

vosk-cli-dictation — User manual

Install, configure and use offline voice dictation

Version v0.5.4-beta

2026-09-18

Contents

Quick start	3
Chapter 1 — Getting Started	4
Prerequisites	4
Several ways to install	4
Retrieve a language model	5
First boot	5
A first dictation	6
Dictating punctuation	6
Uninstall and update	6
Chapter 2 — The Speech Recognition Engine	8
Who is this tool for?	8
What Vosk does	8
Acoustic model and language model	9
Nothing leaves the machine	9
Chapter 3 — The three states: ÉCOUTE, DICTÉE, VEILLE	10
Overview	10
DICTÉE — being dictated	10
ÉCOUTE — pending	10
How to start and stop a dictation	10
VEILLE — almost zero consumption, voluntary	11
The confidence threshold governs the triggering	11
When to use sleep mode	11
The unfolding of a complete session	12
Chapter 4 — Configuration and customization	13
Where is the configuration file?	13

Choose the language of the configuration template	13
Customize keyboard shortcuts	13
Choose how the dictation starts and stops	14
Punctuation: two input methods, per language	14
Set the confidence threshold for triggering	14
Add vocabulary and correct misunderstandings	15
Where are language models searched for?	15
Chapter 5 — System Integration: Wayland and X11	16
How does the text get into the application?	16
The microphone	16
The system tools used, and their automatic selection	16
Enable ydotool (fallback solution)	17
Keyboard shortcuts: X11 and Wayland are not the same	17
What happens if no injection tool is available?	18
Chapter 6 — Troubleshooting	19
Symptom → probable cause → action	19
Where can I find a complete error message?	20
If the problem persists	21

Quick start

For those who want to dictate within two minutes, without reading the rest right away.

`vosk-cli-dictation` is designed for resource-constrained environments and offline dictation, not for achieving maximum accuracy on a powerful workstation (Chapter 2). The keyboard shortcuts below assume an X11 session; under Wayland, they are handled by the desktop environment (Chapter 5).

1. Installation. Native package of the distribution:

```
# Debian 12+, Ubuntu 24.04+, Mint 22+ ...
sudo apt install ./vosk-cli-dictation-<version>_all.deb

# Fedora
sudo dnf install ./vosk-cli-dictation-<version>.fc42.noarch.rpm

# openSUSE (Leap)
sudo zypper install ./vosk-cli-dictation-<version>.leap156.noarch.rpm

# Arch Linux
sudo pacman -U ./vosk-cli-dictation-<version>-any.pkg.tar.zst
```

2. Retrieve a language model, if one is not already present (the package does not include them—they are too large).

```
vosk-cli-dictation --download-model fr
```

The small language model (here `fr`, also `en`, `de`, `es`) is downloaded, verified by its SHA256 checksum, and installed in `~/.local/share/vosk-models/`. The first time the tool is launched without any model, it offers this same download interactively. You can also manually download a model from <https://alphacephei.com/vosk/models> and unpack it into this directory.

3. Dictation.

```
vosk-cli-dictation
```

Say the wake-up phrase (“start dictation”) to start the dictation, speak, and then say the stop phrase (“stop dictation”) to finish, and copy the text to the clipboard.

That’s all for the basics. The following sections explain why and how to go further—customizing shortcuts, understanding sleep mode, and troubleshooting issues.

Chapter 1 — Getting Started

Prerequisites

- A **Linux system** with an X11 or Wayland graphical session. Without a display server, there is no active window where the recognized text can be written.
- **Python 3** and a virtual environment (`python3-venv`).
- A **text injection tool**: `wtype` or `ydotool` under Wayland, `xdotool` under X11. At least one must be present, otherwise the tool will refuse to start (Chapter 5).
- A **microphone recognized by the system** as the default input device (Chapter 5).
- **Disk space for the model**: from 50 MB (small models) to over one gigabyte. During operation, a session uses approximately 300 MB of RAM (Chapter 2).

The Debian package declares and installs all of its dependencies itself, including the system and Python components. In Git clone mode, `pip install -r requirements.txt` installs the Python libraries; the injection tool (`wtype`, `xdotool`, or `ydotool`) must then be installed separately.

Several ways to install

`vosk-cli-dictation` offers several channels, none of which replace each other.

The native package of the distribution (recommended for general use)

```
# Debian 12+, Ubuntu 24.04+, Mint 22+ ...
sudo apt install ./vosk-cli-dictation-<version>_all.deb

# Fedora
sudo dnf install ./vosk-cli-dictation-<version>.fc42.noarch.rpm

# openSUSE (Leap)
sudo zypper install ./vosk-cli-dictation-<version>.leap156.noarch.rpm

# Arch Linux
sudo pacman -U ./vosk-cli-dictation-<version>-any.pkg.tar.zst
```

Regardless of the format, the package installs a single command, `vosk-cli-dictation`, in `/usr/bin/`. It also creates an isolated Python environment and installs the speech recognition engine (Vosk) in it—this step requires a network connection during installation, but only once. No superuser privileges are required afterwards.

The installation itself never modifies personal files: no existing configurations are overwritten, and no user account directories are created or touched at this stage. It's only upon the first launch, and when retrieving a language model (described below), that files appear in the personal directory – never before, and never without an explicit action by the user.

Clone the Git repository (to follow development or contribute)

```
git clone https://github.com/RonanDavalan/vosk-cli-dictation.git
```

```
cd vosk-cli-dictation
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

In this mode, the tool is launched with `python3 src/main.py` instead of through the `vosk-cli-dictation` command of the package, and its configuration is located in `config/config.yaml`, alongside the code—never in the user’s home directory. The two modes do not mix: a Git clone ignores the package’s configuration, and vice versa.

Retrieve a language model

Neither of the channels includes a speech recognition model—they range in size from 50 MB to over a gigabyte, which is too large to be included with the tool.

The simplest way, when installing via a package, is to let the tool handle it.

```
vosk-cli-dictation --download-model fr
```

This command retrieves the small model of the requested language (`fr`, or `en`, `de`, `es`), verifies its SHA256 checksum, installs it in `~/.local/share/vosk-models/`, and then stops. On the very first launch without any models, the same prompt is presented interactively: answering “yes” will download the small language model for the interface.

Alternatively, or for a more comprehensive model than the small default model, download it manually from <https://alphacephei.com/vosk/models>, and then extract it:

- Installation via package: in `~/.local/share/vosk-models/`.
- Git clone: in `vosk-model/`, at the root of the cloned repository.

Without a template present in the correct location, the tool displays the locations it searched and the download address upon startup—it never silently fails.

First boot

```
vosk-cli-dictation
```

(In clone Git mode, the same command is `python3 src/main.py`, from the root of the repository and with the virtual environment activated.)

Three things are displayed: the verification of external dependencies (see Chapter 5), the loaded model, and a reminder of the available commands – wake and stop phrases, keyboard shortcuts, and punctuation words. This reminder is regenerated each time the application is launched from the active configuration; it always reflects what is actually usable and never a static list found in the documentation, which could become inaccurate after customization.

A first dictation

Two equivalent ways to start a dictation, choose one:

- **By voice:** Pronounce the wake-up phrase of the active language (“start dictation”). The complete phrase is required: a single word (“start”) does not trigger anything — the wake-up is only triggered by the entire phrase, with sufficient confidence.
- **By keyboard:** Press **Alt+H** — this shortcut toggles between ÉCOUTE (the starting state, the one you’re in immediately after launching) and DICTÉE (see Chapter 3 for the states); it is used both to start and to pause. It never copies anything to the clipboard: see below for how to end a dictation. Under a **Wayland** session, these shortcuts are only captured for windows that pass through XWayland; for a native Wayland window, they must be passed through the desktop environment (Chapter 5).

Then, speak normally. The recognized text is directly written into the application that has the focus — a terminal, an editor, a browser, anything.

To finish and retrieve the text in the clipboard: say the stop phrase (“stop dictation”) or press **Alt+S**. A stop phrase heard below the confidence threshold is absorbed — it is neither typed into the document nor triggered, and a hint is displayed: repeat it, or use **Alt+S**.

More reliable control, whether under X11 or Wayland, involves linking the sub-commands (`--toggle`, `--finalize`, `--sleep-toggle`) to desktop shortcuts (Chapter 5) rather than using the built-in shortcuts.

Dictating punctuation

Punctuation marks are dictated like the rest of the text — “comma”, “exclamation mark”, “new line”, etc. The complete list depends on the active language and is displayed when the program starts (see above); it can also be customized, and the way to trigger a punctuation command can vary depending on the language input mode (Chapter 4).

Uninstall and update

Native package. The distribution’s package manager removes the command and the isolated Python environment (`/usr/lib/vosk-cli-dictation/venv`): `sudo apt remove vosk-cli-dictation` (Debian 12+ and derivatives; `sudo apt purge vosk-cli-dictation` also removes the factory configuration model), `sudo dnf remove vosk-cli-dictation` (Fedora), `sudo zypper remove vosk-cli-dictation` (openSUSE), `sudo pacman -R vosk-cli-dictation` (Arch Linux). The personal configuration file (`~/.config/vosk-cli-dictation/config.yaml`) and language models (`~/.local/share/vosk-models/`) are never affected by the uninstallation—delete them manually if desired.

Clone Git. Delete the cloned directory: it contains everything (code, virtual environment `venv/`, any templates in `vosk-model/`). Nothing is written elsewhere in this mode.

Update. With a native package, install the latest file using the same command as the initial installation (`apt install`, `dnf install`, `zypper install`, `pacman -U`): it replaces the version in place without affecting your

personal configuration. With a Git clone, run `git pull`, then rerun `pip install -r requirements.txt` within the activated virtual environment to propagate any new dependencies.

Chapter 2 — The Speech Recognition Engine

Who is this tool for?

Before delving into the technical details, let's be upfront: `vosk-cli-dictation` is not designed to compete with the most accurate dictation engines available on the market when running on a modern desktop computer. This isn't a hidden limitation – it's a deliberate architectural choice that makes the tool excellent in a specific area:

- **Constrained Hardware** — Raspberry Pi, old laptop, machine without a GPU. The engine never needs hardware acceleration. In operation, a session uses approximately **300 MB of RAM** (209 to 383 MB depending on the language and machine — measured small models: `vosk-model-small-fr-0.22`, `-en-us-0.15`, `-de-0.15`, `-es-0.42`) — modest compared to a browser, but not negligible on a machine with limited resources.
- **Real-time responsiveness** — the text appears as the speech is being given, without waiting for the end of a segment. This is a property of the engine's architecture (see below), not a setting that could be lost by changing a parameter. The decoding does not lag behind the speech: processing an audio block takes less time than the block itself (real-time factor $RTF < 1.0$), including in the most demanding case: dictation in French on a Raspberry Pi 4 ($RTF\ 0.73$) or on an AMD A4-6210 ($RTF\ 0.79$). French is the most expensive language to decode; on the same modest hardware, English, German, and Spanish run at $RTF\ 0.20$ – 0.33 .
- **Nothing leaves the machine** — no audio or text data is ever transmitted to a remote service, during recognition or at any other time.
- **Minimal dependency chain** — no large deep learning library to install, no graphics card driver required.

In return, the understanding of meaning remains limited: two words that sound the same but are spelled differently (“their”/“there”, “to”/“too”) are not always correctly distinguished based solely on the sound. Chapter 4 explains how to correct these cases one by one, once they are identified through use – this is not automatic.

If the goal is to write long texts with maximum accuracy on a powerful desktop computer, another tool will be better suited. If the goal is to dictate on modest hardware, in real-time, without ever relying on a network connection, `vosk-cli-dictation` is specifically designed for that purpose.

What Vosk does

Vosk is the speech recognition engine used by the tool – a standalone library, not a custom-built component. It relies on an older architecture than transformer-based engines (like Whisper): an acoustic model combined with a statistical language model, following the lineage of the Kaldi software.

This architecture has a direct and useful consequence: the decoding happens in real-time, at the rate at which the sound arrives from the microphone, with results displayed progressively—not in blocks of several seconds that would require waiting. This is what gives the feeling of “live” dictation.

Acoustic model and language model

Each downloaded model (Chapter 1) contains two parts that are not visible during use, but are helpful for understanding:

- **The acoustic model** recognizes the sounds of speech and associates them with language units (phonemes).
- **The language model** evaluates which sequences of words are statistically likely in the relevant language, in order to choose between multiple possible interpretations of the same sound.

The models offered by the Vosk project come in several sizes. `vosk-cli-dictation` defaults to using the compact models (“small,” approximately 50 MB on disk), which provide a good balance for resource-constrained hardware. However, you can specify a larger model in the configuration (Chapter 4) if the target machine allows it, at the cost of a larger memory footprint.

Nothing leaves the machine

No steps in the speech recognition process require a network connection. The only time a connection is needed is during the initial installation of the package (downloading the Python engine) and the manual download of a language model—never during normal use.

Chapter 3 — The three states: ÉCOUTE, DICTÉE, VEILLE

The vosk-cli-dictation tool has three distinct states, each with a different CPU cost. Confusing these is the most common cause of unexpected behavior (“why doesn’t the wake word work anymore?”). This chapter clearly distinguishes them.

ÉCOUTE, DICTÉE, and VEILLE are the labels that the program displays on the screen. They remain in French regardless of the language of the input: they identify an internal state, not text to be translated.

Overview

State	How do we get in?	Active micro	Decoding	Wake
DICTÉE	Alt+H (or wake-up phrase), since ÉCOUTE	Yes	Yes	— (dictation already in progress)
ÉCOUTE	Alt+H, since DICTÉE	Yes	Yes	Yes
VEILLE	Alt+W	No	No	No

DICTÉE — being dictated

What you say is recognized and typed into the active application. To finish and copy the result to the clipboard: say the stop phrase (“stop dictation”), or press Alt+S.

ÉCOUTE — pending

This is the default state between dictations. The microphone remains active, and the engine continues to listen, for a single reason: to be able to hear the wake-up phrase at any time. Therefore, the CPU cost of this mode is not zero—it’s the price of vocal responsiveness.

To reduce this cost, decoding is performed in this state using a restricted grammar that only includes the wake-up phrase. The engine searches for nothing else while waiting, which significantly reduces the workload on the processor.

The alarm is not just a single word; it requires the **complete phrase**, and the engine only recognizes it if the confidence level of recognition is sufficient. A single word (like “start”) is never interpreted as an alarm – this is a safeguard against accidental activations during a conversation.

How to start and stop a dictation

Three ways to control the transition ÉCOUTE → DICTÉE → ÉCOUTE, adjustable via `interaction.mode` in the configuration (Chapter 4):

- **toggle** (by default, unchanged): the one described above — a gesture starts (**Alt+H** or wake-up phrase), a different gesture stops (**Alt+S** or stop phrase).
- **maintenance** (push-to-talk): the dictation lasts exactly as long as the key configured under `hotkeys.push_to_talk` (a single key, not a combination) is held down — when you release it, the session stops and is automatically copied.
- **silence**: a press (or the wake-up phrase) starts; the dictation stops and is automatically copied after a silence of `interaction.silence_timeout_ms` (700 ms by default, adjustable per language), as soon as at least one word has been captured. This is the only of the three modes that leaves your hands completely free during the entire phrase.

In all three modes, the voice stop command and **Alt+S** / `--finalize` remain valid safety nets—no mode removes an existing stop method.

VEILLE — almost zero consumption, voluntary

A third state, explicitly activated by **Alt+W**, and never by voice (by design: the microphone is muted in this state, so no words can be heard). Audio capture and voice decoding are both suspended, the CPU cost drops to **0.0%**, and the microphone is released.

The cost: The wake-up phrase no longer works as long as sleep mode is active. This is a deliberate choice, not a hidden flaw—the tool explicitly displays this when entering sleep mode. Only **Alt+W** remains active during sleep mode (it is not dependent on the microphone) to exit it.

Switching to sleep mode during an ongoing dictation first stops the dictation cleanly, as if **Alt+H** had been pressed — no text is lost in the middle of a sentence.

The confidence threshold governs the triggering

The wake and stop phrases are only accepted if the average confidence score of the words in the sentence reaches the configured threshold (0.55 by default, adjustable per language - Chapter 4). Concrete consequences:

- A stop phrase heard below the threshold is **absorbed**: it is neither typed into the document, nor triggered — an indicator is displayed, and it is enough to repeat it (or use **Alt+S**). This mechanism protects the document: an ordinary phrase resembling “stop dictation” will never appear in it.
- A wake-up or stop phrase said **twice in a row** is typed literally into the text, instead of triggering — this is intentional: immediate repetition is rare in voice control, and this behavior prevents a repetition from being lost.

When to use sleep mode

Useful whenever the microphone needs to remain muted for a significant period without leaving the program—such as a meeting, a call, or a time when CPU usage matters (on a laptop running on battery, or with limited hardware). On a machine where CPU usage is not a concern, staying in ÉCOUTE mode constantly is perfectly reasonable: it’s a choice, not an obligation to optimize.

The unfolding of a complete session

1. **Launch** the tool in a visible terminal. It starts in ÉCOUTE.
2. **Open the target application** (editor, browser, terminal) and give it keyboard focus.
3. **Switch to DICTÉE**: say the wake-up phrase, or press **Alt+H**.
4. **Speak** normally. The text is written in real-time in the application, and punctuation words (“comma”, “exclamation mark”, “new line”...) are interpreted as they are spoken (Chapter 4).
5. **Pause** if needed: **Alt+H** switches back to ÉCOUTE without copying anything; the text already written remains in the application. Press **Alt+H** again to resume.
6. **Finish**: say the stop phrase, or press **Alt+S**, or link `--finalize` to a desktop shortcut (Chapter 5). The complete text of the session is then placed in the clipboard.
7. **Paste** (Ctrl+V) wherever you want — the original clipboard is restored immediately after.
8. **Put it to sleep** between uses: **Alt+W** (microphone released, 0% CPU). Press **Alt+W** again to wake it up.
9. **Quit**: Ctrl+C in the launch terminal.

Chapter 4 — Configuration and customization

All of the behavior of `vosk-cli-dictation` is controlled through a single YAML file, `config.yaml`. This chapter explains where it is located and what can be changed in it.

Where is the configuration file?

The location depends on the installation channel (Chapter 1) and follows a specific order of priority – the first available location takes precedence, and the remaining locations are never considered.

1. The environment variable `VOSK_CONFIG`, if defined, is used as is and is never created automatically.
2. `config/config.yaml` from the repository, in Git clone mode.
3. `~/.config/vosk-cli-dictation/config.yaml`, during package installation, is created automatically on the first launch, from a factory template, with a message indicating where the file was created.
4. The factory template itself is read-only.

In practice: with the Debian package, modify `~/.config/vosk-cli-dictation/config.yaml` – this is a personal file and will never be overwritten by a package update.

Choose the language of the configuration template

The Debian package includes four localized templates — `config.en.yaml`, `config.fr.yaml`, `config.de.yaml`, `config.es.yaml`. They are located under `/usr/share/vosk-cli-dictation/`.

On the first launch, the one that corresponds to the system locale (`LANGUAGE` then `LANG`, falling back to English if none of the four match) is copied to `~/.config/vosk-cli-dictation/config.yaml`.

To switch to a different template after this initial launch, identify it using `dpkg -L vosk-cli-dictation | grep config` and then replace its personal file with the desired one:

```
cp /usr/share/vosk-cli-dictation/config.de.yaml ~/.config/vosk-cli-dictation/config.yaml
```

In Git clone mode, the four templates are directly in `config/` of the repository.

Customize keyboard shortcuts

Section `hotkeys:` of the file. Each shortcut is defined by a list of keys `pynput_keys` and a displayed name `display_name`:

```
hotkeys:
  toggle_recording:
    pynput_keys: ["alt_l", "h"]
    display_name: "[Alt+H]"
```

Three shortcuts are configurable here: `toggle_recording` (Chapter 1), `finalize_session`, and `sleep_toggle` (Chapter 3). `Ctrl+C`, which exits the program, is not included in this list: it is the standard system interrupt signal, always active and never reconfigurable from this file.

Under Wayland, these shortcuts only affect windows running through XWayland (see Chapter 5) — for native Wayland windows, it is the desktop environment that handles the shortcuts, through its control subcommands.

A fourth input, `push_to_talk`, uses only a single key (not a combination): it serves the push-to-talk mode (Chapter 3) and has no effect in the other two modes.

Choose how the dictation starts and stops

Section interaction: of the file (Chapter 3 for the presentation of the three modes):

```
interaction:
  mode: toggle          # toggle | maintainue | silence
  silence_timeout_ms: 700
```

`silence_timeout_ms` The threshold is configured per language, just like the confidence threshold below, under `language_settings.<lang>.silence_timeout_ms`. This is useful if a model decodes more slowly on a particular machine and triggers a timeout that is too short.

Punctuation: two input methods, per language

Each language, under `language_settings.<lang>.punctuation`, defines its list of punctuation marks and its input method:

- **mode: inline** (by default) — punctuation marks are recognized and replaced directly as you dictate: saying “comma” inserts a comma.
- **mode: explicit** — a punctuation command is only applied if it is immediately preceded by the language trigger word: you must say “punctuation comma” to get a comma, whereas “comma” alone is written out in full. This is useful when the language or vocabulary leads to frequent false triggers.

The actual list of words (simple and double punctuation, line breaks, special characters – hyphens, parentheses, quotation marks...) is displayed by the program when it starts; it is defined in the sections `simple`, `double`, `layout`, and `special`. The typography applied to the text (the absence of a space before `?` `!` `:` `;` and the use of straight quotation marks `"..."`...) is controlled by the `typography` block of the language.

Set the confidence threshold for triggering

The start and stop events are only recorded if the average confidence of the words in the sentence reaches the threshold `recognition.confidence_threshold` (default: 0.55, Chapter 3). This can be overridden on a per-language basis:

```
language_settings:
  en:
    confidence_threshold: 0.55
```

A higher threshold makes the trigger more strict (fewer false positives, but the stop phrase is more often absorbed); a lower threshold does the opposite.

Add vocabulary and correct misunderstandings

Two distinct mechanisms, under `language_settings.<lang>` :

- **custom_vocabulary** — Case-sensitive corrections, applied after everything else. Useful for acronyms or proper nouns that the engine consistently writes in lowercase:

```
custom_vocabulary:  
  "api": "API"
```

- **recognition_aliases** — corrects a recurring phonetic confusion in the engine (two words that sound almost the same, Chapter 2). For example, if the engine consistently recognizes “there” instead of “their” :

```
recognition_aliases:  
  "there": "their"
```

These two settings are discovered through use, not in advance — dictating, identifying a recurring confusion, and adding it.

Where are language models searched for?

Similar to the configuration process, the search follows a specific order, with the first directory found on the disk taking precedence.

1. An absolute path — despite its name, the key `path_relative` can also contain a complete absolute path; in this case, it takes precedence over everything else below.
2. `~/.local/share/vosk-models/` (or `$XDG_DATA_HOME/vosk-models/` if this variable is defined).
3. `/usr/share/vosk-models/` — models installed at the system level.
4. `vosk-model/` at the root of the project, in Git clone mode only.

If no model is found, the message displayed at startup lists these locations precisely, along with the download address, and reminds you of the command `vosk-cli-dictation --download-model` — there is never a failure without an explanation. This command installs the small model in location 2 above (`~/.local/share/vosk-models/`), which is the only location where a model downloaded without administrator privileges can be found later.

Chapter 5 — System Integration: Wayland and X11

`vosk-cli-dictation` doesn't have its own graphical interface; instead, it writes the recognized text directly into the application that currently has keyboard focus, regardless of which application it is. This chapter explains how this works and what changes depending on the display system being used.

How does the text get into the application?

Contrary to what one might imagine, the text is **not typed character by character**. The actual mechanism is as follows: the recognized text is placed in the clipboard, and then a “paste” shortcut (`Ctrl+V`) is simulated. This is more reliable and faster than simulating a complete typing process, especially for long passages.

The original clipboard is restored immediately after this pasting operation (`Ctrl+V`) — therefore, dictation does not overwrite what was copied there previously for more than an instant. Under Wayland with a GTK application, this restoration can occur before the window has finished reading the clipboard, which then pastes the old content; the setting `clipboard.defer_restore: true` delays the restoration until the end of the dictation session and eliminates this risk.

The microphone

`vosk-cli-dictation` captures audio from the **system's default input device** (PulseAudio or PipeWire). There is no device setting in `config.yaml`: to change the microphone, you need to set the desired device as the default input - in the sound settings of your desktop environment, using `pavucontrol`, or via the command line (`pactl set-default-source <nom>`, the list being provided by `pactl list sources short` or `arecord -l`).

`vosk-cli-dictation --diagnose` opens a test stream on this device and reports whether the opening fails—for example, if the microphone is absent, muted, or already being used by another application.

The system tools used, and their automatic selection

The injection process requires **two** tools: a keystroke simulator and a clipboard tool. These are automatically selected based on the session type, requiring no configuration.

- **Under Wayland:** `wtype` (input) and `wl-clipboard` (clipboard), which natively speak the Wayland protocol. If `wtype` is missing, the input passes through `ydotool` - but `wl-clipboard` remains necessary.
- **Under X11:** `xdotool` (input) and `xclip` (clipboard).

`ydotool` is the last resort for typing (it works in both worlds via `/dev/uinput`, but depends on the keyboard layout for its key codes). At least one typing simulator must be present — otherwise, the tool refuses to start and clearly indicates this. The Debian package depends on `wl-clipboard` and recommends `xclip`; when using Git, install them manually. Without a clipboard tool, dictated text never reaches the clipboard: nothing is written in the application, and a `Ctrl+V` re-pastes the last manually copied content. The command `vosk-cli-dictation --diagnose` checks the environment (session type, tools, models, microphone, clipboard, control socket) and displays what is missing.

Enable ydotool (fallback solution)

This section only applies to systems that fall back to `ydotool`, in the absence of `wtype` (Wayland) or `xdotool` (X11). When `wtype` or `xdotool` is available, nothing that follows is necessary—in particular, it is unnecessary to join the group `input`. The Debian package applies the same rule: its installation message only suggests adding to the group `input` if you are using the fallback `ydotool`, and specifies that `xdotool` and `wtype` do not require it.

Three conditions to check in order if the text does not appear:

1. **The package `ydotool` is installed.** On Debian, it resides in `trixie-backports`, not in the stable branch — enable this repository if necessary before installation.
2. **The service `ydotool` is running.** `ydotool` needs this background daemon to function.
3. **The user belongs to the group `input`.** It is this group that provides access to `/dev/uinput`, otherwise `ydotool` will fail silently for this user.

```
sudo usermod -aG input $USER
```

Disconnecting and then reconnecting is necessary for this group change to take effect; a simple terminal restart is not sufficient.

Keyboard shortcuts: X11 and Wayland are not the same

The built-in shortcuts **Alt+H**, **Alt+S**, and **Alt+W** are listened to by the tool itself through the `pynput` library, which monitors the X server. Under Wayland, it only detects windows that pass through **XWayland** (many Electron applications and older ones), and never a native **Wayland** window: for the latter, the shortcuts of the desktop environment must control the tool.

Each desktop environment can bind a combination of keys to a command. On the `vosk-cli-dictation` side, the control subcommands to use are:

Desired action	Subcommand
Start/pause dictation	<code>vosk-cli-dictation --toggle</code>
Finish the dictation and copy the text	<code>vosk-cli-dictation --finalize</code>
Entering/exiting sleep mode	<code>vosk-cli-dictation --sleep-toggle</code>
Know the current state	<code>vosk-cli-dictation --status</code>
Cancel the current dictation without typing anything	<code>vosk-cli-dictation --cancel</code>
Cleanly end the session	<code>vosk-cli-dictation --stop</code>

Example in KDE Plasma: In *System Settings* → *Shortcuts* → *Custom Shortcuts*, create a new entry under “Command/URL” with the command being `vosk-cli-dictation --finalize`, and assign it a key combi-

nation (e.g., **Meta+S**). The subcommand connects to the current session, executes the action, and displays the resulting status (OK DICTÉE, OK ÉCOUTE, OK VEILLE).

Even under X11, these subcommands are the most reliable method: a shortcut pynput is not consumed - the key combination reaches the targeted window, and when pressed in a terminal, it inserts a parasitic sequence (for example, `^[s` for **Alt+S**). Linking `--finalize` (or `--toggle`, `--sleep-toggle`) to a desktop shortcut provides a clean exit from any window.

What happens if no injection tool is available?

The tool does not start and explicitly indicates this, with a suggestion for installation. There is no “degraded” mode without injection: without `wtype`/`xdotool`/`ydotool`, there is no reliable way to write the recognized text into another application.

This startup check only covers the typing simulator. A missing clipboard tool (`wl-clipboard` under Wayland, `xclip` under X11) does not prevent startup, but breaks the injection: the dictated text only appears in the terminal, and each `Ctrl+V` re-applies the last content copied manually. `vosk-cli-dictation --diagnose` reports this.

Chapter 6 — Troubleshooting

Symptom → probable cause → action

Symptom	Probable cause	Action
The program refuses to start, with a message appearing in the injection tools	No injection tool (<code>wtype/xdotool/ymdotool</code>) is installed or functional	Installing one of them – see Chapter 5
“No speech recognition model could be loaded” at startup	No model found in the expected locations	The message lists the locations being searched for and the download URL—see Chapter 1
Nothing is detected, the microphone seems to be muted	Bad default input device, or microphone muted at the system level	Check / change the default setting (sound settings, <code>pavucontrol</code>); <code>vosk-cli-dictation</code> <code>--diagnose</code> - see Chapter 5
<code>--diagnose</code> indicates the failure to open the audio stream	Device is being used by another application, or insufficient microphone access permissions	Close the other application; check the device permissions – see Chapter 5
Nothing is being written to the application, but no errors are being displayed	The tool collapsed onto <code>ydotool</code> , but the user is not in the group input	Check group membership, reconnect – see Chapter 5
The application always pastes the last content copied manually; text dictated only appears in the terminal	No clipboard tools installed (<code>wl-clipboard</code> under Wayland, <code>xclip</code> under X11); the dictated text never reaches the clipboard	Install one. The Debian package $\geq 0.5.1$ depends on <code>wl-clipboard</code> and recommends <code>xclip</code> ; with a Git clone, install them by hand – see Chapter 5
Under Wayland, with a GTK application, the window temporarily pastes the previous content of the clipboard	The clipboard restoration occurs before the window has finished reading the selection	<code>clipboard.defer_restore: true</code> in the configuration – in this case, the restoration is delayed until the end of the session – see Chapter 5
No notification sound	<code>paplay</code> absent, or no sound file configured	Minor warning, no impact on recognition – install <code>pulseaudio-utils</code> if sound is desired
Can a single word (“start”) accidentally trigger the dictation feature?	No: The alarm requires the complete sentence	The alarm only triggers when “start dictation” is complete - nothing to correct
The wake-up phrase does not start the dictation	Active sleep mode (<code>Alt+W</code>), or incomplete sentence, or confidence below threshold	Exit sleep mode with <code>Alt+W</code> ; say the full phrase; repeat if necessary - see Chapter 3

Symptom	Probable cause	Action
“stop dictation” doesn’t stop the dictation, and nothing is typed	Stop phrase below confidence threshold — absorbed for security	Repeat it, or use Alt+S / <code>--finalize</code> — see Chapter 3
The wake-up or stop phrase is written in the text instead of being triggered	Repeat the phrase twice in a row - desired behavior	Say it only once, after a short silence—see Chapter 3
Alt+H / Alt+S / Alt+W are not working	Native Wayland window in the foreground - pynput only sees X11 and XWayland windows	Link <code>--toggle</code> , <code>--finalize</code> , <code>--sleep-toggle</code> to the desktop shortcuts – see Chapter 5
Alt+S inserts ^[s into the terminal	Observed behavior: the shortcut is not consumed by the tool	Prefer a desktop shortcut linked to <code>--finalize</code> — see Chapter 5
Alt+H does not copy anything to the clipboard	Normal behavior - Alt+H pauses, it doesn’t complete the process	Use Alt+S or the stop phrase to copy — see Chapter 1
One word is consistently misrecognized	Phonetic engine confusion (words that sound similar)	Add a correction in <code>recognition_aliases</code> — see Chapter 4
The tool ends the session automatically: “the recognition engine fails to recover.”	The decoder returned unreadable frames several times in a row, and the engine reconstruction didn’t change anything	Restart the tool. If this happens repeatedly, attach the file saved under <code>~/.local/state/vosk-cli-dictation/diagnostics/</code> (see below) to a report
The modified configuration never seems to be applied	The edited file is not the one actually being used (there are multiple possible locations)	Check the order of operations (resolution) - see Chapter 4

Where can I find a complete error message?

`vosk-cli-dictation` displays its errors directly in the terminal where it was launched. Relaunching from a visible terminal (not in the background) is the first thing to do to diagnose a problem not listed above. If you are unsure about the environment, `vosk-cli-dictation --diagnose` checks everything (tools, models, microphone, clipboard, control socket) and displays the status of each item.

Only in a single case is a file written: when the decoder returns an unreadable output, the raw string is saved once per session under `~/.local/state/vosk-cli-dictation/diagnostics/`. This file is useful to include in a speech recognition issue report.

If the problem persists

The project repository has a public issue tracking page: <https://github.com/RonanDavalan/vosk-cli-dictation/issues>