depends on whether each pair is identifiable, dated, and measured in a way readers can inspect.

Use verified release dates to anchor the timeline. Use blind-vote preference as one signal, not the whole scorecard. Check the LMArena text leaderboard with style control and inspect the [lmarena-ai/leaderboard-dataset](#) when you need to understand what sits behind a comparison. Then test the versions on your own workload, including regressions, latency, and cost.

The useful question is not simply whether a model is “better.” It is: better at what, measured when, compared with which verified version, and at what cost? An index that makes those answers visible turns 78 pairs from a headline into a practical tool for upgrade decisions.