

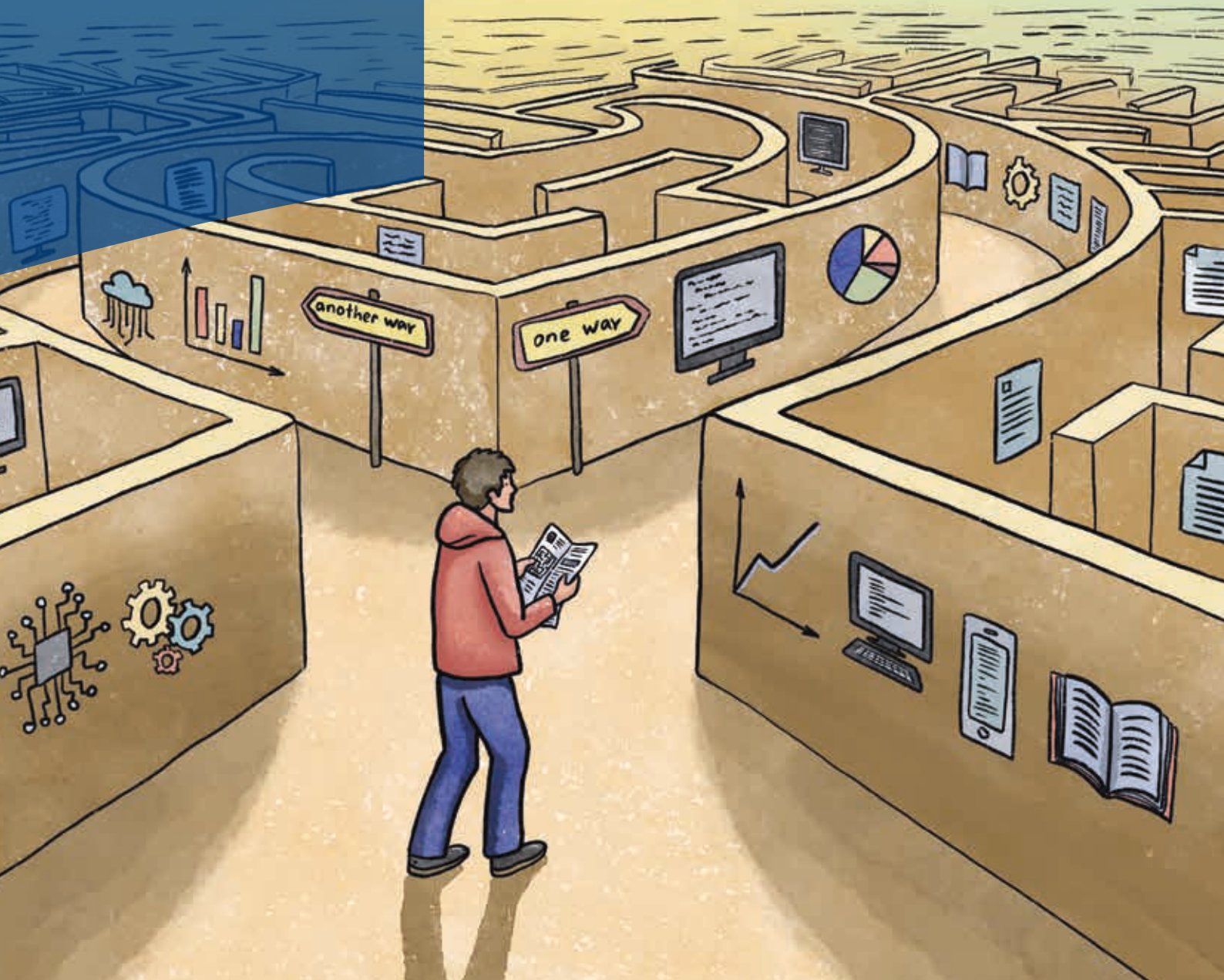
Communicator

The Institute of Scientific and Technical Communicators

Winter 2025

Open Theme

- Writing from multiple sources
- Publishing in XML
- Career path reflections



Ask the right questions: Testing documentation chatbot tools

By Ivana Isadora Devcic and Tamara Rajić.

Choosing the chatbot tool for your documentation is not just about selecting the most advanced LLM (Large Language Model). It is about finding one that fits your use case, supports the features you need, integrates seamlessly into your workflows, and helps users get accurate and consistent answers under different conditions. Structured testing that reflects real-world usage is the most effective way to determine which chatbot tool performs best.

Testing chatbot tools is a multi-step process that typically includes the following phases:

- Defining use cases and intents, and selecting multiple chatbot options that align with your goals.
- Collecting real queries from support tickets, internal feedback, or user interactions to build a comprehensive question database.
- Including variations in phrasing and organizing queries by category to reflect how users naturally ask questions.
- Focusing on accuracy and the chatbot's ability to handle differently worded but similar questions without fabricating answers; ambiguous or undocumented topics; and questions unrelated to the contents of your documentation.
- Ensuring that the chatbot is configurable to clearly indicate when a question or topic is outside the scope of the documentation, rather than guessing or providing incorrect information.
- Testing multiple chatbot tools with the same queries, recording their responses, and performing side-by-side comparisons to highlight their strengths and weaknesses.
- Involving real users (coworkers or product customers) to test the chatbots and provide feedback on performance and usability. This ensures you're not testing in isolation, but confirming how well the tool serves your actual audience.

To help you get started, this article covers four types of questions - from basic product concepts to ambiguous or undocumented topics - that your chatbot testing process should include to evaluate both the accuracy and the effectiveness of the responses. Most of the examples in the article focus on software documentation, but the advice should be widely applicable.

Basic product concepts

In the early stages of exploring a product, the users will want to understand its fundamental concepts. To achieve this, they may use the chatbot to learn more about the terminology and the underlying product architecture. Your testing questions should evaluate whether the chatbot is able to explain how the product works at a conceptual level. Because many concepts depend upon others (either general or product-

specific), the chatbot should be able to identify those relationships in its responses, and inform the users about any technical knowledge prerequisites they need to work with the product. Example questions for testing could include:

- "How is [concept 1] different from [concept 2]?"
 - "What is the purpose of [feature]?"
 - "Does [product] support [some technology or framework]?"
- Additionally, the chatbot should be able to understand universal technical terminology and correctly explain it in the context of the product's implementation:

- "What is an API?"
 - "How does authentication work in [product]?"
- It is important that the chatbot can handle overloaded terms or terminology collisions - instances where your product documentation uses terms in a specific meaning that is different from its standard, generic usage. For instance, if "policy" has a specific meaning in your product, the chatbot response should not attempt to define it in the generic sense:

- "What is a policy?"
 - "How do policies work in [product]?"
- Answers to conceptual questions should be consistent and without contradiction even when the questions are phrased differently. They should reflect the chatbot's ability to synthesize relevant information from multiple sources.

Common usage

With software products, common or general usage questions refer to typical questions users ask about product installation, day-to-day workflows, troubleshooting, or key product functionality. If your documentation covers multiple products, your testing questions should take that into consideration to ensure the chatbot is evaluated across the full range of supported use cases.

Some example questions, along with variations of the same question, include:

- "Can I share a project with people outside my team?"
 - ♦ "Can I share a project with others?"
 - ♦ "Can a project be shared with people from other teams?"
 - ♦ "Can a project be shared with someone outside my team?"
- "Why won't my file upload?"
 - ♦ "Why can't I upload a file?"
 - ♦ "Why does [product] throw an error during upload?"
 - ♦ "Why can't I upload a file to [product]?"

When asking questions on common usage, the chatbot's answers should be clear and concise. This means that instead of providing step-by-step instructions, it should give a direct explanation and, when appropriate, include a link to the relevant documentation page.

Instructions

Instruction queries are made when users need step-by-step

guidance to complete a task with the product. The expected response is a clear and structured sequence of actionable steps, often supported by links to relevant parts of the documentation.

Chatbot responses should adapt to the type of instructions the user requested. To ensure that, try testing the following scenarios:

1. Distinguishing between multiple products: The chatbot should either guess the most likely product based on context or, if the product name is explicitly mentioned, provide the correct instructions for that product. For actions possible across multiple products, the chatbot should include instructions for all relevant products. For example, users might ask the following question along with its variations:

"How do I export reports in [product]?"

- ♦ "How can reports be exported in [product]?"
- ♦ "How does report export work in [product]?"
- ♦ "I want to export a report"

2. No specific documentation available: The chatbot should notice when instructions are not available and, if possible, provide general guidance, or point users to the relevant support contact. It is important that the chatbot does not make up the steps for undocumented, partially documented, or unsupported features. For example, users might ask the following question or its variations:

"How to scan [unsupported file type] with [product]?"

- ♦ "Show me how to scan [unsupported file type]"
- ♦ "What do I need to do to scan [unsupported file type]?"
- ♦ "I want to scan [unsupported file type]"

3. Troubleshooting: When users want to understand why something isn't working as expected and resolve the issue, the chatbot should provide step-by-step solutions without fabricating answers. If the issue is too complex or undocumented, the chatbot should redirect users to the relevant customer support contact. For example, users might ask the following question and its variations:

"Why can't I scan files with [product]?"

- ♦ "Why won't [product] scan my files?"
- ♦ "What do I need to do for [product] to start scanning files?"
- ♦ "I want to scan a file but [product] is throwing an error"

This approach ensures that the chatbot gives actionable instructions when possible, while also handling debugging or unsupported scenarios effectively. As a result, the chatbot remains reliable, providing practical answers and clear explanations without inventing misleading or factually incorrect information.

Ambiguity, content gaps, and out-of-scope requests

To remain useful, the chatbot tool needs to tolerate ambiguous user input to a degree, and have a fallback mechanism in place for topics that are outside of the documentation scope.

There are many cases where ambiguity can arise. For instance, users may ask vague questions phrased so that they could refer to any other product, or that have several possible interpretations:

- "How does this work?"
- "I want to remove the permissions"
- "How do I upgrade from the old version?"

The chatbot should rely on the existing conversation context to determine the most likely interpretation, or ask the user to clarify their intent before generating a response.

Users may also ask questions on topics entirely unrelated to

the product, or not covered in the current version of the documentation. These questions can be about:

- Information undocumented on purpose (such as internal or confidential product details),
- Missing or outdated information (for example, features deprecated in previous versions),
- External product knowledge (like error messages caused by the operating system),
- Irrelevant topics (the weather, fashion advice, or math problems)

The chatbot should be able to deflect such requests in an appropriate way. The responses should clearly indicate when and why the requested information cannot be retrieved, instead of inventing non-existent features, fabricating documentation content, or producing errors and unhelpful output messages.

Building the interrogation toolkit

The goal of testing the chatbot with a variety of question types is to simulate realistic scenarios and reveal as many issues as possible before your users discover them. You can then try to resolve the detected issues by modifying the chatbot configuration, improving the documentation content, or both.

The testing process will also help you understand the chatbot behavior as you follow and compare the ways in which it navigates different levels of complexity in your questions. Depending on the configuration available in the chatbot tool, you may be able to direct this behavior to create a better user experience. Some chatbot tools provide a rating system that you can leverage to filter and prioritize the responses by quality, effectively "teaching" the chatbot how to respond to future similar queries.

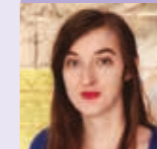
Although many parts of the testing process may be automated, especially if you are working with large documentation sets, the human element is critical and irreplaceable when validating and reviewing chatbot responses. Because of the non-deterministic nature of chatbot outputs, it is unrealistic to expect perfect, fail-proof responses even after continuous testing and improvement. However, by asking the right questions, assessing the answers, and adjusting relevant parameters, it is possible to reach the stage where the chatbot becomes more of a companion and less of a liability. ■

Tamara Rajić



Tamara Rajić is a technical writer from Croatia. She works at ReversingLabs, where she has spent the past four years honing her skills in communicating technical and cybersecurity-related topics. With a master's degree in English and Computational Linguistics, along with an interest in programming, she enjoys exploring how language and technology come together. She is passionate about translating complex ideas into content that people can understand and trust.

Ivana Devcic



Ivana Devcic is a technical writer, editor and open-source advocate with a background in linguistics and translation. Building on her experience as a documentation developer for enterprise security solutions and RESTful APIs, Ivana has made it her mission to help people use software more efficiently and productively.