elvex

The elvex Prompting & AI-Assistant Guide

Updated November 2026

elvex.ai

elvex is the enterprise platform that makes it easy to bring generative AI to work.

People who use AI can work much more efficiently. However, it's frankly a weird new way to interact with your computer and do work.

Learning how to prompt effectively is key to getting better results. Frequently, this takes about 10 hours of experimentation. This guide should help you get there faster, and will also walk you through some elvex-specific prompting techniques.

The beginning of this guide is for people who are newer to using Large Language Models (LLMs) at work. The middle of the guide walks through bringing data to your elvex assistants. The end of the guide contains instructions on how to build assistants, and advanced prompting techniques to use specific tools on our platform.

Contents:

1. The Learning Curve: From Learning LLMs to Automating Work
2. What Model Should I Use?
3. Step one: Best practices for learning to chat with LLMs
4. Step Two: Bringing your data to LLM interactions
5. Step Three: Solidifying your workflow into an elvex assistant

The Learning Curve: From Learning LLMs to Automating Work

Large Language Models (LLMs) generate text by predicting the next most likely word or sequence. Their responses are guided by patterns learned during training. The role of a good prompt is to give your LLM a very strong pattern for it to match—so that the text it predicts fits your exact need.

Models will "hallucinate" if they do not have the information necessary to do the job being asked of them. For example, if you are asking it to help solve a specific customer support issue for your company's software, it might not have information specific to that problem—and it will make up an answer. However, if you include your software's documentation in your prompt, it will reliably give a good answer every time.

This is why connecting your model to data is a key part of getting reliable results. "Connecting your model to data" effectively means simply adding more information to the prompt that gets sent to the LLM. For example, if you attach a Word document to your chat with an LLM, and say "summarize this doc," the prompt that gets sent to the LLM is both the words you typed and the entire text of that attached document.

There are several ways to connect data to your prompt. The first method is within the chat itself: you can attach files or paste in thousands of words to the chat window. The second method is to

connect an elvex datasource to your assistant.

This broadly maps to our suggested “learning curve” of using elvex and LLMs:

- **Step One:** Learn how to chat with LLMs, and what they are good at. Get a feel for how changing what you type into the box will change what is returned to you.
- **Step Two:** Learn how to equip LLMs with work data for better outputs. Begin to understand what kind of data is useful for your workflows, where to find the data, and how to attach it to an assistant.
- **Step Three:** Solidify your workflow into an “assistant,” and share it with your colleagues. Once the workflows have been tested and are found reliable, find other people who can benefit from them.

What Model Should I Use?

What AI providers do you support?

We support OpenAI, Anthropic, Google, Cohere, Mistral and numerous open source models. We also support Amazon BedRock and Azure OpenAI for hosted private instances.

What model should I use?

It depends on the job. The leading models are close on everyday work, so pick by task and cost, and test on your own examples.

- **The hardest work** (deep analysis, long multi-step agent tasks): Claude Fable 5.2 scores highest on Artificial Analysis's Intelligence Index. Gemini 4 Carbon and Claude Opus 5.5 are close behind, and Gemini 4 Carbon costs a quarter as much.
- **Everyday writing, coding and research:** Claude Sonnet 5.5, GPT-6.1 Sol and Gemini 4 Carbon all cost $2 / $10 per million tokens. Of the three, Gemini 4 Carbon scores highest.
- **Cheap, fast, simple tasks:** Claude Haiku 5.5 and GPT-6 Luna ($0.10 / $0.50), or DeepSeek V4 Flash ($0.15 / $0.60). Haiku 5.5 scores highest of the three.
- **Very long inputs:** every model above takes about 1 million tokens. Quality is most reliable up to about 60–70% of that.
- **Reasoning:** all of these models reason. Instead of choosing a separate "reasoning model", you set the effort level (low to max). Higher effort gives better answers but uses more tokens.

How much do they cost?

1 million tokens is about 750,000 words; War and Peace is about 560,000. Here's an example for one heavy user: 200 requests a day, each with about 5,000 tokens in and 1,000 out, over 22 working days.

- Claude Haiku 5.5: about **$4** a month
- Claude Sonnet 5.5, GPT-6.1 Sol or Gemini 4 Carbon: about **$88** a month
- Claude Fable 5.2: about **$350** a month

Most people sit well below that. Prices change monthly, so check models.dev for current rates.

Model breakdown

Prices per million tokens. Sources: models.dev and Artificial Analysis, November 13, 2026.

Model	Input	Output	Context	Intelligence Index (best setting)
Claude Fable 5.2	$8	$40	1M	64.6
Gemini 4 Carbon	$2	$10	1M	60.0
Claude Opus 5.5	$4	$20	1M	59.1
GPT-6 Astra	$10	$50	1.05M	56.9
GPT-6.1 Sol	$2	$10	1.05M	54.6
Claude Sonnet 5.5	$2	$10	1M	54.5
Claude Haiku 5.5	$0.10	$0.50	1M	36.5
DeepSeek V4 Flash	$0.15	$0.60	1M	32.6
GPT-6 Luna	$0.10	$0.50	1.05M	27.9

Step one: Best practices for learning to chat with LLMs

If you’re starting from zero, the first thing to do is simply chat with LLMs. Get a feel for how they work. These tips and examples will help you get started.

Iteratively Refine Prompts

- Test different versions, review the outputs, and adjust your prompts accordingly.

- Be prepared to engage in interactive, multi-turn dialogues; even slight adjustments can yield better results.

Role Play

- Models give better outputs when you give it a specific role for it to play. For example, “You are a demand generation manager at a B2B SaaS company…” or “You are a SEO specialist at a major newspaper…” or “You are a finance controller at a CPG company…”

Balance Specificity and Ambiguity

- For certain tasks where you want specific output, avoid ambiguity in your prompting by defining the task, format, and any constraints explicitly. Use active, declarative language instead of vague instructions.
- Other tasks benefit from more vague instructions, for example when you are uncertain of the outcome you want, and you want to see what the model comes up with.
- Specify word limits, output structure (e.g., “use a numbered list”), or tone (e.g., “explain as if to a beginner”).
- Direct the model by specifying exact output requirements (for example, “output in JSON format” or “respond with three distinct points”).

Context and Tasks Are Opposites: Give More Context, Give Less Tasks

- Context is information the model needs to do a good job, and generally they can handle a lot of context—many of them can handle the equivalent of War and Peace by Leo Tolstoy.
- Tasks, on the other hand, are things you are asking the model to do. The more steps and instructions you try to accomplish with one prompt, the more the LLM will struggle to do it all perfectly. Frequently, it’s better to have 5 interactions with 1 instruction each, than 1 interaction with 5 instructions.

Use Examples If You Can

- Including one or more examples (few-shot prompting) can set an expectation and significantly increase output quality.

Encourage Reflection

- Ask the AI for step-by-step reasoning or for self-critique of its previous response to refine and improve the answer.
- Ask the AI, “Explain your reasoning step by step,” or include a small chain-of-thought in your prompt that lays out intermediate steps. Or prompt it with “Review the above answer and point out any inaccuracies or missing details. Then, produce an improved version. This works especially well with the reasoning models.

Practical examples for how to chat with LLMs

Here are some examples of what prompting LLMs for work looks like. Remember, after this step, you improve your LLM interaction by bringing your own corporate data to the interaction. Many advanced are much more complicated than these—these are basic examples.

Customer Support & E-Commerce

Use Case: Handling a Damaged Product Query

Task: Develop a prompt for a customer service chatbot to resolve complaints about damaged items.

Example Prompt:

> "You are an experienced customer support agent for an online retail platform. A customer writes, 'I just received a damaged product, and I’m unsure what to do.' Provide a clear, step-by-step resolution process including options for refund, replacement, or repair, and include any necessary safety checks or return instructions."

Expected Outcome: The AI produces a detailed, empathetic response that outlines the process (e.g., apologizes, asks for order details, describes the return process, offers replacement/refund options, and mentions any required actions like packing instructions or shipping labels).

Technical Writing & Documentation

Use Case: API Documentation Creation

Task: Create clear documentation for a new API.

Example Prompt:

> "You are a professional technical writer. Your job is to document a new RESTful API endpoint that retrieves user data. Write a concise explanation of the API endpoint, including the URL, HTTP method (GET), query parameters (user_id), expected response format in JSON, and a brief example usage. The documentation should be suitable for intermediate developers."

Expected Outcome: The output includes section headers (e.g., Overview, Endpoint, Parameters, Response, Example), clear explanations of each section, sample JSON output, and tips on error handling.

Educational Content / Tutoring

Use Case: Explaining a Complex Concept in Simple Terms

Task: Develop a prompt for creating educational content that makes difficult subjects accessible.

Example Prompt:

> "You are an experienced college tutor. Explain the concept of quantum entanglement in simple terms for high school students. Use analogies, real-world examples, and a step-by-step breakdown. Keep the explanation under 300 words."

Expected Outcome: The response includes a clear, simplified definition of quantum entanglement, relatable analogies (such as comparing it to linked pairs of shoes or a pair of gloves), and a clear, linear explanation that makes the complex topic accessible without oversimplifying the science.

Healthcare & Medical Guidance

Use Case: Virtual Health Assistant for Symptom Triage

Task: Craft a prompt to help an AI provide preliminary advice for common symptoms while emphasizing that it is not a substitute for professional care.

Example Prompt:

> "You are a virtual health assistant designed for symptom triage. A user reports experiencing a mild fever and a sore throat. Provide general advice on self-care measures, when to seek medical attention, and suggestions for over-the-counter remedies. Clearly state that this advice is informational and not a substitute for professional medical care."

Expected Outcome: The AI produces a response that outlines basic fever care, lists symptoms that should prompt a doctor's visit, advises rest and hydration, and emphasizes that the information isn't a diagnosis.

Legal and Compliance

Use Case: Summarizing Contract Terms

Task: Generate a summary that highlights key points of a legal contract in plain language.

Example Prompt:

> "You are an expert legal consultant. Summarize the key terms and conditions of the following contract (insert contract text). Focus on the obligations, rights, durations, and termination clauses, and present them in clear, non-legal language suitable for a non-expert client."

Expected Outcome: The output outlines key clauses of the contract in simple terms, lists responsibilities and benefits for the involved parties, and highlights any important deadlines or conditions that could affect the client.

Looking for more examples of business workflows? Download our Workflows Guide for a walkthrough of 66 workflows, what types of roles use them, the results they generate, and copy-paste prompt engineering.

Step Two: Bringing your data to LLM interactions

In this section, we'll introduce some of the concepts involved in attaching data to your work, and then show you how to do it in elvex.

What is Retrieval Augmented Generation (RAG)?

LLMs will hallucinate when left to their own devices—and the way to solve this is to add your data to the interaction. This practice is what is known as "grounding."

The most common way to ground your interactions is through a technique called "Retrieval Augmented Generation" (RAG). Don't worry, elvex makes this easy.

Effectively, with RAG, you're taking what was a one-step interaction with an LLM and making it a two-step interaction, by adding a preparation step before the final interaction. This preparation step uses an LLM to search the attached data for the most relevant information.

Imagine you're writing a research paper for school: you need reliable sources, so you look up information in textbooks or websites. Then you use those facts to write your paper.

RAG works the same way. With RAG, the model doesn't have to rely on only its own training, which might be incomplete or outdated. It's like giving the AI a quick reference sheet, so it can provide answers supported by real facts, rather than guessing.

With elvex, RAG is automatically provided and maintained via our "Data Sources" feature.

When attaching data works—and when it doesn't

Think of RAG like doing a literature review: instead of reading every single page of every single paper in a massive library, you query a catalog for keywords and only read the relevant chapters or articles. This saves you from information overload, ensures you focus on what matters, and keeps the conversation manageable.

It's fantastic for things like answering customer support—you upload a bunch of solved customer support tickets, and it will intelligently find the correct ones to reference.

However, there are a number of scenarios where attaching a ton of data to an LLM interaction won't give you the answer you want, unless you're very specific in your prompting.

For example, let's say you're in marketing, and you're writing a blog about your product. You have an assistant that is connected to all your product documentation, some product reviews, and also your brand's style guide.

If you simply ask it "Write me a blog on our new features," it will do a great job pulling information from the product documentation—but it won't know to reference the style guide to make sure that the tone of the blog correctly matches your brand style. Instead, it will look through the style guide for information on the new features, probably won't find anything useful, and not use that data source at all.

A better approach would be to break this interaction into multiple tasks. "Write me a blog on our new features" would be the first command, and then "Read the entirety of our Style Guide, and then edit the blog you wrote to match our Brand Style" would be the second interaction.

It's important to note here is that you've asked the language model to read the entirety of the style guide, and not just search through it for chunks of information. That way, you know that all of the style information is given to the model, and not just parts of it.

Formatting can be important

LLMs are great at handling ambiguity and a lack of structure, but they can still fail. It's certainly possible to overcorrect and spend too much time prepping the data you're using, but there are times when cleaning up your data is worth the effort.

PDFs can be tricky

The PDF file format doesn't have very strong inherent structuring to it. So, while the information on the page might look cleanly formatted to people reading it, what the computer receives is actually jumbled. This doesn't mean you shouldn't use PDFs—but if the same data is available in another document format, it's generally better to use the other format.

Spreadsheets should be simple and standardized if possible

Many corporate spreadsheets are a complex mess of formatting: merged cells, multiple tables within the same sheet, and so on.

For the most part, models work much better when there's one table per sheet, with column headers, row headers, data—and little else. Dates, currency, and numerical formats should be standardized.

Provide the LLM a brief description of the table structure if the data is complex—explaining what each column means can guide the LLM during interaction.

Setting up an elvex data source

Datasource creation can be found on the left hand side of the elvex user interface. From there, you can see already-created datasources for your business, or click "Create new datasource" to start a new one.

It's important to note that not everyone is allowed to create data sources. Only people with the creator, admin, and owner roles in the elvex platform can create data sources. If you can't create data sources, contact your team's admin.

Once you're creating a new datasource, there are a few things to keep in mind:

- **Name and Description:** Give your datasource standard, self-evident names that make sense. These names and descriptions are used by the LLMs to find the correct data to use, so having descriptive names help them find information more accurately.
- **Tag:** If your business has a lot of datasources, keeping them organized by tags will help keep your workspace manageable.
- **Permissions:** Keep in mind that if a datasource is public, anyone in your businesses' workspace will be able to add it to their apps. If your datasource has sensitive information, keep it private, and only share it with the other users that need to interact with it.
- **Adding other people:** You don't have to maintain a datasource by yourself—give someone else editorial privileges and they can help you.

Data formats elvex works with

elvex supports these data formats within our data source feature: csv, doc, docx, htm, html, js, jsonl, md, pdf, ppt, pptx, py, rtf, ts, tsv, tsx, txt, xls, xlsx.

Within the chat window, you can also submit images and audio files.

Data integrations elvex works with

There are two main ways to integrate your company's data sources to elvex.

- **Cloud drives:** we work with Google Drive, OneDrive, Sharepoint, Box, and Dropbox.
- **Databases:** we work with Snowflake, Bigquery, and Postgres.

Cloud drives are the simpler way to connect data—working with a database will likely require involving one of your database admins for permissioning.

In both cases, elvex connects to your data sources live—if you update one of your files, you won't need to reconnect it to elvex. We will sync with your data and stay up to date.

Step Three: Solidifying your workflow into an elvex assistant

Now that you've gotten a feel for how to chat with LLMs, and how to bring data to an LLM interaction, your next step is solidify your workflow into something you can share with other team members.

This is important! Working with AI is not a skill-set everyone has, and not everyone takes the time to figure out good AI workflows. By creating, refining, and perfecting elvex workflows, you can up-level everyone on your team.

This section focuses on the assistant creation process and advanced prompting—specifically tool calling, which can help you improve the reliability of your assistants.

The App Builder

The easiest way to create a new assistant is to use elvex's App Builder. Simply type out what you want to achieve, and the App Builder will do the preliminary work for you!

If you want to create on your own, however, you can go to the Apps screen and click "Create new app." There are then seven steps (some of which can be skipped if need be).

The App Configuration Screen

There are seven steps to assistant configuration. Not all are always necessary. We'll walk you through all of them. We are using a hypothetical "Content Streaming Analysis" assistant as an example.

Overview

This is where you Name, Tag, Describe, and give your app Rules. Think of the Name, Description, and Rules as your prompt engineering—the same instructions you gave the LLM while simply chatting are what you'd put in here.

Context & Datasources

Context is additional information your assistant should know to do its job such as company acronyms, product names, or other domain-specific knowledge.

This will help your assistant generate more relevant content. Datasources are the sources of information we set up in Step Two of this guide, the information your app will search to help perform its tasks.

Tools

Tools expand the capabilities of your assistant beyond text input and output. They include Web Browsing, Drafts, Data Analysis, and Image Generation. Note: there are advanced tools you can instruct your assistant to access that aren't visible in this graphic interface. Read on below for a walkthrough.

Conversation Introduction

The conversation introduction gives users of your assistant ready-made options to get started with, instead of a blank screen.

Security & Permissions

Filters ensure that sensitive information is redacted from data sent to an AI provider. Some filters may be applied globally to all apps by your administrator.

App Visibility is very important as well—everything that is public will be visible to the whole company. Make sure your assistant is shared with the correct people. Similarly, you can give other people the ability to edit your assistant.

Slack Settings

You can add your assistant to Slack! They can be available to everyone in your Slack workspace, or just the people you've shared the assistant with in elvex.

Advanced

This is where you'll select which model your assistant uses, and what temperature of responses you want. Increase the temperature for more variability.

If you are using the API, here is where you will find the information required to make API calls to this assistant.

Advanced Prompting: Tool Calling

Tool Calls are built-in functions designed to let the AI access databases or external resources on your behalf. They enable the system to perform complex tasks—ranging from data extraction and SQL queries to creating customized text documents—dynamically.

This section provides best-practice recommendations for designing prompts and effectively using specific tool calls on the elvex platform. Frequently, dialing in the Tool Calls can improve the reliability of your assistant. This is important when sharing assistants with colleagues who may be less familiar with LLM interactions or the specific assistant you created.

Crafting a Clear and Specific Prompt

Step-by-Step Instructions: Break down complex requests into numbered steps.

For example:

1. Use `create_draft` to start a new draft document with a title and identifier.
2. Make edits using `update_draft` if further refinements are needed.
3. Perform data searches with `search_google` or query spreadsheets with `run_sql_on_datasource_file`.

Be Specific: Clearly state which tool call you want to use (e.g., `create_draft`, `update_draft`, `search_google`, or `run_sql_on_datasource_file`). For example, "create a text document summarizing our quarterly report" is more effective than “summarize our quarterly report.”

Provide Context: Explain necessary background and include all relevant parameters. If you need SQL insights, mention which DatasourceFile and relevant tables to use.

Define Format: If you expect a particular output format (e.g., markdown document, SQL query results, or a chart), specify that clearly. For example, when needing a new document use `create_draft` with type "document", or for code use type "code/python.” Mention if you want detailed output or if abbreviated responses are acceptable. For instance, you might add, "please provide a full breakdown using `run_sql_on_datasource_file` for SQL queries."

Include Examples: Provide a sample of the expected output. For example, if you need a specific query, provide a sample structure using the tool call syntax with examples.

Verification of External Data: When appropriate, request sources explicitly and verify the authenticity of URLs.

Specific Tool Calls and How to Use Them Effectively

`create_draft`: Use when you need to generate new documents (emails, reports, proposals) or code files. Always provide a clear title, identifier, type (e.g., "document" for rich text, "code/python" for code, or “spreadsheet” for spreadsheets), and content.

`update_draft`: Use for making precise edits to an existing document. Always refer to the identifier used when the document was created. This tool is ideal for iterating on drafts or refining specific sections (e.g., "shorten the introduction").

`render_graph`: Use for visual data representation. Provide clear data keys for the x-axis and y-axis and specify the type of graph (bar, line, or area). A well-defined JSON input ensures that the rendered graph matches the intended insights.

`generate_image`: Use this to create visual images.

`search_google`: Use when external search results are necessary. Provide clear search queries, including operators if needed (e.g., "site:wikipedia.org"). This ensures that you target the most relevant and reliable results.

`get_webpage_content`: Use when fetching webpage content is needed, for example when real-time information and context sourcing is necessary.

`date_add`: Use this for date and time related tasks.

`run_sql_on_datasource_file`: Use for executing SQL queries on tabular data within DatasourceFiles. Define the table data, including DatasourceFile IDs and sheet names. This is particularly useful if you need an exact answer or data analysis from provided spreadsheets or CSV files.

`list_all_datasources_and_files`: To list accessible Datasources and DatasourceFiles.

`search_within_datasource`: To perform semantic searches across DatasourceFiles in a specific Datasource.

`search_within_datasource_file`: To perform semantic searches within a specific DatasourceFile.

`read_datasource_file`: Read the entire contents of the file directly.

`get_spreadsheet_overview`: This tool retrieves column names, sample rows, and the table schema for a specific spreadsheet file.

Note: adding specific language about tool calls isn’t always necessary. Frequently the models are smart enough to figure it out on their own. However, being specific enhances reliability. Most people find it useful to be specific when they are asking for more complex tasks from the model.

Mastering LLM prompting is less about discovering a “magic phrase” and more about understanding how the models work, and iteratively learning what works best in a given context.

By combining clear instructions, relevant context, examples, and iterative refinement, you can harness the full potential of modern generative AI.

Get in touch with us via www.elvex.ai