Documentation Fonctionnelle et Technique

Projet OrlVoice

Nom du développeur : Abdoulaye Chaibou Saidou

5 mai 2025

Table des matières

1 Introduction

OrlVoice est une application PC, composante entière, de commande vocale et textuelle destinée à automatiser des interactions avec l'interface utilisateur (clics, frappes, ouverture d'applications, etc.). Elle fonctionne localement, sans connexion Internet, et exploite des modèles de reconnaissance vocale, de NLP et éventuellement de computer vision.

2 Objectifs du projet

— Permettre à l'utilisateur de contrôler un ordinateur ou un smartphone par la voix.
— Offrir une interface simple pour basculer entre commandes vocales et textuelles.
— Intégrer des fonctionnalités intelligentes comme la reconnaissance d'écrans, de boutons, d'applications.
— Offrir un mode totalement hors-ligne.

3 Vue d'ensemble de l'application

3.1 Architecture générale

Le projet est structuré en plusieurs modules :

— **Interface utilisateur (UI)** : Interface graphique (Tkinter sur PC / Jetpack Compose sur Android) permettant d'entrer une commande ou d'activer la reconnaissance vocale.
— **Reconnaissance vocale** : Module d'analyse audio permettant de convertir la voix en texte (speech-to-text).
— **Analyse NLP** : Analyse de l'intention de l'utilisateur (ex : ouvrir une application, écrire un texte, cliquer, etc.).
— **Moteur d'action** : Exécution des actions à l'écran (via pyautogui ou API Android).
— **Gestion des chemins** : Système de mémorisation des chemins d'accès aux applications.

4 Fonctionnalités principales

— Détection vocale et exécution automatique.
— Saisie manuelle de commande (en mode texte).
— Ouverture d'applications par leur nom ou leur raccourci.
— Simulation de clics, double-clics, clics droits.
— Mouvements de souris (haut, bas, gauche, droite).
— Écriture de texte dicté.
— Scroll de la page (haut / bas).
— Enregistrement des chemins personnalisés si l'application n'est pas détectée automatiquement.

5 Interface utilisateur

5.1 Fenêtre principale

— Titre : OrlVoice Interface
— Mode vocal : activation/désactivation via une case à cocher.
— Bouton de reconnaissance vocale.
— Champ texte pour saisie manuelle.
— Bouton pour envoyer la commande lorsqu'elle est textuelle.

5.2 Pop-ups et alertes

— Notification lorsque le mode vocal est activé.
— Alerte si le mode vocal est désactivé.
— Avertissement si la commande texte est vide.
— Pop-up de sélection de fichier pour les applications non reconnues.

6 Spécifications fonctionnelles

6.1 Cas d'utilisation principaux

— **UC1** : L'utilisateur clique sur "Parler" → la voix est transcrite → commande exécutée.
— **UC2** : L'utilisateur écrit une commande → clic sur "Envoyer" → commande exécutée.
— **UC3** : Si l'application à ouvrir est inconnue, l'utilisateur sélectionne manuellement le fichier .exe.

6.2 Gestion des erreurs

— Fichier de l'application introuvable : demande manuelle.
— Transcription incorrecte : possibilité de re-saisir la commande.
— Problème avec la reconnaissance : retour visuel par message d'erreur.

7 Spécifications techniques

— **Langage** : Python (PC)
— **Framework UI** : Tkinter (PC)
— **Reconnaissance vocale** : API Speech Reconition de Google
— **Traitement NLP** : classification de texte ou modèle CamemBert
— **Exécution d'actions** : pyautogui (PC)
— **Stockage des chemins d'applications** : chargés dans un fichier JSON dans le dossier .cache

8 Données d'entrée / sortie

8.1 Entrées possibles

— Audio en direct (microphone)
— Texte saisi via l'interface (plus pour pouvoir tester l'application)

8.2 Sorties attendues

— Actions exécutées sur la machine (clic, saisie, lancement, ecrire)
— Logs ou messages de retour dans la console et l'interface

9 Évolutions possibles

— Ajout de la reconnaissance d'éléments visuels avec la vision par ordinateur (YOLO, TFLite)
— Ajout de profils personnalisés pour des utilisateurs spécifiques
— Support multilingue (reconnaissance vocale + NLP multilingues)
— Intégration mobile complète via une app Android

10 Reconnaissance Vocale : défis et solutions

10.1 Limites des modèles avancés comme Whisper

Whisper, développé par OpenAI, est l'un des modèles les plus puissants en transcription vocale, notamment multilingue. Cependant, son usage présente plusieurs défis :

- **Consommation de ressources importante** : Les modèles Whisper, même dans leurs versions petites (small, base), nécessitent un GPU puissant pour une transcription fluide.
- **Poids du modèle** : Le modèle complet dépasse les 1 Go, ce qui le rend difficile à embarquer sur des appareils mobiles ou pour un usage embarqué hors-ligne.
- **Temps de latence** : Même sur GPU, le temps de réponse est parfois trop long pour des interactions utilisateur en temps réel.
- **Pas d'adaptation dynamique** : Whisper ne permet pas l'entraînement en ligne sur la voix spécifique de l'utilisateur.

10.2 Choix de l'API Google Speech Recognizer

L'API `speech_recognition` de Google a été choisie pour son accessibilité, sa simplicité d'intégration avec Python, et sa rapidité dans des cas simples. Elle permet de :

- Transcrire rapidement des fichiers courts.

- Gérer plusieurs accents avec une bonne précision.
- Retourner des résultats en quelques secondes via une API Cloud.

10.3 Limitations de l'API Google

- **Connexion Internet obligatoire** : l'API fonctionne uniquement en ligne, ce qui limite son usage hors réseau.
- **Quota et facturation** : au-delà d'un certain volume de requêtes, l'API devient payante.
- **Confidentialité** : Les données vocales sont envoyées aux serveurs de Google, ce qui pose des questions de protection des données sensibles.

10.4 Solutions prévues

Pour pallier ces limitations, plusieurs approches sont en cours d'étude :

- **Passage à une transcription hors-ligne** grâce à l'utilisation de modèles légers comme `Vosk`, `Coqui STT` ou `Whisper-tiny` optimisé en `ONNX` pour une intégration avec `DJL` en Java.
- **Création d'un fallback** : utiliser Whisper localement si la connexion est absente, et Google Speech en ligne si disponible.
- **Filtrage des données sensibles** avant envoi via l'API, ou suppression immédiate après transcription.
- **Entraînement personnalisé** sur des commandes courtes, avec un modèle CTC plus simple et spécialisé, pour de meilleures performances en usage vocal limité.

10.5 Perspectives

À terme, l'objectif est de proposer une transcription vocale embarquée, multilingue, fiable, et capable de détecter des commandes spécifiques à l'utilisateur, tout en assurant la confidentialité et un fonctionnement entièrement hors-ligne.

11 Annexes

— Manuel d'utilisation (à rédiger séparément)
— Fichiers de configuration (.json, .ini, .db)
— Dictionnaire de commandes acceptées (à intégrer dans l'app)

Fin du document