How to optimize skill descriptions

An incomplete description means the skill won't activate when needed; an overly general description means it will activate when not needed.

How can I improve my skill description so that it's reliably triggered when the appropriate request is made?

A skill is only useful when it's activated. The description field at the beginning of the SKILL.md file is the primary mechanism agents use to decide whether or not to load that skill for a given task. An incomplete description means the skill won't be activated when needed; a description that's too general means it will be activated when not needed.

This guide will show you how to systematically check and improve your skill descriptions to ensure accuracy when activated.

How skill activation works

Agents use a gradual disclosure method to manage context. Upon startup, they only load `name`the `description`available skill descriptions—just enough to determine when a skill might be suitable. When the user's task matches the description, the agent reads the entire SKILL.md file into the context and follows its instructions.

This means the description is entirely responsible for activation. If the description doesn't convey when the skill is useful, the agent won't know to use it.

An important point to note: Agents typically only invoke skills for tasks requiring knowledge or capabilities beyond their processing capacity. A simple, one-step request like 'read this PDF file' might not trigger the PDF reading skill even if the description perfectly matches, because the agent can handle it using basic tools. Tasks involving specialized knowledge—an unfamiliar API, a field-specific workflow, or an uncommon format—are where a well-written description can make all the difference.

Write effective descriptions

Before attempting the experiment, it's helpful to know what a good description looks like. There are a few guidelines:

1. **Use imperative sentences** . Format the description as a guide for the agent: 'Use this skill when…' instead of 'this skill performs…' The agent is deciding whether or not to act, so let it know when to act.

2. **Focus on the user's intent, not the implementation** . Describe what the user is trying to achieve, not the skill's internal mechanism. The agent will match the user's request.
3. **A compelling description should be provided** . Clearly list the contexts in which the skill applies, including instances where the user doesn't directly specify a domain: 'even if they don't explicitly mention 'CSV' or 'analysis".
4. **Keep the content concise** . A few sentences to a short paragraph is usually appropriate—long enough to cover the scope of the skill, short enough not to inflate the agent's context across multiple skills. The specification imposes a hard limit of 1024 characters.

Design trigger evaluation queries

To test for activation, you need a set of evaluation queries—actual user prompts labeled as indicating whether they should trigger your skill.

```
[ { "query": "Tôi có m?t b?ng tính trong ~/data/q4_results.xlsx v?
i doanh thu ? c?t C và chi phí ? c?t D — b?n có th? thêm c?t t? su?t l?
i nhu?n và highlight b?t k? kho?n nào d??
i 10% không?", "should_trigger": true }, { "query": "cách nhanh nh?t ??
 chuy?n ??i file JSON này sang YAML là gì", "should_trigger": false } ]
```

Aim for around 20 queries: 8-10 queries should be triggered and 8-10 queries should not be triggered.

The queries should trigger

These queries check if the description accurately captures the scope of the skill. Let's modify them in a few aspects:

1. **Style of expression** : Some are formal, some are informal, and some contain spelling errors or abbreviations.
2. **Clarity** : Some queries directly state the domain of the skill ('analyze this CSV file'), while others describe the need without naming it ('my boss wants a chart from this data file').
3. **Details** : Combine concise prompts with more contextual prompts – a short prompt 'analyze my sales CSV file and create a chart' along with a longer message that includes the file path, column names, and context.
4. **Complexity** : Vary the number of steps and decision points. Include single-step tasks alongside multi-step workflows to test whether employees can recognize the relevance of the skill when the task it addresses is part of a larger sequence.

The queries that should trigger most effectively are those where the skill would be useful but the connection isn't clear from the query alone. These are the cases where descriptive wording makes all the difference – if the query asks precisely what the skill does, any reasonable description will trigger.

The queries should not be triggered.

The most valuable negative test cases are the approximate ones – queries that share keywords or concepts with your skill but actually require something different. These cases check if the description is accurate, not just general.

For CSV analysis skills, some slightly negative examples would be:

1. "Write a Fibonacci function" - obviously irrelevant, no testing involved.
2. "What's the weather like today?" - no duplicate keywords, too easy.

Extremely negative examples:

1. "I need to update the formulas in my Excel budget spreadsheet" - sharing the concepts of "spreadsheet" and "data," but needing to edit Excel, not analyze CSV.
2. "Can you write a Python script that reads CSV files and uploads each row to our PostgreSQL database?" - related to CSV files, but the task is database ETL, not analysis.

Tips for being more realistic

Real user prompts contain context that generic test queries lack. This includes:

1. File path (~/Downloads/report_final_v2.xlsx)
2. Personal context ("The manager asked me.")
3. Specific details (column name, company name, data value)
4. Common language, abbreviations, and sometimes spelling errors.

Check if the description is activated.

Basic approach: Run each query through your agent with the skill installed and observe whether the agent calls it. Ensure the skill is registered and can be found by your agent – ??how this works varies depending on the client (e.g., skill directory, configuration file, or CLI flag).

Most agent clients provide some form of observation—execution logs, tool call history, or detailed output—that allows you to see which skills were referenced during runtime. Check the client documentation for details. A skill is triggered if the agent loads the skill's SKILL.md file; it is not triggered if the agent continues without referencing that file.

A query is considered 'successful' if:

1. `should_trigger`is `true`and the skill is activated, or
2. `should_trigger`That means `false`the skill is not activated.

Run multiple times

The model's behavior is non-deterministic – the same query might trigger the skill on one run but not on the next. Run each query multiple times (3 is a reasonable starting point) and calculate the trigger rate: the ratio of runs in which the skill is triggered.

A query that should trigger will succeed if its trigger rate is higher than the threshold (0.5 is a reasonable default value). A query that should not trigger will succeed if its trigger rate is lower than that threshold.

With 20 queries, each running 3 times, that's a total of 60 triggers. You'll need to write a script for this. Here's the general structure - replace the trigger and claude detection logic in `check_triggered` with whatever the client agent provides:

```
#!/bin/bash QUERIES_FILE="${1:?Usage: $0 }" SKILL_NAME="my-skill" RUNS=3 # Ví d
? này s? d?ng ??u ra JSON c?a Claude Code ?? ki?m tra các l?nh g?i công c?
 Skill. # Thay th? hàm này b?ng logic phát hi?n dành cho agent client c?a b
?n. # Nên tr? v? 0 (thành công) n?u skill ???c kích ho?t, ng??c l?i tr? v?
 1. check_triggered() { local query="$1" claude -p "$query" --output-format json
```

? If your agent client supports this feature, you can stop the process early when the results are clear – the agent has either consulted the skill or started working without that skill. This can significantly reduce the time and cost of running the entire evaluation dataset.

Avoid overfitting by splitting training/testing sessions.

If you optimize descriptions for all your queries, you risk overfitting—creating a description that works well with certain expressions but fails with new ones.

The solution is to split your query set:

1. **Training set (~60%)** : The queries you use to identify errors and guide improvements.
2. **Verification set (~40%)** : Queries you keep separate and use only to check if the improvements are generalizable.

Make sure both sets contain a balanced ratio of queries that should trigger and those that shouldn't—don't accidentally put all the positive results in one set. Randomly mix and maintain the split ratio throughout the iterations so you're comparing similar things. If you're using a script like the one above, you can split your queries into two files—train_queries.json and validation_queries.json—and run the script on each file separately.

Optimization loop

1. Evaluate the current description on both the training and validation sets. The training results will guide your changes; the validation results will tell you whether those changes are generalizable.

2. Identify errors in the training set: Which queries should have been triggered but weren't? Which queries shouldn't have been triggered but were?

1. Use only the errors in the training set to guide your changes – whether you're modifying the description yourself or using an LLM , remove the validation set's results from the process.

3. Revise the description. Focus on generality:

1. If the queries that should have triggered the skill fail, the description might be too narrow. Expand the scope or add context about when this skill would be useful.
2. If queries that shouldn't trigger cause errors, the description might be too broad. Add more detail about what this skill doesn't do, or clarify the boundaries between this skill and adjacent capabilities. Avoid adding specific keywords from failing queries—that's overfitting. Instead, find the general category or

concept those queries represent and focus on that.
3. If you're having trouble after a few attempts, try a different approach to structuring the description instead of just minor tweaks. A different wording or sentence structure might be far more effective than simple adjustments.
4. Check if the description is within the 1024-character limit – descriptions tend to get longer during the optimization process.

4. Repeat steps 1-3 until all queries in the training set are passed or you no longer see significant improvement.

5. Choose the best version based on the test pass rate – the percentage of queries in the test set that passed. Note that the best description may not be the final description you create; an earlier version might have a higher test pass rate than later versions that were too well-fitted to the training set.

Five attempts are usually sufficient. If performance doesn't improve, the problem might lie with the queries themselves (too easy, too difficult, or poorly labeled) rather than the description.

Skill -creator automates this entire process: It splits the evaluation dataset, evaluates activation rates in parallel, suggests descriptive improvements using Claude, and generates live HTML reports that you can monitor as it runs.

Apply the results

After you have selected the best description:

1. Update the description field in the frontmatter section of SKILL.md.
2. Verify that the description does not exceed the 1024-character limit.
3. Verify that the trigger descriptor works as expected. Try a few manual prompts for a quick check. For a more thorough test, write 5-10 new queries (a mix of queries that should trigger and those that shouldn't) and run them through the evaluation script – since these queries haven't been part of the optimization process, they'll give you an honest check of whether the descriptor is generalizable.

Before and after comparison:

```
# Tr??c description: X?
 lý các file CSV. # Sau description: > Phân tích các file d? li?u CSV và d?
ng b?ng - tính toán s? li?u th?ng kê tóm t?t, thêm các c?t d?n xu?t, t?o bi
?u ?? và làm s?ch d? li?u l?n x?n. S? d?ng skill này khi ng??
i dùng có file CSV, TSV ho?c Excel và mu?n khám phá, chuy?n ??i ho?c tr?
c quan hóa d? li?u, ngay c? khi h? không ?? c?p rõ ràng ??n "CSV" ho?
c "phân tích".
```

The description has been improved to be more specific about the skill's functionality (summary statistics, derived columns, charts, data cleaning) and more comprehensive about its scope of application (CSV, TSV, Excel; even without explicit keywords).