

vosk-cli-dictation — Benutzerhandbuch

Offline-Sprachdiktat installieren, konfigurieren und verwenden

Version v0.5.4-beta

2026-09-18

Inhaltsverzeichnis

Schnellstart	3
Kapitel 1 — Starten	4
Voraussetzungen	4
Es gibt mehrere Möglichkeiten, dies zu installieren	4
Ein Sprachmodell abrufen	5
Erster Start	5
Das erste Diktat	6
Zeichensetzung diktieren	6
Deinstallieren und aktualisieren	6
Kapitel 2 — Die Spracherkennungs-Engine	8
Für wen ist dieses Werkzeug?	8
Was Vosk macht	8
Akustisches Modell und Sprachmodell	9
Nichts verlässt die Maschine	9
Kapitel 3 — Die drei Zustände: ÉCOUTE, DICTÉE, VEILLE	10
Überblick	10
DICTÉE — Diktat im Gange	10
ÉCOUTE — in Wartestellung	10
Wie man eine Diktierfunktion startet und stoppt	10
VEILLE — nahezu kein Verbrauch, freiwillig	11
Der Konfidenzschwellenwert steuert die Auslösung	11
Wann sollte man den Energiesparmodus verwenden?	12
Der Ablauf einer kompletten Sitzung	12
Kapitel 4 — Konfiguration und Anpassung	13
Wo befindet sich die Konfigurationsdatei?	13

Die Sprache der Konfigurationsvorlage auswählen	13
Tastenkombinationen anpassen	13
Wählen Sie, wie die Diktierfunktion startet und stoppt	14
Zeichensetzung: zwei Eingabemodi, je nach Sprache	14
Den Konfidenzschwellenwert einstellen	15
Vokabeln hinzufügen und Unklarheiten beseitigen	15
Wo werden Sprachmodelle gesucht?	15
Kapitel 5 — Systemintegration: Wayland und X11	17
Wie gelangt der Text in die Anwendung?	17
Das Mikrofon	17
Die verwendeten Systemwerkzeuge und deren automatische Auswahl	17
Aktivieren Sie ydotool (Ausweichlösung)	18
Tastenkombinationen: X11 und Wayland sind nicht gleichwertig	18
Was passiert, wenn kein Injektionswerkzeug verfügbar ist?	19
Kapitel 6 — Fehlerbehebung	20
Symptome → wahrscheinliche Ursache → Handlung	20
Wo finde ich eine vollständige Fehlermeldung?	22
Wenn das Problem weiterhin besteht	22

Schnellstart

Für diejenigen, die in zwei Minuten diktieren möchten, ohne den Rest sofort zu lesen.

vosk-cli-dictation ist für ressourcenbeschränkte Systeme und Offline-Diktierfunktionen gedacht, nicht für maximale Genauigkeit auf einem leistungsstarken Arbeitsplatzrechner (Kapitel 2). Die unten aufgeführten Tastenkombinationen setzen eine X11-Session voraus; unter Wayland werden sie über die Desktop-Umgebung abgewickelt (Kapitel 5).

1. Installation. Nativ in der Distribution enthaltenes Paket:

```
# 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. Ein Sprachmodell herunterladen, falls noch keines vorhanden ist (das Paket enthält diese nicht – sie wären zu groß):

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

Das kleine Sprachmodell (hier `fr`, auch `en`, `de`, `es`) wird heruntergeladen, durch seine SHA256-Summe überprüft und in `~/.local/share/vosk-models/` installiert. Beim ersten Start ohne ein Modell bietet das Tool diesen Download interaktiv an. Man kann auch manuell ein Modell von <https://alphacephei.com/vosk/models> herunterladen und in diesem Verzeichnis entpacken.

3. Diktat.

```
vosk-cli-dictation
```

Sagen Sie den Aktivierungssatz (“diktat starten”), um die Diktierfunktion zu starten, sprechen Sie, und sagen Sie dann den Abbruchsatz (“diktat stoppen”), um sie zu beenden, und kopieren Sie den Text in die Zwischenablage.

Das ist alles für den Anfang. Der Rest erklärt, warum und wie man weiter vorgeht – die Tastenkombinationen anpassen, den Energiesparmodus verstehen, Probleme beheben.

Kapitel 1 — Starten

Voraussetzungen

- **Ein Linux-System** mit einer grafischen X11- oder Wayland-Session. Ohne einen Anzeigeserver gibt es kein aktives Fenster, in das der erkannte Text geschrieben werden kann.
- **Python 3** und eine virtuelle Umgebung (`python3-venv`).
- **Ein Textinjektionswerkzeug**: `wtype` oder `xdotool` unter Wayland, `xdotool` unter X11. Mindestens eines muss vorhanden sein, andernfalls lehnt das Werkzeug den Start ab (Kapitel 5).
- **Ein vom System erkanntes Mikrofon** als Standardeingabegerät (Kapitel 5).
- **Festplattenspeicher für das Modell**: von 50 MB (kleine Modelle) bis zu mehr als einem Gigabyte. Während des Betriebs verwendet eine Session etwa 300 MB RAM (Kapitel 2).

Das Debian-Paket deklariert und installiert alle seine Abhängigkeiten selbst, sowohl für das System als auch für Python. Im Git-Clone-Modus installiert `pip install -r requirements.txt` die Python-Bibliotheken; das Injektionswerkzeug (`wtype`, `xdotool` oder `ydotool`) muss dann separat installiert werden.

Es gibt mehrere Möglichkeiten, dies zu installieren

`vosk-cli-dictation` bietet mehrere Kanäle, wobei keiner den anderen ersetzt.

Das native Paket der Distribution (empfohlen für den normalen Gebrauch)

```
# 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
```

Unabhängig vom Format installiert das Paket einen einzelnen Befehl, `vosk-cli-dictation`, in `/usr/bin/`. Es erstellt außerdem eine isolierte Python-Umgebung und installiert darin die Spracherkennungs-Engine (Vosk) – dieser Schritt benötigt während der Installation eine Netzwerkverbindung, und zwar nur einmal. Nach der Installation sind keine Berechtigungen als Superuser erforderlich.

Die Installation selbst ändert niemals die persönlichen Dateien: keine bestehenden Konfigurationen werden überschrieben, und kein Benutzerverzeichnis wird erstellt oder verändert. Erst beim ersten Start und beim Laden eines Sprachmodells (siehe unten) werden Dateien im persönlichen Verzeichnis erstellt – niemals vorher und niemals ohne eine explizite Aktion des Benutzers.

Der Git-Clone (zum Verfolgen der Entwicklung oder zur Mitarbeit)

```
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 diesem Modus wird das Tool mit `python3 src/main.py` gestartet, anstatt über den Befehl `vosk-cli-dictation` des Pakets, und seine Konfiguration befindet sich in `config/config.yaml`, neben dem Code – niemals im persönlichen Verzeichnis. Die beiden Modi vermischen sich nicht: ein Git-Klon ignoriert die Konfiguration des Pakets, und umgekehrt.

Ein Sprachmodell abrufen

Keiner der beiden Kanäle enthält ein Spracherkennungsmodell – diese Modelle wiegen zwischen 50 MB und mehr als einem Gigabyte, was zu viel ist, um sie mit dem Tool zu liefern.

Die einfachste Methode bei der Installation über ein Paket ist, dem Tool die Aufgabe zu überlassen.

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

Dieser Befehl lädt das kleine Modell der angeforderten Sprache (`fr`, oder `en`, `de`, `es`), überprüft dessen SHA256-Summe, installiert es in `~/.local/share/vosk-models/` und wird dann beendet. Beim allerersten Start ohne jegliches Modell wird der gleiche Vorschlag interaktiv angezeigt: Mit “Ja” wird das kleine Sprachmodell der Benutzeroberfläche heruntergeladen.

Alternativ, oder für ein umfassenderes Modell als das kleine Standardmodell, laden Sie das Modell manuell von <https://alphacephei.com/vosk/models> herunter und entpacken Sie es:

- Installation per Paket: in `~/.local/share/vosk-models/`.
- Git-Klon: in `vosk-model/`, im Root-Verzeichnis des geklonten Repositoriums.

Ohne ein entsprechendes Modell am richtigen Ort zeigt das Tool beim Start die Orte an, an denen es gesucht hat, und die Download-Adresse – es kommt niemals zu einem stillen Fehler.

Erster Start

```
vosk-cli-dictation
```

(Im Klon-Modus von Git ist der gleiche Befehl `python3 src/main.py`, ausgehend vom Stammverzeichnis des Repositories und mit aktivierter virtueller Umgebung)

Drei Dinge werden angezeigt: die Überprüfung der externen Abhängigkeiten (siehe Kapitel 5), das geladene Modell und eine Übersicht über die verfügbaren Befehle – Befehle zum Starten und Stoppen, Tastenkombinationen, Satzzeichen. Diese Übersicht wird bei jedem Start aus der aktuellen Konfiguration neu generiert; sie spiegelt immer wider, was tatsächlich verwendbar ist, und ist keine in der Dokumentation festgelegte Liste, die nach einer Anpassung möglicherweise veraltet sein könnte.

Das erste Diktat

Zwei äquivalente Möglichkeiten, eine Diktat zu beginnen, zur Auswahl:

- **Per Stimme:** Sprechen Sie den Aktivierungssatz der aktiven Sprache („diktat starten“). Der vollständige Satz ist erforderlich: Ein einzelnes Wort („diktat“) löst nichts aus – die Aktivierung wird nur für den gesamten Satz mit ausreichender Sicherheit erkannt.
- **Per Tastatur:** Drücken Sie **Alt+H** – dieser Tastenkombination wechselt zwischen ÉCOUTE (dem Ausgangszustand, der Zustand direkt nach dem Start) und DICTÉE (siehe Kapitel 3 für die Zustände); er dient sowohl zum Starten als auch zum Pausieren. Er kopiert niemals etwas in die Zwischenablage: siehe unten, um eine Diktierfunktion zu beenden. Unter einer **Wayland**-Session werden diese Tastenkombinationen nur für Fenster erfasst, die über XWayland laufen; für ein natives Wayland-Fenster müssen sie über die Desktop-Oberfläche erfolgen (Kapitel 5).

Sprich anschließend normal. Der erkannte Text wird direkt in der Anwendung angezeigt, die den Fokus hat – sei es ein Terminal, ein Editor, ein Browser oder etwas anderes.

Um die Eingabe zu beenden und den Text in die Zwischenablage zu übertragen: Sagen Sie den Abbruchbefehl („diktat stoppen“) oder drücken Sie **Alt+S**. Ein Abbruchbefehl, der unterhalb der Konfidenzschwelle liegt, wird akzeptiert – er wird weder in das Dokument eingegeben noch ausgeführt, und ein Hinweis wird angezeigt: Wiederholen Sie den Befehl oder verwenden Sie **Alt+S**.

Eine zuverlässigere Steuerung, sowohl unter X11 als auch unter Wayland, besteht darin, die Unterbefehle (`--toggle`, `--finalize`, `--sleep-toggle`) an Desktop-Verknüpfungen (Kapitel 5) zu binden, anstatt die integrierten Verknüpfungen zu verwenden.

Zeichensetzung diktieren

Die Satzzeichen werden wie der restliche Text eingegeben – „Komma“, „Ausrufezeichen“, „neue Zeile“, usw. Die vollständige Liste hängt von der aktiven Sprache ab und wird beim Start angezeigt (siehe oben); sie ist außerdem anpassbar, und die Art und Weise, wie ein Satzzeichenbefehl ausgelöst wird, kann je nach Eingabemodus der Sprache variieren (Kapitel 4).

Deinstallieren und aktualisieren

Nativem Paket. Der Paketmanager entfernt den Befehl und die isolierte Python-Umgebung (`/usr/lib/vosk-cli-dictation/venv`): `sudo apt remove vosk-cli-dictation` (Debian 12+ und Derivate; `sudo apt purge vosk-cli-dictation` löscht zusätzlich das Werkseinstellungen-Konfigurationsmodell), `sudo dnf remove vosk-cli-dictation` (Fedora), `sudo zypper remove vosk-cli-dictation` (openSUSE), `sudo pacman -R vosk-cli-dictation` (Arch Linux). Die persönliche Konfigurationsdatei (`~/.config/vosk-cli-dictation/config.yaml`) und die Sprachmodelle (`~/.local/share/vosk-models/`) werden durch die Deinstallation niemals verändert – sie können bei Bedarf manuell gelöscht werden.

Git-Repository klonen. Den geklonten Ordner löschen: er enthält alles (Code, virtuelle Umgebung `venv/`, eventuelle Vorlagen in `vosk-model/`). Nichts wird anderswo in diesem Modus gespeichert.

Update. Mit einem nativen Paket wird die neueste Datei mit dem gleichen Befehl wie bei der ersten Installation installiert (`apt install`, `dnf install`, `zypper install`, `pacman -U`): sie ersetzt die vorhandene Version, ohne die persönlichen Einstellungen zu verändern. Bei einem Git-Clone: `git pull`, und dann `pip install -r requirements.txt` im aktivierten virtuellen Umfeld erneut ausführen, um eventuelle neue Abhängigkeiten zu berücksichtigen.

Kapitel 2 — Die Spracherkennungs-Engine

Für wen ist dieses Werkzeug?

Bevor wir uns den technischen Details zuwenden, eine ehrliche Erklärung: “vosk-cli-dictation” ist nicht dafür gedacht, mit den präzisesten Spracherkennungs-Engines auf dem Markt auf einem modernen Desktop-Computer zu konkurrieren. Das ist keine versteckte Schwäche – es ist eine bewusste Designentscheidung, die das Tool in einem bestimmten Bereich hervorragend macht:

- **Eingeschränkte Hardware** — Raspberry Pi, alter Laptop, Maschine ohne GPU. Der Motor benötigt niemals Hardwarebeschleunigung. Im laufenden Betrieb verbraucht eine Sitzung ungefähr **300 MB RAM** (209 bis 383 MB, abhängig von der Sprache und dem Gerät — gemessene Modelle: `small`, `vosk-model-small-fr-0.22`, `-en-us-0.15`, `-de-0.15`, `-es-0.42` — ein bescheidener Wert im Vergleich zu einem Browser, aber nicht unerheblich auf einer Maschine mit wenig Speicher.
- **Echtzeit-Reaktivität** — Der Text erscheint während des Sprechens, ohne auf das Ende eines Segments warten zu müssen. Dies ist eine Eigenschaft der Architektur der Engine (siehe unten) und keine Einstellung, die man durch Ändern eines Parameters verlieren könnte. Die Dekodierung verfolgt die Sprache nicht mit Verzögerung: die Verarbeitung eines Audioblocks dauert weniger Zeit als die Dauer des Blocks selbst (Echtzeitfaktor $RTF < 1,0$), auch im anspruchsvollsten Fall: die Diktierfunktion auf Französisch auf einem Raspberry Pi 4 (RTF 0,73) oder auf einem AMD A4-6210 (RTF 0,79). Französisch ist die teuerste Sprache zum Dekodieren; auf der gleichen bescheidenen Hardware laufen Englisch, Deutsch und Spanisch mit einem RTF von 0,20–0,33.
- **Keine Daten verlassen das Gerät** — keine Audio- oder Textdaten werden jemals an einen externen Dienst übertragen, weder während der Erkennung noch zu anderen Zeiten.
- **Minimale Abhängigkeitskette** — keine schweren Bibliotheken für maschinelles Lernen müssen installiert werden, kein Grafikkartentreiber ist erforderlich.

Im Gegenzug ist das Verständnis der Bedeutung begrenzt: Zwei Wörter, die gleich klingen, aber unterschiedlich geschrieben werden („das“/„dass“, „Meer“/„mehr“), werden nicht immer korrekt anhand des reinen Klangs unterschieden. Kapitel 4 erklärt, wie diese Fälle einer nach dem anderen korrigiert werden können, sobald sie durch die Nutzung erkannt wurden – dies geschieht nicht automatisch.

Wenn das Ziel darin besteht, lange Texte mit höchster Genauigkeit auf einem leistungsstarken Desktop-Computer zu verfassen, ist ein anderes Tool besser geeignet. Wenn das Ziel jedoch darin besteht, auf einfacher Hardware in Echtzeit zu diktieren, ohne jemals auf ein Netzwerk angewiesen zu sein, ist vosk-cli-dictation genau dafür konzipiert.

Was Vosk macht

Vosk ist die Spracherkennungs-Engine, die von diesem Tool verwendet wird – eine unabhängige Bibliothek, kein selbst entwickeltes Bauteil. Sie basiert auf einer älteren Architektur als die auf Transformer-Modellen basierenden Engines (wie Whisper): ein akustisches Modell in Kombination mit einem statistischen Sprachmodell, ähnlich wie die Software Kaldi.

Diese Architektur hat eine direkte und nützliche Auswirkung: Die Transkription erfolgt in Echtzeit, im Takt, in dem der Ton vom Mikrofon empfangen wird, wobei die Ergebnisse schrittweise angezeigt werden – nicht in Blöcken von mehreren Sekunden, auf die man warten müsste. Das erzeugt das Gefühl einer “Live”-Diktierfunktion.

Akustisches Modell und Sprachmodell

Jedes heruntergeladene Modell (Kapitel 1) besteht aus zwei Teilen, die bei der Nutzung nicht sichtbar sind, aber für das Verständnis hilfreich sind:

- **Das akustische Modell** erkennt Sprache und ordnet sie sprachlichen Einheiten (Phonemen) zu.
- **Das Sprachmodell** bewertet, welche Wortfolgen in der betreffenden Sprache statistisch wahrscheinlich sind, um zwischen verschiedenen möglichen Interpretationen desselben Klangs zu entscheiden.

Die vom Vosk-Projekt angebotenen Modelle gibt es in verschiedenen Größen. `vosk-cli-dictation` verwendet standardmäßig die kompakten Modelle (“small”, etwa 50 MB auf der Festplatte) – ein guter Kompromiss für Systeme mit begrenzten Ressourcen. Es steht jedoch nichts dagegen, in der Konfiguration (Kapitel 4) ein größeres Modell anzugeben, wenn das Zielsystem dies zulässt, was jedoch einen größeren Speicherbedarf bedeutet.

Nichts verlässt die Maschine

Keine der Schritte der Spracherkennung benötigt eine Netzwerkverbindung. Der einzige Zeitpunkt, an dem eine Verbindung erforderlich ist, ist bei der erstmaligen Installation des Pakets (Download der Python-Engine) und beim manuellen Herunterladen eines Sprachmodells – niemals während der normalen Nutzung.

Kapitel 3 — Die drei Zustände: ÉCOUTE, DICTÉE, VEILLE

vosk-cli-dictation hat drei verschiedene Zustände, die jeweils unterschiedliche CPU-Kosten verursachen. Verwechslungen zwischen diesen Zuständen sind die häufigste Ursache für Probleme bei der Verwendung (“Warum funktioniert das Weckphrase nicht mehr?”) – dieses Kapitel unterscheidet die Zustände klar voneinander.

ÉCOUTE, DICTÉE und VEILLE sind die Bezeichnungen, die das Programm auf dem Bildschirm anzeigt. Sie bleiben auf Französisch, unabhängig von der Sprache, in der der Text diktiert wird: sie identifizieren einen internen Zustand, nicht einen zu übersetzenden Text.

Überblick

Zustand	Wie komme ich da rein?	Mikroaktiv	Dekodierung	Weckphrase
DICTÉE	Alt+H (oder Aktivierungsphrasen), seit ÉCOUTE	Ja	Ja	– (Diktat läuft bereits)
ÉCOUTE	Alt+H, seit DICTÉE	Ja	Ja	Ja
VEILLE	Strg+W	Nein	Nein	Nein

DICTÉE — Diktat im Gange

Was Sie sagen, wird erkannt und in der aktiven Anwendung eingegeben. Um den Vorgang abzuschließen und das Ergebnis in die Zwischenablage zu kopieren: Sagen Sie den Abbruchbefehl (“diktat stoppen”), oder drücken Sie Alt+S.

ÉCOUTE — in Wartestellung

Dies ist der Standardzustand zwischen zwei Diktaten. Das Mikrofon bleibt aktiv und die Engine hört weiterhin, und das aus nur einem Grund: um jederzeit die Aktivierungsphrase erkennen zu können. Der CPU-Verbrauch dieses Modus ist also nicht Null – das ist der Preis für die sprachgesteuerte Reaktionsfähigkeit. Um diesen zu reduzieren, erfolgt die Decodierung in diesem Zustand mit einer eingeschränkten Grammatik, die sich nur auf die Aktivierungsphrase beschränkt: Der Motor sucht während des Wartens nichts anderes, was die Arbeit des Prozessors deutlich reduziert.

Die Weckphrase ist nicht nur ein einzelnes Wort: Es ist die **vollständige Phrase** erforderlich, und die Engine speichert sie nur, wenn die Erkennungssicherheit ausreichend ist. Ein einzelnes Wort (wie „diktat“) wird niemals als Weckphrase interpretiert – dies ist ein Schutz vor ungewollten Auslösungen während eines Gesprächs.

Wie man eine Diktierfunktion startet und stoppt

Drei Möglichkeiten, den Übergang ÉCOUTE → DICTÉE → ÉCOUTE zu steuern, einstellbar über `interaction.mode` in der Konfiguration (Kapitel 4):

- **toggle** (standardmäßig, unverändert): die oben beschriebene – eine Geste startet (**Alt+H** oder Aufruf-Phrase), eine andere Geste stoppt (**Alt+S** oder Stopp-Phrase).
- **maintenue** (Push-to-Talk): die Diktierfunktion läuft genau so lange, wie die unter `hotkeys.push_to_talk` konfigurierte Taste (eine einzelne Taste, keine Kombination) gedrückt gehalten wird – beim Loslassen stoppt die Sitzung und wird automatisch kopiert.
- **silence**: Ein Druck (oder die Aufruf-Phrase) startet; die Diktierfunktion stoppt und wird automatisch kopiert, nachdem eine Pause von `interaction.silence_timeout_ms` (standardmäßig 700 ms, sprachabhängig einstellbar) erfolgt ist, sobald mindestens ein Wort erfasst wurde. Dies ist der einzige der drei Modi, der die Hände während des gesamten Satzes vollständig frei lässt.

In allen drei Modi bleiben die Sprachausgabe und **Alt+S** / `--finalize` gültige Sicherheitsmechanismen – kein Modus entfernt eine bestehende Notfallfunktion.

VEILLE – nahezu kein Verbrauch, freiwillig

Ein dritter Zustand, der explizit durch **Alt+W** aktiviert wird, niemals durch Sprache (aus konstruktiven Gründen: das Mikrofon ist in diesem Zustand deaktiviert, sodass keine Sprache gehört werden kann). Audioaufnahme und Spracherkennung werden beide unterbrochen – die CPU-Auslastung sinkt auf **0,0 %** und das Mikrofon wird freigegeben.

Was es kostet: Die Weckphrase funktioniert nicht mehr, solange der Ruhezustand aktiv ist. Das ist eine bewusste Entscheidung, kein versteckter Fehler – das System zeigt dies explizit an, wenn der Ruhezustand aktiviert wird. Nur **Alt+W** bleibt während des Ruhezustands aktiv (es ist nicht vom Mikrofon abhängig), um den Ruhezustand zu beenden.

Beim Umschalten in den Ruhezustand während einer laufenden Diktierfunktion wird diese ordnungsgemäß beendet, ähnlich wie wenn **Alt+H** gedrückt würde – kein Text geht dabei innerhalb eines Satzes verloren.

Der Konfidenzschwellenwert steuert die Auslösung

Der Weckalarm wird nur ausgelöst, wenn der durchschnittliche Konfidenz der Wörter im Satz den konfigurierten Schwellenwert erreicht (standardmäßig 0,55, einstellbar pro Sprache – Kapitel 4). Konkrete Auswirkungen:

- Ein Stoppbefehl, der unterhalb des Schwellenwerts erkannt wird, wird **unterdrückt**: er wird weder in das Dokument eingegeben noch ausgelöst – stattdessen wird ein Hinweis angezeigt, und es reicht aus, ihn zu wiederholen (oder **Alt+S** zu verwenden). Dieser Mechanismus schützt das Dokument: ein normaler Satz, der „diktat stoppen“ ähnelt, wird niemals eingegeben.
- Ein Stopp- oder Startbefehl, der **zweimal hintereinander** ausgesprochen wird, wird wörtlich in den Text eingegeben, anstatt ausgelöst zu werden – dies ist beabsichtigt: eine unmittelbare Wiederholung ist in Sprachsteuerung selten, und dieses Verhalten verhindert, dass eine Wiederholung verloren geht.

Wann sollte man den Energiesparmodus verwenden?

Nützlich jedes Mal, wenn das Mikrofon für eine längere Zeit stummgeschaltet bleiben muss, ohne das Programm zu verlassen – beispielsweise bei einem Meeting, einem Anruf oder wenn die CPU-Auslastung wichtig ist (z. B. bei einem Laptop mit Akku oder bei leistungstechnisch eingeschränkter Hardware). Auf einem Computer, bei dem die CPU-Auslastung keine große Rolle spielt, ist es vollkommen in Ordnung, dauerhaft im Modus ÉCOUTE zu bleiben: das ist eine Wahl, keine Notwendigkeit zur Optimierung.

Der Ablauf einer kompletten Sitzung

1. **Starte** das Programm in einem sichtbaren Terminal. Es beginnt in ÉCOUTE.
2. **Öffne die Zielanwendung** (Editor, Browser, Terminal) und gib ihr den Tastaturfokus.
3. **Wechsle zu DICTÉE**: Sage den Aktivierungssatz oder drücke **Alt+H**.
4. **Sprich normal**. Der Text wird in Echtzeit in der Anwendung eingegeben, und Interpunktionszeichen („Komma“, „Ausrufezeichen“, „neue Zeile“...) werden entsprechend interpretiert (Kapitel 4).
5. **Mache eine Pause**, falls nötig: **Alt+H** wechselt zurück zu ÉCOUTE ohne etwas zu kopieren; der bereits eingegebene Text bleibt in der Anwendung. Drücke **Alt+H** erneut, um fortzufahren.
6. **Beende**: Sage den Deaktivierungssatz, drücke **Alt+S**, oder binde `--finalize` an eine Desktop-Tastenkombination (Kapitel 5). Der vollständige Text der Sitzung wird dann in die Zwischenablage kopiert.
7. **Füge ein** (Ctrl+V) dort ein, wo du möchtest – die ursprüngliche Zwischenablage wird unmittelbar danach wiederhergestellt.
8. **Schalte in den Energiesparmodus** zwischen zwei Anwendungen: **Alt+W** (Mikrofon deaktiviert, 0 % CPU). Drücke **Alt+W** erneut, um ihn aufzuwecken.
9. **Beende**: Strg+C im Startterminal.

Kapitel 4 – Konfiguration und Anpassung

Das gesamte Verhalten von vosk-cli-dictation wird über eine einzige YAML-Datei gesteuert, `config.yaml`. Dieses Kapitel erklärt, wo sich diese Datei befindet und was darin geändert werden kann.

Wo befindet sich die Konfigurationsdatei?

Der Speicherort hängt vom Installationspfad ab (Kapitel 1) und folgt einer bestimmten Prioritätsreihenfolge. Der erste gefundene Speicherort wird verwendet, die folgenden werden niemals berücksichtigt.

1. Die Umgebungsvariable `VOSK_CONFIG`, falls definiert – wird unverändert verwendet, wird niemals automatisch erstellt.
2. `config/config.yaml` des Repositories, im Git-Clone-Modus.
3. `~/.config/vosk-cli-dictation/config.yaml`, bei Paketinstallation – wird beim ersten Start automatisch aus einer Vorlage erstellt, zusammen mit einer Meldung, die angibt, wo die Datei erstellt wurde.
4. Die Vorlage selbst, schreibgeschützt.

In der Praxis: Mit dem Debian-Paket, ändern Sie `~/.config/vosk-cli-dictation/config.yaml` – dies ist eine persönliche Datei, die niemals durch ein Paket-Update überschrieben wird.

Die Sprache der Konfigurationsvorlage auswählen

Das Debian-Paket enthält vier lokalisierte Vorlagen – `config.en.yaml`, `config.fr.yaml`, `config.de.yaml`, `config.es.yaml`. Sie befinden sich unter `/usr/share/vosk-cli-dictation/`.

Beim ersten Start wird die Sprachdatei, die der lokalen Systemeinstellung entspricht (`LANGUAGE` bzw. `LANG`, und falls keine der vier übereinstimmt, wird auf Englisch zurückgegriffen), nach `~/.config/vosk-cli-dictation/config.yaml` kopiert.

Um nach diesem ersten Start zu einer anderen Vorlage zu wechseln, müssen Sie diese mit `dpkg -L vosk-cli-dictation | grep config` identifizieren und dann ihre persönliche Datei durch die gewünschte Datei ersetzen:

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

Im Git-Clone-Modus befinden sich die vier Vorlagen direkt in `config/` des Repositorys.

Tastenkombinationen anpassen

Abschnitt `hotkeys:` der Datei. Jeder Tastenkombination wird durch eine Liste von Tasten `pynput_keys` und einen angezeigten Namen `display_name` definiert:

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

Drei Verknüpfungen sind hier konfigurierbar – `toggle_recording` (Kapitel 1), `finalize_session` und `sleep_toggle` (Kapitel 3). `Ctrl+C`, das das Programm beendet, gehört nicht dazu: es ist das Standard-Interrupt-Signal des Systems, das immer aktiv ist und niemals aus dieser Datei heraus neu konfiguriert werden kann.

Unter Wayland erreichen diese Tastenkombinationen nur Fenster, die über XWayland laufen (siehe Kapitel 5). Für native Wayland-Fenster ist es die Desktop-Umgebung, die die Tastenkombinationen verwaltet, und zwar über Unterbefehle zur Steuerung.

Eine vierte Eingabe, `push_to_talk`, hat nur eine einfache Taste (keine Kombination): sie dient dem “Push-to-Talk”-Modus (Kapitel 3) und hat in den beiden anderen Modi keine Funktion.

Wählen Sie, wie die Diktierfunktion startet und stoppt

Abschnitt `interaction`: der Datei (Kapitel 3 zur Beschreibung der drei Modi):

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

`silence_timeout_ms` Diese Überschlagung erfolgt pro Sprache genau wie der unten angegebene Konfidenzschwellenwert, unter `language_settings.<lang>.silence_timeout_ms` – nützlich, wenn ein Modell auf einer bestimmten Maschine langsamer decodiert und dadurch eine zu kurze Zeitüberschreitung ausgelöst wird.

Zeichensetzung: zwei Eingabemodi, je nach Sprache

Jede Sprache definiert, unter `language_settings.<lang>.punctuation`, ihre Liste von Satzzeichen und ihre Eingabemethode:

- **mode: inline** (standardmäßig) – Interpunktionszeichen werden erkannt und direkt während der Diktierphase ersetzt: das Sagen von „Komma“ fügt ein Komma ein.
- **mode: explicit** – Ein Interpunktionsbefehl wird nur angewendet, wenn er unmittelbar von dem sprachspezifischen Auslösewort vorangestellt wird: um ein Komma zu erhalten, muss man „satzzeichen Komma“ sagen, während „Komma“ allein vollständig ausgeschrieben wird. Dies ist nützlich, wenn die Sprache oder der Wortschatz häufig zu falschen Auslösungen führt.

Die tatsächliche Liste der Wörter (einfache und doppelte Satzzeichen, Zeilenumbrüche, Sonderzeichen – Bindestriche, Klammern, Anführungszeichen...) wird von dem Programm beim Start angezeigt; sie ist in den Abschnitten `simple`, `double`, `layout` und `special` definiert. Die Formatierung, die auf den Text angewendet wird (die deutschen Anführungszeichen „ ... “ und die Tatsache, dass kein Leerzeichen vor ? ! : ; eingefügt wird...) wird durch den Block `typography` der Sprache festgelegt.

Den Konfidenzschwellenwert einstellen

Der Start und das Stoppen werden nur dann berücksichtigt, wenn das durchschnittliche Vertrauen der Wörter im Satz den Schwellenwert `recognition.confidence_threshold` erreicht (standardmäßig 0,55, Kapitel 3). Er kann pro Sprache überschrieben werden:

```
language_settings:
  de:
    confidence_threshold: 0.55
```

Ein höherer Schwellenwert führt zu einer strengeren Auslösung (weniger falsch-positive Ergebnisse, aber häufiger unterbrochene Sätze); ein niedrigerer Schwellenwert bewirkt das Gegenteil.

Vokabeln hinzufügen und Unklarheiten beseitigen

Zwei unterschiedliche Mechanismen, unter `language_settings.<lang>`:

- **custom_vocabulary** — Groß-/Kleinschreibungs-sensible Korrekturen, die nach allen anderen Schritten angewendet werden. Nützlich für Abkürzungen oder Eigennamen, die die Engine immer in Kleinbuchstaben schreibt:

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

- **recognition_aliases** — behebt eine häufig auftretende phonetische Verwechslung im Motor (zwei Wörter, die fast gleich klingen, Kapitel 2). Zum Beispiel, wenn die Engine systematisch „dass“ anstelle von „das“ erkennt:

```
recognition_aliases:
  "dass": "das"
```

Diese beiden Einstellungen werden erst durch die Anwendung entdeckt, nicht im Voraus – diktieren, eine wiederkehrende Verwirrung erkennen und hinzufügen.

Wo werden Sprachmodelle gesucht?

Wie bei der Konfiguration folgt die Suche einer bestimmten Reihenfolge, wobei das zuerst gefundene Verzeichnis auf der Festplatte Vorrang hat:

1. Ein absoluter Pfad – obwohl der Name es anders vermuten lässt, kann der Schlüssel `path_relative` auch einen vollständigen absoluten Pfad enthalten; in diesem Fall hat er Vorrang vor allem anderen, was unten steht.
2. `~/local/share/vosk-models/` (oder `$XDG_DATA_HOME/vosk-models/`, falls diese Variable definiert ist).
3. `/usr/share/vosk-models/` – systemweit installierte Modelle.
4. `vosk-model/` im Stammverzeichnis des Projekts, nur im Git-Clone-Modus.

Wenn kein Modell gefunden wird, listet die beim Start angezeigte Meldung genau diese Orte und die Download-Adresse auf und erinnert an den Befehl `vosk-cli-dictation --download-model` – niemals ein Fehler ohne Erklärung. Dieser Befehl installiert das kleine Modell an dem oben genannten Ort 2 (`~/local/share/vosk-models/`), dem einzigen Ort, an dem ein ohne Administratorrechte heruntergeladenes Modell später wiedergefunden wird.

Kapitel 5 — Systemintegration: Wayland und X11

`vosk-cli-dictation` hat keine eigene grafische Benutzeroberfläche – es schreibt den erkannten Text direkt in die Anwendung, die den Fokus der Tastatur hat, unabhängig davon, welche Anwendung dies ist. Dieses Kapitel erklärt, wie dies funktioniert und was sich je nach verwendetem Anzeigesystem ändert.

Wie gelangt der Text in die Anwendung?

Im Gegensatz zu dem, was man vielleicht vermuten könnte, wird der Text **nicht Zeichen für Zeichen eingegeben**. Der tatsächliche Mechanismus: Der erkannte Text wird in die Zwischenablage kopiert, und dann wird ein Tastenkürzel zum “Einfügen” (Ctrl+V) simuliert. Das ist zuverlässiger und schneller als eine vollständige Simulation der Eingabe, insbesondere bei längeren Textabschnitten.

Der ursprüngliche Textpuffer wird direkt nach diesem Einfügen (Strg+V) wiederhergestellt – die Spracherkennung überschreibt also nicht, was zuvor hineinkopiert wurde, mehr als für einen Moment. Unter Wayland mit einer GTK-Anwendung kann diese Wiederherstellung erfolgen, bevor das Fenster das Lesen des Textpuffers abgeschlossen hat, wodurch der alte Inhalt eingefügt wird; die Einstellung `clipboard.defer_restore: true` verzögert die Wiederherstellung bis zum Ende der Spracherkennungs-Sitzung und vermeidet dieses Risiko.

Das Mikrofon

`vosk-cli-dictation` erfasst den Ton über das **Standard-Eingabegerät des Systems** (PulseAudio oder PipeWire). Es gibt keine Geräteauswahl in `config.yaml` : um ein anderes Mikrofon zu verwenden, muss das gewünschte Gerät als Standardeingabe eingestellt werden – in den Soundeinstellungen der Desktop-Umgebung, mit `pavucontrol`, oder über die Kommandozeile (`pactl set-default-source <nom>`, wobei die Liste mit `pactl list sources short` oder `arecord -l` angezeigt wird).

`vosk-cli-dictation --diagnose` öffnet einen Test-Stream auf diesem Gerät und gibt an, ob das Öffnen fehlschlägt – z. B. wegen eines fehlenden, unterbrochenen oder bereits von einer anderen Anwendung verwendeten Mikrofons.

Die verwendeten Systemwerkzeuge und deren automatische Auswahl

Die Injektion erfordert **zwei** Tools: einen Tastenanschlag-Simulator und ein Tool zur Zwischenablage. Diese werden automatisch anhand des Sitzungstyps ausgewählt, ohne dass eine Konfiguration erforderlich ist:

- **Unter Wayland:** `wtype` (Eingabe) und `wl-clipboard` (Zwischenablage), die das Wayland-Protokoll nativ unterstützen. Wenn `wtype` nicht vorhanden ist, erfolgt die Eingabe über `ydotool` – aber `wl-clipboard` bleibt erforderlich.
- **Unter X11:** `xdotool` (Eingabe) und `xclip` (Zwischenablage).

`ydotool` ist die letzte Möglichkeit für die Eingabe (es funktioniert in beiden Welten über `/dev/uinput`, ist aber von der Tastaturbelegung für seine Tastenkombinationen abhängig). Mindestens ein Eingabe-Simulator muss vorhanden sein – andernfalls startet das Tool nicht und gibt dies deutlich an.

Das Debian-Paket hängt von `wl-clipboard` ab und empfiehlt `xclip`; installieren Sie diese im Git-Clone manuell. Ohne ein Text-Editor-Tool erreicht der diktierter Text niemals die Zwischenablage: nichts wird in der Anwendung geschrieben, und ein `Ctrl+V` fügt den zuletzt manuell kopierten Inhalt ein. Der Befehl `vosk-cli-dictation --diagnose` prüft die Umgebung (Sitzungstyp, Tools, Vorlagen, Mikrofon, Zwischenablage, Kontroll-Socket) und zeigt an, was fehlt.

Aktivieren Sie ydotool (Ausweidlösung)

Dieser Abschnitt bezieht sich nur auf Systeme, die auf `ydotool` zurückgreifen, da es keine `wtype` (Wayland) oder `xdotool` (X11) gibt. Wenn `wtype` oder `xdotool` verfügbar ist, ist nichts von dem Folgenden erforderlich – insbesondere ist es unnötig, der Gruppe `input` beizutreten. Das Debian-Paket wendet die gleiche Regel an: seine Installationsmeldung bietet nur die Möglichkeit, der Gruppe `input` beizutreten, wenn die Rückwärtskompatibilität `ydotool` verwendet wird, und weist darauf hin, dass `xdotool` und `wtype` dies nicht benötigen.

Drei Bedingungen, die in der angegebenen Reihenfolge überprüft werden müssen, falls der Text nicht geschrieben werden kann:

1. **Das Paket `ydotool` ist installiert.** Unter Debian befindet es sich in `trixie-backports`, nicht im stabilen Zweig – aktivieren Sie dieses Repository gegebenenfalls vor der Installation.
2. **Der Dienst `ydotool` läuft.** `ydotool` benötigt diesen Hintergrunddienst, um zu funktionieren.
3. **Der Benutzer gehört zur Gruppe `input`.** Diese Gruppe gewährt Zugriff auf `/dev/uinput`, andernfalls schlägt `ydotool` für diesen Benutzer stillschweigend fehl:

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

Es ist erforderlich, sich ab- und wieder anzumelden, damit diese Gruppenänderung wirksam wird – ein einfacher Neustart des Terminals reicht nicht aus.

Tastenkombinationen: X11 und Wayland sind nicht gleichwertig

Die integrierten Tastenkombinationen **Alt+H**, **Alt+S**, **Alt+W** werden von dem Tool selbst überwacht über die Bibliothek `pynput`, die den X-Server beobachtet. Unter Wayland werden nur die Fenster abgefangen, die über **XWayland** laufen (viele Electron-Anwendungen und ältere), niemals ein natives **Wayland-Fenster**: für diese müssen die Tastenkombinationen der Desktop-Umgebung das Tool steuern.

Jedes Desktop-System kann eine Tastenkombination mit einem Befehl verknüpfen. Auf der Seite “`vosk-cli-dictation`” sind die zu verwendenden Unterbefehle zur Steuerung:

Gewollte Aktion	Unterbefehl
Starten / Pause der Diktierfunktion	<code>vosk-cli-dictation --toggle</code>
Das Diktat beenden und den Text abschreiben	<code>vosk-cli-dictation --finalize</code>
Energiesparmodus ein / aus	<code>vosk-cli-dictation --sleep-toggle</code>

Gewollte Aktion	Unterbefehl
Den aktuellen Status kennen	<code>vosk-cli-dictation --status</code>
Die aktuelle Diktierfunktion abbrechen, ohne etwas einzutippen	<code>vosk-cli-dictation --cancel</code>
Die Sitzung ordnungsgemäß beenden	<code>vosk-cli-dictation --stop</code>

Beispiel unter KDE Plasma: In *Systemeinstellungen* → *Tastaturkürzel* → *Benutzerdefinierte Tastenkombinationen*, erstellen Sie einen neuen Eintrag mit dem Namen „Befehl/URL“, dessen Befehl `vosk-cli-dictation --finalize` ist, und weisen Sie ihm eine Tastenkombination zu (z. B. **Meta+S**). Der Unterbefehl verbindet sich mit der aktuellen Sitzung, führt die Aktion aus und zeigt den resultierenden Status an (OK DICTÉE, OK ÉCOUTE, OK VEILLE).

Unter X11 sind diese Unterbefehle die zuverlässigste Methode: ein Tastenkürzel `pynput` wird nicht “aufgegriffen” – die Tastenkombination erreicht auch das angezielte Fenster, und wenn sie in einem Terminal gedrückt wird, wird eine “parasitäre” Sequenz eingefügt (z. B. `^[s` für **Alt+S**). Das Binden `--finalize` (oder `--toggle`, `--sleep-toggle`) an einen Desktop-Shortcut ermöglicht ein sauberes Beenden von jedem Fenster aus.

Was passiert, wenn kein Injektionswerkzeug verfügbar ist?

Das Werkzeug startet nicht und gibt dies explizit an, wobei es eine Installationsvorschlag macht. Es gibt keinen “Degradationsmodus” ohne Injektion: ohne `wtype/xdotool/ydotool`, gibt es keine zuverlässige Möglichkeit, den erkannten Text in eine andere Anwendung zu schreiben.

Diese Startprüfung betrifft nur den Schreibsimulator. Ein fehlendes Zwischenablage-Tool (`wl-clipboard` unter Wayland, `xclip` unter X11) verhindert nicht den Start, verhindert aber die Texterfassung: der diktierter Text erscheint nur im Terminal, und jedes `Ctrl+V` fügt den zuletzt manuell kopierten Inhalt ein. `vosk-cli-dictation --diagnose` meldet dies.

Kapitel 6 — Fehlerbehebung

Symptome → wahrscheinliche Ursache → Handlung

Symptome	Wahrscheinliche Ursache	Aktion
Das Programm startet nicht, eine Meldung erscheint im Zusammenhang mit den Injektionswerkzeugen	Kein Injektionswerkzeug (wtype/xdotool/ymdotool) ist installiert oder funktionsfähig	Installieren Sie eines davon – siehe Kapitel 5
“Beim Start konnte kein Spracherkennungsmodell geladen werden.”	Kein Modell an den erwarteten Stellen gefunden	Die Nachricht listet die gesuchten Speicherorte und die Download-URL auf – siehe Kapitel 1
Nichts wird erkannt, das Mikrofon scheint stumm zu sein	Schlechtes Standard-Eingabegerät oder Mikrofon, das auf Systemebene deaktiviert ist	Überprüfen / Ändern der Standardeinstellungen (Audioeinstellungen, pavucontrol); vosk-cli-dictation --diagnose – siehe Kapitel 5
--diagnose signalisiert das Fehlschlagen beim Öffnen des Audio-Streams	Ein anderes Programm verwendet das Gerät, oder die Berechtigungen für den Zugriff auf das Mikrofon sind nicht ausreichend	Schließen Sie die andere Anwendung; überprüfen Sie die Geräteberechtigungen – siehe Kapitel 5
Nichts wird in der App geschrieben, aber es wird keine Fehlermeldung angezeigt	Das Werkzeug wurde auf ydotool umgelegt, aber der Benutzer gehört nicht zur Gruppe input	Gruppenzugehörigkeit prüfen, ab- und wieder anmelden – siehe Kapitel 5
Die Anwendung fügt immer den zuletzt manuell kopierten Inhalt wieder ein; der diktierte Text erscheint nur im Terminal	Kein Clipboard-Tool installiert (wl-clipboard unter Wayland, xclip unter X11); der diktierte Text erreicht niemals das Zwischenablage-Puffer	Installieren Sie es. Das Debian-Paket ≥ 0.5.1 hängt von wl-clipboard ab und empfiehlt xclip; bei einem Git-Klon installieren Sie diese manuell – siehe Kapitel 5
Unter Wayland und mit einer GTK-Anwendung wird der Fensterinhalt kurzzeitig durch den alten Inhalt der Zwischenablage ersetzt Kein Benachrichtigungston	Das Aktualisieren des Bildschirms (der Papierkorb wird aktualisiert) erfolgt, bevor das Fenster die Auswahl fertig gelesen hat paplay Nicht gefunden oder kein Audio-Gerät konfiguriert	clipboard.defer_restore: true in der Konfiguration – die Wiederherstellung erfolgt dann am Ende der Sitzung – siehe Kapitel 5 Leichte Warnung, ohne Auswirkungen auf die Erkennung – installieren Sie pulseaudio-utils falls gewünschten Klang

Symptome	Wahrscheinliche Ursache	Aktion
Kann ein einzelnes Wort („diktat“) versehentlich eine Diktierfunktion auslösen?	Nein: Die Weckphrase erfordert den vollständigen Satz	Die Weckphrase funktioniert nur bei vollständigen „diktat starten“ – nichts zu korrigieren
Die Weckphrase startet die Diktierfunktion nicht	Aktiver Energiesparmodus (Alt+W), oder unvollständiger Satz, oder Konfidenz unterhalb des Grenzwerts	Den Ruhezustand mit Alt+W beenden; den vollständigen Satz aussprechen; bei Bedarf wiederholen – siehe Kapitel 3
„diktat stoppen“ stoppt die Diktierfunktion nicht, und nichts wird eingegeben	Sicherheitsbedingte Stilllegungsanweisung – automatisch aktiviert	Sie erneut ausführen, oder Alt+S / --finalize verwenden – siehe Kapitel 3
Der Aktivierungs- oder Deaktivierungscode wird im Text selbst geschrieben, anstatt ausgelöst zu werden	Satz zweimal hintereinander gesprochen – gewünschtes Verhalten	Es einmal sagen, nach einer kurzen Pause – siehe Kapitel 3
Alt+H / Alt+S / Alt+W funktionieren nicht	Fenster im nativen Wayland-Modus werden im Vordergrund angezeigt – pynput zeigt nur X11- und XWayland-Fenster	Binden Sie --toggle, --finalize, --sleep-toggle an die Verknüpfungen im Arbeitsbereich – siehe Kapitel 5
Alt+S fügt ^[s im Terminal ein	Bekanntes Verhalten: Der Shortcut wird nicht vom Programm verarbeitet	Bevorzuge eine Verknüpfung auf dem Desktop, die mit --finalize verknüpft ist – siehe Kapitel 5
Alt+H kopiert nichts in die Zwischenablage	Normales Verhalten – Alt+H pausiert, es wird nicht abgeschlossen	Verwenden Sie Alt+S oder die Abbruchtaste zum Kopieren – siehe Kapitel 1
Ein Wort wird immer wieder falsch erkannt	Phonetische Verwechslungen der Engine (Wörter, die beim Hören ähnlich klingen)	Fügen Sie eine Korrektur in recognition_aliases ein – siehe Kapitel 4
Das Tool beendet die Sitzung selbst: „Der Erkennungsdienst wird nicht wiederhergestellt.“	Der Decoder hat mehrmals hintereinander unlesbare Datenpakete zurückgegeben, und die Motorrekonstruktion hat daran nichts geändert	Starten Sie das Programm erneut. Wenn dies wiederholt auftritt, fügen Sie dem Bericht die Datei hinzu, die unter ~/ .local/state/ vosk-cli-dictation/ diagnostics/ gespeichert ist (siehe unten)

Die geänderte Konfiguration scheint nicht berücksichtigt zu werden	Die bearbeitete Datei ist nicht die tatsächlich verwendete (es gibt mehrere mögliche Speicherorte)	Überprüfen Sie die Lösungsreihenfolge – siehe Kapitel 4
--	--	---

Wo finde ich eine vollständige Fehlermeldung?

vosk-cli-dictation zeigt seine Fehler direkt im Terminal an, in dem es gestartet wurde. Um ein nicht oben aufgeführtes Problem zu diagnostizieren, sollte man es zuerst in einem sichtbaren Terminal (nicht im Hintergrund) neu starten. Wenn man sich bezüglich der Umgebung unsicher ist, `vosk-cli-dictation --diagnose` überprüft alles (Tools, Modelle, Mikrofon, Zwischenablage, Control-Socket) und zeigt den Status jedes einzelnen Punktes an.

Nur in einem einzigen Fall wird eine Datei geschrieben: wenn der Decoder eine unlesbare Ausgabe liefert, wird die rohe Zeichenkette einmal pro Sitzung unter `~/.local/state/vosk-cli-dictation/diagnostics/.` gespeichert. Diese Datei ist nützlich, um sie an einen Fehlerbericht zur Spracherkennung anzuhängen.

Wenn das Problem weiterhin besteht

Das Projekt-Repository hat eine öffentliche Seite zur Verfolgung von Problemen: <https://github.com/RonanDavalan/vosk-cli-dictation/issues>