UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR
Serigne Cheikh Ahmet Tidiane Sy Thioune

AUTOMATISATION DE LA DOCUMENTATION MÉDICALE À L'AIDE DE MODÈLES DE MACHINE LEARNING EMBARQUÉS DANS LE MOBILE

Octobre 2025

Résumé

La numérisation du système de santé a conduit à l'adoption croissante des applications mobiles par les médecins pour faciliter la gestion des dossiers patients. Cependant, la charge de travail associée à l'enregistrement manuel des consultations, combinée à un nombre croissant de patients, complique considérablement la tâche des médecins. Cette surcharge compromet les opérations de saisie des données via des interfaces électroniques. Par conséquent, la qualité des dossiers médicaux s'en trouve négativement affectée. Ce mémoire propose l'automatisation de la documentation médicale à l'aide de modèles de Machine Learning, spécifiquement dédiés à la transcription audio-texte et à la labellisation automatique des informations. Le premier modèle convertit les enregistrements audio des consultations en texte, tandis que le second attribue des libellés appropriés à ce texte pour compléter les formulaires de consultation médicale. L'application mobile développée permettra aux médecins de transcrire et d'étiqueter automatiquement les consultations en temps réel, offrant ainsi une solution pour améliorer la gestion des dossiers patients tout en réduisant la charge administrative. Ce projet vise à optimiser la qualité et l'efficacité de la documentation médicale à travers l'intégration de l'intelligence artificielle dans les outils mobiles destinés à être utilisés par les professionels de santé.

Serigne Cheikh Ahmet Tidiane Sy Thioune

Fadel Toure

Abstract

The digitization of the healthcare system has led to the increasing adoption of mobile applications by physicians to facilitate patient record management. However, the workload associated with manually recording consultations, combined with a growing number of patients, significantly complicates the physicians' task. This overload compromises data entry operations via electronic interfaces. Consequently, the quality of medical records is negatively affected. This thesis proposes the automation of medical documentation using machine learning models specifically designed for audio-to-text transcription and automatic information labeling. The first model converts audio recordings of consultations into text, while the second assigns appropriate labels to this text to complete medical consultation forms. The developed mobile application will allow physicians to automatically transcribe and label consultations in real time, thus providing a solution to improve patient record management while reducing the administrative burden. This project aims to optimize the quality and efficiency of medical documentation through the integration of artificial intelligence into mobile tools intended for use by healthcare professionals.

Avant-propos

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

Mes premiers remerciements vont à Dieu, le Tout-Puissant, qui m'a donné la force et le courage de poursuivre mes études et de mener à bien ce projet.

Je remercie du fond du cœur mes très chers parents qui ont toujours été là pour moi, ainsi que mes frères et sœurs pour leur soutien constant et leurs encouragements.

Je voudrais ensuite exprimer ma plus profonde reconnaissance à mon directeur de recherche, Monsieur Fadel Touré. Au-delà d'être mon professeur et directeur de mémoire, vous avez été pour moi bien plus qu'un simple encadrant académique. Votre confiance indéfectible en mes capacités, même dans les moments de doute, a été le moteur de ma persévérance. Vous avez cru en moi lorsque j'avais du mal à croire en moi-même, et vous avez tout mis en œuvre pour me permettre de réussir. Votre générosité, tant intellectuelle qu'humaine, votre disponibilité sans faille, vos conseils éclairés et votre patience infinie ont façonné non seulement ce travail, mais aussi ma vision de la recherche et de l'excellence. Votre passion communicative pour les mathématiques et l'informatique appliquées m'a poussé à repousser mes limites. Je vous considère comme un mentor, un guide, et plus encore, comme un membre de ma famille. Pour tout cela, recevez ma gratitude éternelle et mon respect le plus profond.

Je tiens également à remercier l'ensemble des professeurs du département de Mathématiques et Informatique, qui m'ont fourni les outils nécessaires à la réussite de mes études et de ce mémoire.

J'exprime ma sincère gratitude envers Madame Christiane Baril, agente d'administration aux affaires départementales, pour sa disponibilité et ses précieux conseils tout au long de mon parcours.

Je tiens spécialement à remercier mon frère Badara Thioune, qui est bien plus qu'un frère : un ami, un confident et un pilier. Tu m'as toujours soutenu et tu as tout fait pour moi par tes conseils avisés et tes encouragements constants.

Enfin, mes vifs remerciements et ma profonde reconnaissance vont à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce travail.

Je dédie ce modeste travail à la mémoire de ma très chère mère, que son âme repose en paix. À mes chers parents et frères pour leur soutien indéfectible, moral et financier, tout au long de mon parcours académique. À toute ma famille pour son amour et ses encouragements constants. À mes adorables amis qui ont toujours été présents à mes côtés, ainsi qu'à tous mes camarades informaticiens, compagnons de ce beau voyage d'apprentissage.

Table des matières

Liste des tableaux

Table des figures

Chapitre 1

Introduction

La digitalisation du système de santé a conduit à l'adoption croissante des applications mobiles par les professionnels de santé pour faciliter la gestion des dossiers patients. Cependant, la charge de travail associée à la transcription des consultations enregistrées, aux efforts d'extraction des informations tabulaires, et de classification (labellisation) de leur contenu, combinée à un nombre de patients croissant, complique la tâche des médecins, compromettant ainsi la qualité des dossiers. Ce mémoire propose l'automatisation de la documentation médicale à l'aide d'un pipeline de modèles embarqués sur mobile de Machine Learning, spécifiquement dédiés à la transcription audio-texte et à la labellisation automatique des informations. Le premier modèle convertit les enregistrements audio des consultations en texte, tandis que le second attribue des libellés appropriés à ce texte pour compléter les formulaires médicaux. L'application mobile développée permet aux médecins de transcrire et d'étiqueter automatiquement les consultations en temps réel, offrant ainsi une solution pour améliorer la gestion des dossiers patients tout en réduisant la charge administrative. Ce projet vise à optimiser la qualité et l'efficacité de la documentation médicale à travers l'intégration de l'intelligence artificielle dans les outils mobiles destinés à être utilisés par les professionnels de la santé.

1.1 Contexte

La digitalisation du système de santé représente une avancée significative dans la modernisation des soins et la gestion des données médicales. Cette transformation numérique vise à optimiser les processus administratifs et cliniques, à améliorer la qualité des dossiers patients et à faciliter l'accès à l'information pour les professionnels de la santé. Cependant, dans ce contexte d'évolution technologique, la documentation médicale demeure un défi majeur. Les médecins doivent gérer un flux croissant de consultations tout en consignant avec précision les informations pertinentes pour assurer un suivi efficace des patients.

1.2 Problématique

La charge de travail des médecins augmente sans cesse, compliquant leur capacité à remplir exhaustivement les dossiers médicaux. Ce manque de précision dans la documentation peut entraîner des lacunes importantes, compromettant la qualité des données nécessaires à une prise de décision éclairée et au suivi des patients. Bien que des solutions numériques existent pour assister les médecins, elles restent souvent limitées par leur nature manuelle et chronophage. Comment peut-on alléger cette charge administrative tout en garantissant une documentation précise et structurée ?

Afin de mesurer l'ampleur du défi posé par la documentation médicale manuelle, nous avons analysé les exigences de la fiche standard québécoise pour la tenue des dossiers médicaux[1]. Le tableau 1.1 présente une analyse détaillée des champs requis selon les normes du Collège des Médecins du Québec(CMQ).

TABLE 1.1 – Analyse des champs requis selon la fiche standard québécoise[1]

Champ	**Type**	**Caractères estimés**
Date	Date	10
Heure (si urgence)	Heure	5
Raison de consultation	Texte court	50-100
Symptomatologie	Texte long	200-400
Durée des symptômes	Texte court	20-50
Éléments positifs/négatifs	Texte long	150-300
Épisodes antérieurs	Texte moyen	100-200
Interventions tentées	Texte moyen	100-150
Liste des médicaments	Liste	100-300
Allergies	Liste	50-150
Température	Numérique	4
Tension artérielle	Numérique	7
Pouls	Numérique	2-3
Fréquence respiratoire	Numérique	2
Poids	Numérique	5
Taille	Numérique	5
IMC	Numérique (calculé)	4
Tour de taille	Numérique	3
Examen physique détaillé	Texte long	300-600
Diagnostic	Texte moyen	50-150
Examens complémentaires	Liste	100-200
Consultations demandées	Liste	50-150
Médicaments prescrits	Liste détaillée	200-400

Suite à la page suivante

Table 1.1 – suite de la page précédente

Champ	**Type**	**Caractères estimés**
Traitements non pharmacologiques	Texte moyen	100-200
Vaccinations/injections	Tableau structuré	150-300
Communications patient	Texte moyen	100-200
Délai prochaine visite	Texte court	20-50
Facteurs de consultation urgente	Texte moyen	50-100
Signature	Signature	10

L'analyse de ce tableau révèle un total de 28 champs distincts à remplir, représentant entre 2 000 et 4 500 caractères selon la complexité de la consultation, avec une moyenne d'environ 3 250 caractères. Des études récentes ont démontré que les médecins consacrent une part considérable de leur temps à la documentation électronique. Sinsky et al.[3], dans une étude observationnelle portant sur quatre spécialités médicales, ont révélé que les médecins passent en moyenne 49,2% de leur temps de travail sur les dossiers médicaux électroniques et autres tâches administratives, contre seulement 27% en interaction directe avec les patients.

En considérant une vitesse de frappe moyenne de 40 mots/minute (environ 200 caractères/minute) pour un utilisateur standard, et en tenant compte du temps nécessaire pour la réflexion et la navigation entre les champs du formulaire électronique, on peut estimer un débit effectif de 100 à 150 caractères/minute. Pour un dossier contenant en moyenne 3 250 caractères, le temps total de saisie serait de 30 à 45 minutes par consultation complète, incluant la saisie brute (22 à 33 minutes), la recherche d'informations dans le système (3-5 minutes), la navigation entre différents écrans (2-3

minutes) et la vérification (2-4 minutes).

Cette charge administrative représente un fardeau significatif, particulièrement pour les médecins effectuant 20 à 30 consultations par jour, soit potentiellement 10 à 22,5 heures de documentation par jour, un volume manifestement insoutenable qui explique pourquoi la qualité des dossiers est souvent compromise[3]. Notre solution vise à réduire ce temps en automatisant la transcription et l'extraction des constantes vitales et, dans de futures investigations, l'extraction d'autres champs textuels. Cela permet aux médecins de se concentrer sur les aspects à forte valeur ajoutée de la consultation, qui nécessitent réellement leur expertise clinique.

1.3 Objectifs de l'étude

L'objectif principal de cette étude est de réduire la charge administrative des professionnels de santé tout en améliorant la qualité et la précision tout en gardant la structuration des données consignées dans les dossiers médicaux. Pour atteindre cet objectif, cette recherche se concentre sur la transcription des enregistrements audio des consultations médicales en texte exploitable, garantissant la fidélité des informations. Elle s'intéresse également à la labellisation automatique, permettant d'extraire et de structurer les informations pertinentes sous forme de mentions précises pour compléter automatiquement les fiches de consultation et les formulaires médicaux. Enfin, ces modèles sont intégrés dans une application mobile dédiée, offrant aux médecins une solution pratique pour enregistrer leurs consultations, générer des transcriptions instantanées et bénéficier d'une organisation automatisée des données.

1.4 Organisation du mémoire

Ce mémoire est structuré de manière à exposer de façon cohérente les différentes étapes de notre démarche de recherche et de développement. Le chapitre 2 est consacré à la revue de la littérature, dans laquelle sont présentés les travaux existants en matière de numérisation du secteur de la santé, les approches en apprentissage automatique appliquées à la documentation médicale ainsi que les solutions mobiles existantes. Le chapitre 3 décrit la méthodologie adoptée, incluant la conception des modèles de transcription et d'extraction, la construction du corpus d'entraînement, le fine-tuning du modèle d'extraction, ainsi que l'architecture logique de notre application. Le chapitre 4 présente la validation expérimentale des modèles développés, en détaillant les protocoles de test, les métriques de performance et l'analyse des résultats obtenus en environnement de développement. Le chapitre 5 porte sur le développement de l'application mobile, en détaillant les choix techniques, l'architecture logicielle, le déploiement des modèles et l'implémentation de l'interface utilisateur. Le chapitre 6 évalue les performances de l'application sur dispositif mobile en conditions réelles, analyse ses limites et propose des perspectives d'amélioration. Enfin, le chapitre 7 conclut le mémoire en synthétisant les apports de ce travail et en proposant des perspectives d'évolution futures.

Chapitre 2

État de l'art

2.1 Description de l'existant

De nombreuses applications proposent des fonctionnalités de transcription audio en texte. Parmi celles-ci, on peut citer :

CMU Sphinx : Un système de reconnaissance vocale open-source développé par l'Université Carnegie Mellon, conçu pour fonctionner entièrement hors ligne sur des appareils mobiles et embarqués. Il offre une transcription audio en temps réel sans nécessiter de connexion Internet, ce qui le rend particulièrement adapté aux environnements à connectivité limitée. CMU Sphinx a été utilisé pour développer des applications de transcription de consultations médicales, permettant aux professionnels de santé de convertir automatiquement leurs enregistrements audio en texte structuré et d'extraire des informations cliniques pertinentes comme les constantes vitales[4].

Speechnotes : une application en ligne permettant de convertir automatiquement des fichiers audio en texte. Elle offre une interface intuitive qui permet aux utilisateurs

d'enregistrer leur voix et de recevoir une transcription en temps réel. L'application prend en charge plusieurs langues et propose des fonctionnalités supplémentaires telles que l'édition des transcriptions et leur exportation dans différents formats.

Google Cloud Speech API : un service de reconnaissance vocale développé par Google, reconnu pour sa grande précision dans la conversion audio-texte. Ce service propose des fonctionnalités avancées, notamment la détection automatique des langues, la reconnaissance des noms propres et l'ajout automatique de ponctuation. Il prend également en charge le streaming en temps réel pour des transcriptions instantanées.

Otter.ai : une application largement utilisée pour la transcription audio en texte, reposant sur l'intelligence artificielle. Elle offre une transcription automatique des fichiers audio avec une précision élevée. Otter.ai inclut également des fonctionnalités pratiques telles que la recherche dans les transcriptions et la synchronisation des enregistrements audio avec le texte, ce qui simplifie l'organisation et la navigation dans les contenus transcrits.

2.2 Critique de l'existant

Bien que les solutions mentionnées précédemment offrent des fonctionnalités importantes pour la transcription audio en texte, elles présentent certaines limites notables :

Fiabilité des transcriptions : Malgré les progrès réalisés dans les technologies de reconnaissance vocale, la précision des transcriptions reste influencée par divers facteurs tels que la qualité de l'enregistrement audio, la clarté de la diction et la complexité du langage employé.

Coût élevé : Plusieurs solutions de transcription peuvent être coûteuses, particulièrement pour un usage intensif ou lorsque des fonctionnalités avancées sont nécessaires. Les tarifs sont souvent déterminés en fonction du volume de transcription, des services supplémentaires et des options de stockage.

Restrictions linguistiques : Certaines applications présentent des limites quant aux langues disponibles, ce qui peut représenter un obstacle pour les utilisateurs travaillant avec des langues peu courantes.

À cela s'ajoute le problème de portabilité mobile. La plupart héberge le modèle de transcription. Dans le contexte médical, envoyer des données a tiers pose des problème confidentialité.

2.3 Utilisation des modèles de machine learning dans la santé

Le machine learning (ML) transforme de nombreux aspects des systèmes de santé en offrant des solutions innovantes pour réduire la charge de travail administrative et améliorer la qualité des soins. Plusieurs travaux illustrent ces avancées.

En 2006, Huggins-Daines et al.[5] ont conçu PocketSphinx, un système de reconnaissance vocale continue adapté aux appareils mobiles et embarqués. Ce système open-source, dérivé de Sphinx-II, est optimisé pour fonctionner en temps réel sur des plateformes à ressources limitées, comme le Sharp Zaurus SL5500. Ils ont réduit les besoins en calcul grâce à des optimisations algorithmiques, telles que l'arithmétique à point fixe et l'utilisation d'arbres kd pour la sélection des gaussiennes. Le système atteint un taux d'erreur de mots (WER) de 13,95 % sur un corpus spécifique, avec une vitesse de traitement 8 fois plus rapide que les systèmes précédents. Cette étude

démontre la faisabilité de la reconnaissance vocale en temps réel pour des appareils à faible puissance de calcul. De la même manière, en 2014, [6] Frid et al. ont exploré l'utilisation des signaux vocaux pour le diagnostic précoce de la maladie de Parkinson. Leur système automatique extrait des caractéristiques acoustiques telles que la fréquence fondamentale, l'énergie à court terme et les coefficients cepstraux pour alimenter un classificateur SVM (Support Vector Machine). Les résultats montrent une précision moyenne de 81,8 % pour distinguer les différents stades de la maladie, y compris les patients sains.

Lakdawala et al.[4] ont développé en 2018 une application mobile exclusivement conçue pour les organisations de santé. Utilisant PocketSphinx, l'application transcrit les consultations médicales audio en texte tout en fonctionnant hors ligne, ce qui garantit une utilisation indépendante d'une connexion Internet. L'architecture repose sur trois composants principaux : un modèle acoustique, un dictionnaire phonétique et un modèle de langage. Les performances incluent une précision de transcription de 70 % pour l'anglais et de 75 % pour le hindi. L'application permet également d'extraire des informations médicales clés, telles que les valeurs d'hémoglobine et la pression artérielle. Tandis que ces approches se concentrent sur la transcription hors ligne, en 2019, Kusumah et al. ont proposé un modèle hybride combinant reconnaissance vocale hors ligne et en ligne pour les smartphones. Le décodeur hors ligne (CMU Sphinx) offre une réponse rapide sans connexion Internet, tandis que le décodeur en ligne (Google Cloud Speech API) fournit une meilleure précision. Cette approche permet de combiner les avantages des deux systèmes : la rapidité du traitement local et la précision accrue des services nuagiques. Bien que les performances quantitatives ne soient pas présentées, cette architecture hybride constitue une solution prometteuse pour équilibrer vitesse et précision dans des contextes mobiles, notamment pour les environnements médicaux.

Enfin, en 2023, D. B. Costa et al. [7] ont proposé une méthodologie innovante pour transformer les appels d'urgence non structurés en données exploitables. Leur ap-

proche repose sur Wav2Vec 2.0 pour la transcription et sur un modèle de compréhension du langage naturel (NLU) pour classifier les transcriptions. Les performances incluent un taux d'erreur de mots (WER) de 42,12 % et une précision de classification de 73,9%. Pour pallier le manque de données annotées, ils ont généré un corpus artificiel, améliorant les résultats globaux. Cette solution met en avant le potentiel de l'IA pour optimiser les systèmes d'urgence, en particulier dans des environnements à faibles ressources.

- **Défis spécifiques du machine learning en médecine**

L'application du machine learning dans le domaine médical présente plusieurs défis significatifs notamment la gestion des données sensibles, les limitations des modèles pour les langues sous-représentées, et la précision dans des environnements variés. M. K. Abd Ghani et I. N. Dewi [8] ont montré que, bien que la reconnaissance vocale réduise le temps de saisie, elle reste sujette à des erreurs dues aux accents et aux limitations des modèles linguistiques. Par ailleurs, N. Pitaksirianant et al. [9] ont développé un système pour tester l'intelligibilité vocale chez les patients post-chirurgie orale, mais ont relevé des limites dans la corrélation avec les évaluations humaines.

2.4 Traitement du langage naturel (NLP) appliqué aux données médicales

Le traitement du langage naturel (NLP) joue un rôle essentiel dans l'analyse des données médicales textuelles, en facilitant l'extraction d'informations pertinentes à partir de grandes quantités de textes, telles que des transcriptions de consultations ou des dossiers patients.

En 2018, P. Withanage et al. [10] ont démontré l'intégration du NLP dans un système de navigation routière intitulé "Direct Me", combinant reconnaissance vocale automatique (ASR) et NLP pour analyser les commandes vocales et fournir des itinéraires personnalisés. Bien que ce travail ne soit pas directement appliqué à la médecine, il illustre la capacité du NLP à extraire des informations structurées à partir de textes complexes, une compétence transférable aux systèmes médicaux pour interpréter des transcriptions de consultations.

Dans un contexte spécifiquement médical, T.W. Sung et al. [11] ont exploré en 2019 une méthode de traduction audio-texte end-to-end basée sur un modèle à double décodeur. Contrairement aux approches traditionnelles qui passent par une transcription intermédiaire avant la traduction, leur modèle produit directement le texte traduit à partir de l'audio source. Cette approche s'est montrée particulièrement efficace pour les langues peu dotées, un défi fréquent dans les contextes médicaux multilingues.

Enfin, A. Radford et al. [2] ont présenté Whisper, un modèle multitâche basé sur une architecture Transformer, entraîné sur 680 000 heures d'audio multilingue. Whisper excelle non seulement dans la transcription audio-texte, mais aussi dans la traduction et la détection d'activité vocale. Sa robustesse et sa capacité à généraliser sans ajustements spécifiques aux données en font un outil précieux pour l'analyse des données médicales multilingues.

Ces travaux montrent que le NLP, associé à des modèles avancés, peut transformer la gestion des données médicales. Qu'il s'agisse de transcription, de traduction ou d'analyse, le NLP contribue à rendre les systèmes de santé plus intelligents et efficaces.

2.5 Annotateurs humains et limitations de la labellisation manuelle

La labellisation manuelle est une étape cruciale dans le développement de modèles de machine learning (ML), en particulier dans le domaine médical. Cependant, elle est souvent coûteuse, chronophage et sujette à des erreurs humaines. Ces limitations ont conduit à l'émergence de méthodes semi-automatisées et automatisées qui intègrent des modèles avancés tout en conservant l'expertise humaine pour des ajustements finaux.

A. Goel et al. [12] ont proposé une approche hybride qui combine des annotations initiales générées par des grands modèles de langage (LLM) avec une validation humaine. Cette stratégie a permis de réduire le temps d'annotation de 58 %, tout en garantissant un score F1 élevé (0,91). L'idée principale repose sur la rapidité et la précision initiale des LLM pour effectuer les tâches répétitives, tandis que les experts humains corrigent les erreurs et apportent des ajustements contextuels, améliorant ainsi la fiabilité globale des annotations.

De manière complémentaire, C. Lavigne et al. [13] ont exploré le fine-tuning de Whisper sur des corpus spécialisés pour améliorer la précision des annotations. Leur travail montre une réduction significative du taux d'erreur de mots (WER), notamment dans des domaines complexes, avec des applications potentielles pour les contextes médicaux. Ce fine-tuning repose sur l'utilisation de données annotées par des experts pour entraîner le modèle à gérer des situations précises et nuancées.

En 2022, A. Elakkiya et al. [14] ont proposé une méthode innovante combinant des modèles de Markov cachés (HMM) et des réseaux neuronaux profonds (DNN) pour améliorer la précision des transcriptions et des annotations. Leur approche utilise les HMM pour modéliser les séquences temporelles et les DNN pour capturer

des représentations complexes des données audio. Cette combinaison a permis d'atteindre une précision de 95,2 %, réduisant considérablement les erreurs par rapport aux méthodes traditionnelles. Cependant, ils ont souligné l'importance d'annotations humaines de haute qualité pour entraîner efficacement les modèles, montrant ainsi que l'expertise humaine reste indispensable pour garantir des performances optimales.

Les chercheurs S. C. Hidayati et al. [15] ont présenté une approche intégrée qui combine un modèle d'apprentissage par ensemble (Stacking Ensemble Learning) avec une validation humaine. Leur système analyse les transcriptions audio-texte pour détecter des émotions et d'autres aspects complexes, en laissant aux annotateurs humains le soin de vérifier et d'affiner les résultats. Cette approche semi-automatisée réduit les erreurs et garantit une meilleure cohérence des annotations dans des contextes cliniques critiques.

2.6 Conclusion

L'état de l'art présente des avancées significatives dans l'automatisation de la documentation médicale, notamment grâce à l'utilisation de modèles de traitement du langage naturel (NLP) et de systèmes hybrides combinant reconnaissance vocale et analyse textuelle. Les recherches mettent en évidence des gains en termes d'efficacité, de rapidité et de précision, tout en soulignant l'importance des approches adaptées aux spécificités médicales, comme les langues peu dotées et les environnements cliniques multilingues.

Cependant, plusieurs limitations persistent. La plupart des travaux actuels se concentrent sur des contextes bien définis, sans toujours prendre en compte la variabilité et la complexité des consultations médicales réelles. Par ailleurs, l'intégration des solutions dans les flux de travail cliniques reste un défi, notamment en raison des

exigences en matière de confidentialité et de sécurité des données.

Notre contribution vise à combler ces lacunes en développant une approche combinée basée sur deux modèles de machine learning : le premier pour convertir les enregistrements audio des consultations en texte et le second pour effectuer une annotation automatique et contextuelle des données extraites. Contrairement aux travaux précédents, notre solution sera optimisée pour gérer des scénarios variés et multilingues, tout en assurant une intégration fluide dans les systèmes cliniques existants.

Pour atteindre ces objectifs, notre méthodologie reposera sur le développement et la validation de modèles de transcription et d'annotation adaptés aux spécificités des données médicales. Elle inclura également des tests utilisateurs pour garantir l'adéquation aux besoins réels des médecins.

Chapitre 3

Méthodologie

3.1 Introduction

Dans ce chapitre, nous présentons la méthodologie adoptée pour le développement de notre solution de transcription audio en texte et d'extraction d'informations. Nous décrivons d'abord les modèles de reconnaissance vocale existants, puis nous expliquons le choix du modèle Whisper d'OpenAI pour notre application. Ensuite, nous détaillons l'intégration de ce modèle dans notre système et la manière dont nous avons conçu une approche d'extraction d'informations basée sur un modèle de type Question-Réponse (QR). Enfin, nous présentons l'architecture logique de notre application, mettant en évidence l'interaction entre les différents composants du système. L'objectif de cette méthodologie est d'assurer une transcription fiable et une extraction d'informations pertinente, tout en garantissant une facilité d'utilisation pour les utilisateurs finaux.

3.2 Développement des modèles ASR transcription audio en texte

Les progrès récents dans le domaine de la reconnaissance automatique de la parole (ASR - Automatic Speech Recognition) ont considérablement amélioré la qualité de la transcription audio en texte. Ces avancées reposent principalement sur des approches d'apprentissage profond, qui permettent d'optimiser la précision et la rapidité des transcriptions.

Ces dernières années, plusieurs modèles ASR ont émergé, chacun avec ses propres caractéristiques et spécificités. Le choix du modèle dépend notamment des besoins en termes de langues supportées, de formats audio pris en charge, de la disponibilité en open source et des performances globales.

Le tableau 3.3 ci-dessous compare trois des principaux modèles de reconnaissance vocale utilisés actuellement : DeepSpeech, Wav2Vec 2.0 et OpenAI Whisper.

Modèle	**Description**	**Caractéristiques**			
		Open source	**Langues**	**Format audio**	**Tailles**
DeepSpeech	C'est un modèle de reconnaissance vocale open-source développé par Mozilla qui utilise des réseaux de neurones profonds pour transcrire la parole en texte. Il est entraîné sur de grandes quantités de données de parole pour obtenir une transcription précise.	Oui	60	WAV, FLAC, Opus	47M
Wav2Vec 2.0	C'est un modèle de reconnaissance vocale développé par Facebook AI qui utilise un réseau de neurones à convolution pour extraire des représentations de bas niveau à partir de l'audio. Il est ensuite entraîné à prédire les représentations de haut niveau à partir de ces représentations de bas niveau, ce qui permet une transcription précise de la parole en texte.	Oui	50	WAV	Base : 95M Large : 317M
OpenAI Whisper	Le modèle Whisper développé par OpenAI, également connu sous le nom d'OpenAI Whisper, est un modèle de reconnaissance vocale à usage général. Il a été entraîné sur un vaste ensemble de données audio variées et possède des capacités multilingues étendues, y compris la reconnaissance vocale multilingue, la traduction vocale et l'identification linguistique[16].	Oui	96	Tous les formats audio et vidéos	Tiny : 39M Base : 74M Small : 244M Medium : 769M Large : 1550M

TABLE 3.1 – Comparaison des modèles existants de reconnaissance vocale

Comme l'illustre le tableau 3.3 ci-dessus, DeepSpeech et Wav2Vec 2.0 sont des solutions efficaces et open source adaptée à des besoins variés en transcription vocale. Cependant, le modèle OpenAI Whisper se distingue par sa polyvalence accrue, sa prise en charge d'un plus grand nombre de langues et sa compatibilité avec une large gamme de formats audio et vidéo. Ces caractéristiques en font un choix privilégié pour des applications multilingues nécessitant une flexibilité optimale.

3.2.1 Solution adoptée

Dans le cadre de ce projet, nous avons choisi d'intégrer Whisper, un modèle de reconnaissance vocale avancé développé par OpenAI, au sein de notre application de transcription audio en texte et d'extraction d'informations. Grâce à ses capacités d'apprentissage automatique de pointe, Whisper offre une transcription de haute qualité, garantissant une grande précision, même dans des conditions acoustiques variées. Notre solution a été pensée pour être polyvalente et accessible. Elle prend en charge une large gamme de formats audio et vidéo, et permet la transcription automatique dans plus de 96 langues. De plus, Whisper inclut une fonctionnalité de traduction automatique vers l'anglais, élargissant ainsi son champ d'application à des contextes multilingues.

3.2.2 Notions Fondamentales

3.2.2.1 Machine Learning

Le Machine Learning est une branche de l'intelligence artificielle qui permet aux systèmes informatiques d'apprendre à partir de données sans être explicitement programmés pour chaque tâche. Contrairement aux approches traditionnelles qui reposent

sur un ensemble de règles définies par un programmeur, le Machine Learning utilise des algorithmes capables de détecter des tendances et d'identifier des patterns afin d'effectuer des prédictions et des classifications automatiques. Son efficacité dépend fortement de la qualité et du volume des données utilisées pour l'entraînement des modèles[17].

3.2.2.2 Deep Learning

Le Deep learning ou apprentissage profond est l'une des technologies principales du Machine learning. Avec le Deep Learning, nous parlons d'algorithmes capables de mimer les actions du cerveau humain grâce à des réseaux de neurones artificiels. Les réseaux sont composés de dizaines voire de centaines de couches de neurones, chacune recevant et interprétant les informations de la couche précédente[17].

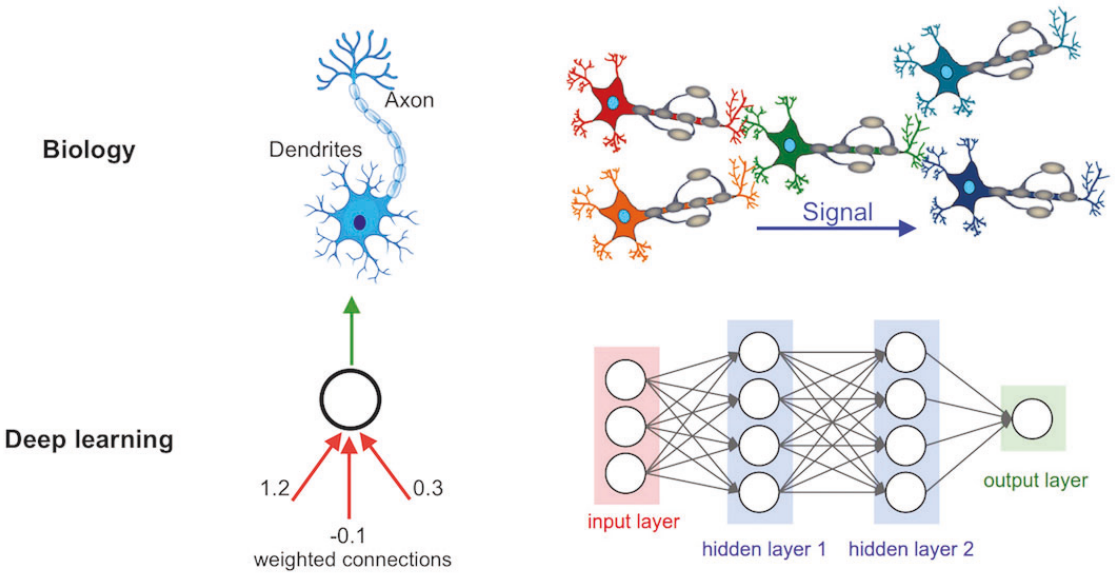


FIGURE 3.1 – Regard comparatif entre le deep learning et la biologie [18]

3.2.2.3 Réseau de neurones

Un réseau de neurones artificiels est un système interconnecte de neurones utilisés dans l'intelligence artificielle pour résoudre des problèmes complexes tels que la reconnaissance vocale et le traitement du langage naturel. Il est capable d'apprendre à partir des données et d'extraire des informations significatives pour accomplir ces tâches[19].

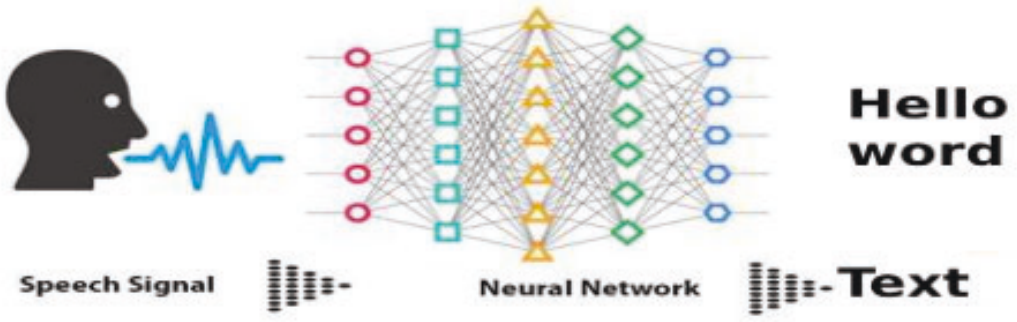


FIGURE 3.2 – Réseaux de neurones

3.2.2.4 Réseau de neurones convolutif(CNN)

Un réseau de neurones convolutif (CNN) est une architecture de réseau de neurones spécifiquement conçue pour analyser des données structurées, telles que des images ou des fichiers audio, en utilisant des opérations de convolution pour extraire automatiquement des caractéristiques significatives et effectuer des tâches telles que la reconnaissance vocale ou la détection d'objets.

3.2.2.5 Apprentissage faiblement supervisé

L'apprentissage faiblement supervisé (Weakly Supervised Learning) est une forme d'apprentissage automatique où l'apprenant dispose de peu d'étiquettes pour travailler, ce qui peut rendre l'apprentissage plus difficile. Il s'agit d'une technique utilisée lorsque seules des étiquettes bruitées ou incomplètes sont disponibles. Elle peut également être employée lorsque les données sont insuffisantes pour un apprentissage supervisé fort, ou pour créer des modèles plus robustes qui ne dépendent pas d'une seule source de données.

3.2.2.6 Modèle séquence à séquence

Un modèle séquence à séquence (Seq2Seq) est une architecture de modèle qui permet de traiter des données séquentielles, telles que des phrases, des séquences de mots, des séquences temporelles, etc. Il prend en entrée une séquence d'éléments et génère en sortie une autre séquence d'éléments. Ces éléments peuvent être des mots, des lettres, des caractéristiques d'une image ou d'un audio[20].

3.2.2.7 Modèle Transformer séquence-à-séquence

Le modèle Transformer séquence-à-séquence est une amélioration du modèle Seq2Seq qui utilise une architecture basée sur les Transformers. L'architecture Transformer introduit l'utilisation de couches d'attention pour capturer les relations entre les mots dans une séquence de manière plus efficace. Contrairement aux modèles Seq2Seq traditionnels, qui utilisent des couches récurrentes pour capturer les dépendances séquentielles, le Transformer exploite les propriétés de l'auto-attention pour effectuer des calculs parallèles sur l'ensemble de la séquence.[21].

3.2.2.8 Jeu de données

Les jeux de données, largement utilisés en machine learning, regroupent des ensembles cohérents de données sous différentes formes telles que du texte, des chiffres, des images ou des vidéos. Ils peuvent être représentés sous forme de tableaux, de graphes, d'arbres ou d'autres structures[22].

- **Quelques catégories de données pour l'évaluation des systèmes ASR (Automatic Speech Recognition)**

LibriSpeech Clean : se réfère à une catégorie spécifique de données de haute qualité dans le corpus LibriSpeech. Ces données sont soigneusement sélectionnées et prétraitées pour éliminer les bruits, les interférences et les problèmes d'enregistrement. Elles sont utilisées pour évaluer les performances des systèmes de reconnaissance automatique de la parole et sont considérées comme étant relativement faciles à traiter par rapport à d'autres catégories du corpus.

Common Voice : est un projet initié par Mozilla qui vise à créer une base de données libre et ouverte pour la reconnaissance automatique de la parole. Des volontaires enregistrent des phrases avec un microphone, tandis que d'autres vérifient ces enregistrements. Les données audio et leurs transcriptions sont ensuite rassemblées dans une base de données publique sous la licence CC0 [1], permettant aux développeurs d'utiliser ces données sans restriction ni frais pour leurs applications de reconnaissance vocale[23].

Artie : est un ensemble de données en anglais comprenant des paires audio-transcription validées par des experts, accompagnées de balises démographiques telles que l'âge, le

1. CC0 (Creative Commons Zero) est une licence de domaine public permettant l'utilisation libre et sans restriction des données vocales pour le développement de systèmes de reconnaissance automatique de la parole.

genre et l'accent. Il a été créé dans le but de détecter les biais démographiques dans les systèmes de reconnaissance automatique de la parole[24].

Afin d'évaluer les performances des modèles ASR, nous comparons leurs taux d'erreur de mots (WER) sur différents ensembles de données. Le WER est un indicateur clé qui mesure la précision d'un système de reconnaissance vocale en quantifiant les erreurs de transcription. La figure 3.3 présente les résultats obtenus par plusieurs modèles, y compris Whisper, Wav2Vec 2.0 et DeepSpeech.

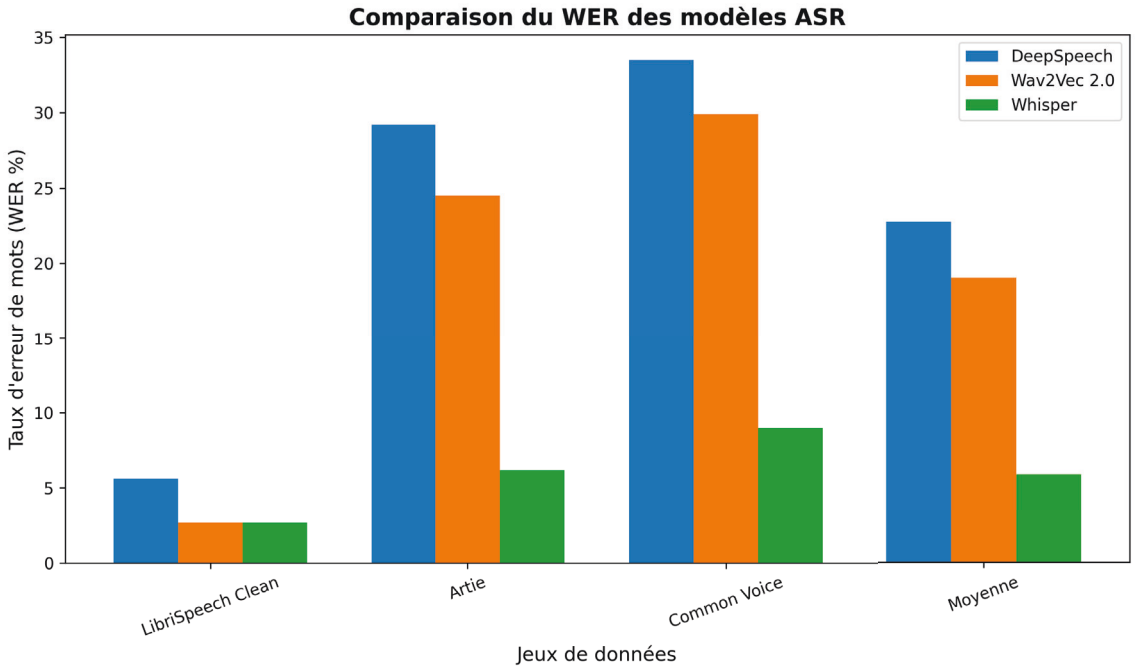


FIGURE 3.3 – Comparaison des performances des modèles ASR en fonction du taux d'erreur de mots (WER).

3.2.3 Ouverture sur la solution adoptée

Les avancées dans le domaine de la reconnaissance vocale ont été stimulées par le développement de techniques d'apprentissage supervisé et non supervisé. L'apprentissage non supervisé utilise des encodeurs audio pré-entraînés pour apprendre des

représentations de haute qualité de la parole, mais nécessite un décodeur performant pour convertir ces représentations en sorties utilisables, ce qui demande une étape de fine-tuning.

D'un autre côté, les systèmes de reconnaissance de la parole pré-entraînés de manière supervisée sur plusieurs ensembles de données présentent une meilleure robustesse et une meilleure généralisation aux données inconnues que les modèles entraînés sur une seule source. Cependant, il y a une quantité limitée de données disponibles pour cette approche.

Pour résoudre ce problème, OpenAI a utilisé une supervision faible (Weak Supervision) pour augmenter le volume de données de 5 000 heures à 680 000 heures en utilisant des méthodes de filtrage automatisées pour améliorer la qualité des transcriptions. Ils ont également développé des heuristiques pour détecter et supprimer les transcriptions générées par machine. Après avoir entraîné un modèle initial, ils ont effectué une inspection manuelle des sources de données en se basant sur le taux d'erreur et la taille des données pour identifier et supprimer efficacement les sources de faible qualité[25].

3.2.4 Architecture du modèle Whisper

Whisper est un modèle encodeur-décodeur basé sur les Transformers, également connu sous le nom de modèle de séquence à séquence. Il associe une séquence de caractéristiques de spectrogramme audio a une séquence de jetons de texte. Tout d'abord, l'audio d'entrée est divisé en segments de 30 secondes et converti en spectrogrammes log-Mel à l'aide de l'extracteur de caractéristiques. Ensuite, l'encodeur Transformer encode le spectrogramme pour former une séquence d'états cachés de l'encodeur. Enfin, le décodeur prédit de manière autonome les jetons de texte en se basant à la fois sur les jetons précédents et les états cachés de l'encodeur.

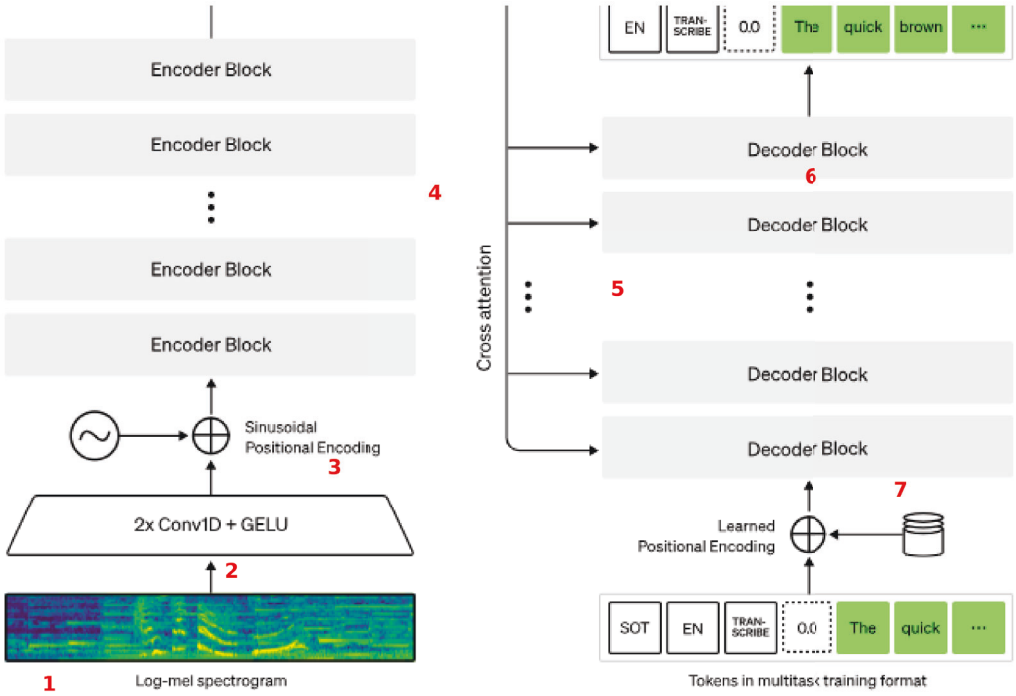


FIGURE 3.4 – Architecture Whisper

[26]

Les explications de chaque nombre qui représente les différentes composantes de l'architecture whisper se trouvent dans la section suivante.

3.2.5 Les composantes

3.2.5.1 Log-mel spectrogramme

Le Log-mel spectrogramme (1) est une représentation visuelle de l'énergie fréquentielle d'un signal audio. Il est obtenu en appliquant une transformée de Fourier à court

terme STFT [2] sur des segments temporels de l'audio, puis en calculant le logarithme de l'énergie spectrale obtenue. Les fréquences sont ensuite converties en une échelle de Mel, qui correspond à l'échelle de perception auditive humaine.

3.2.5.2 2×Conv1D+GELU

Conv1D (2) est une opération de convolution utilisée dans les réseaux de neurones, où le 1D fait référence à une dimension. Contrairement à la convolution 2D utilisée pour le traitement d'images, la convolution 1D est utilisée pour traiter des séquences unidimensionnelles telles que des séries temporelles ou des signaux audio.

La fonction GELU (Gaussian Error Linear Unit) est une fonction d'activation non linéaire qui est utilisée dans les réseaux de neurones. Elle est définie mathématiquement comme suit[27] :

$$\mathrm{GELU}(x) = \frac{1}{2}x\left[1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715\,x^3\right)\right)\right] \tag{3.1}$$

2. STFT (Short-Time Fourier Transform) est une technique de traitement du signal qui permet d'analyser les variations fréquentielles d'un signal audio sur de courtes périodes de temps.

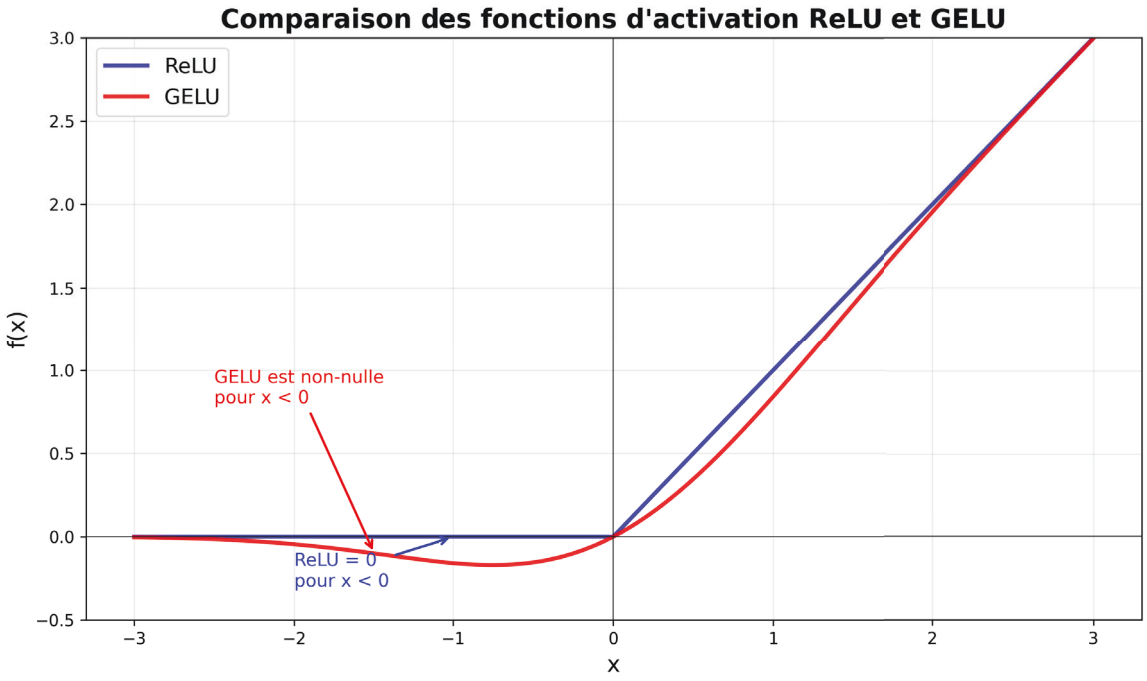


FIGURE 3.5 – Comparaison des fonctions d'activation ReLU et GELU

Comme illustré dans la Figure 3.5, la fonction GELU présente des avantages significatifs par rapport à ReLU. Contrairement à ReLU qui annule complètement les valeurs négatives (ReLU(x) = 0 pour x ¡ 0), GELU permet des valeurs non nulles même pour les entrées négatives, grâce à sa forme de courbe lisse et continue. Cette propriété permet aux gradients de se propager même pour les neurones avec des activations négatives, facilitant ainsi l'apprentissage dans les couches profondes et réduisant le problème des "neurones morts" observé avec ReLU. C'est pour ces raisons que GELU est privilégiée dans les architectures Transformer modernes, incluant Whisper.

3.2.5.3 Sinusoidal Positional Encoding

Sinusoidal Positional Encoding (3) est une technique de codage utilisée dans les modèles de réseaux de neurones pour attribuer des vecteurs de position uniques à chaque élément d'une séquence. Ces vecteurs sont calculés à l'aide de fonctions tri-

gonométriques (sinus et cosinus) et sont **fixes**, c'est-à-dire non entraînables durant le processus d'apprentissage[28].

Contrairement aux encodages positionnels appris (learned positional embeddings) qui sont optimisés durant l'entraînement, les encodages sinusoïdaux utilisent une formule mathématique déterministe. Cette approche permet au modèle de capturer l'ordre séquentiel des données d'entrée et d'incorporer des informations sur la position relative de chaque élément, tout en offrant l'avantage de pouvoir généraliser à des séquences plus longues que celles vues durant l'entraînement.

3.2.5.4 Transformers Encoder Blocks

Les Transformers Encoder Blocks (4) sont des blocs constitutifs de l'architecture du Transformer utilisés dans les tâches de traitement du langage naturel. Ils sont responsables de l'encodage des informations de la séquence d'entrée en utilisant des mécanismes d'attention et de transformation non linéaires. Ces blocs jouent un rôle essentiel dans la modélisation des relations contextuelles entre les différents éléments de la séquence, permettant ainsi au modèle d'apprendre des représentations de haute qualité.

3.2.5.5 Cross-attention

L'attention croisée (cross-attention) (5) est un mécanisme utilisé dans les modèles Transformers pour permettre au décodeur de se focaliser sur des parties spécifiques du contexte encodé lors de la génération de la séquence de sortie.

3.2.5.6 Transformer Decoder Blocks

Les blocs de décodage Transformer (6) sont les composants clés du décodeur d'un modèle Transformer. Ils prennent en entrée une séquence de jetons et appliquent des mécanismes d'attention pour capturer les informations contextuelles pertinentes. Chaque bloc de décodage est composé de plusieurs sous-modules, tels que l'attention multi-têtes et les couches de réseaux de neurones positionnels, qui permettent de prendre en compte à la fois les informations locales et globales lors de la génération de la séquence de sortie. Ces blocs de décodage travaillent de manière récursive, en utilisant les prédictions précédentes pour guider les prédictions suivantes, jusqu'à ce que la séquence de sortie complète soit générée.

3.2.5.7 Learned Positional Encoding

Learned Positional Encoding (7) est une méthode par laquelle un modèle apprend des représentations spécifiques pour chaque position dans une séquence, permettant ainsi de capturer les relations spatiales et temporelles. Cela favorise la compréhension de l'ordre des éléments dans les tâches de traitement de séquence.

3.2.6 Famille de modèles Whisper

Whisper, développé par OpenAI, est proposé en plusieurs variantes qui se distinguent par leur capacité de traitement, leur taille de réseau neuronal et leur performance en transcription multilingue. Chaque modèle Whisper se différencie par le nombre de couches, la largeur des couches, le nombre de têtes d'attention et le nombre total de paramètres, ce qui influence directement leur précision et leur rapidité.

La tableau 3.2 résume ces caractéristiques pour chaque version du modèle.

Modèle	**Couches**	**Largeur**	**Têtes**	**Paramètres**
Tiny	4	384	6	39M
Base	6	512	8	74M
Small	12	768	12	244M
Medium	24	1024	16	769M
Large	32	1280	20	1550M

TABLE 3.2 – Caractéristiques des différents modèles Whisper [2]

3.2.7 Fonctionnement du modèle Whisper

Le modèle Whisper suit un pipeline structuré en plusieurs étapes pour convertir un fichier audio en texte exploitable. Voici les différentes phases de son fonctionnement :

1 Prétraitement de l'audio : l'audio d'entrée est normalisé en étant échantillonné à 16 000 Hz afin d'harmoniser les données. Ensuite, une transformation en spectrogramme log-mel est appliquée, générant une représentation de l'audio avec 80 canaux fréquentiels. Cette conversion s'effectue en fenêtres de 25 millisecondes avec un chevauchement de 10 millisecondes, permettant de capturer à la fois les informations temporelles et fréquentielles du signal sonore.

2 Encodage : l'encodeur prend en entrée le spectrogramme log-mel et en extrait les caractéristiques pertinentes pour la reconnaissance vocale. Il suit plusieurs étapes :

- Une première phase de traitement ("stem") applique deux couches de convolution avec une largeur de filtre de 3, associées à une activation GELU, optimisant ainsi l'apprentissage des caractéristiques phonétiques ;

- Ensuite, des embeddings positionnels sinusoïdaux sont ajoutés pour préserver l'ordre séquentiel des données ;

- Enfin, les données passent à travers une série de blocs Transformer qui analysent les relations entre les éléments de la séquence audio et extraient des informations de haut niveau.

3 Décodage : Le décodeur transforme les représentations encodées en une séquence de mots ou tokens correspondant au texte transcrit. Il s'appuie sur :

- Des embeddings positionnels appris, permettant au modèle de mieux comprendre la structure des phrases ;

- Un mécanisme de relation entrée-sortie, qui aide à contextualiser chaque mot généré en fonction des précédents ;

- Des blocs Transformer, qui analysent les dépendances linguistiques pour produire des prédictions précises et grammaticalement correctes.

4 Tokenization : le texte produit par Whisper est ensuite découpé en tokens à l'aide de BPE[3] (Byte Pair Encoding). Ce procédé garantit une représentation efficace et compacte des unités linguistiques, permettant une meilleure généralisation du modèle à différents langages et accents.

3.2.8 Métrique d'évaluation

Afin d'évaluer la performance de du système de transcription automatique Whisper, nous utilisons des métriques standardisées en reconnaissance vocale. L'une des mesures les plus couramment utilisées est le Taux d'Erreur de Mots (WER).

3. BPE est une méthode de segmentation de texte largement utilisée dans le domaine du traitement automatique du langage naturel. Elle permet de diviser un texte en sous-unités plus petites, appelées tokens, en fonction de leur fréquence d'apparition dans le corpus.

- **Taux d'erreur de mots (WER)**

WER : Le terme WER ("Word Error Rate") est utilisé dans le domaine de la reconnaissance automatique de la parole (ASR) pour mesurer la performance d'un système de reconnaissance vocale. Le WER est un indicateur du taux d'erreurs lors de la conversion de la parole en texte[29].

Le WER est défini par la formule suivante :

$$\text{WER} = \frac{S + D + I}{N} \tag{3.2}$$

où :

S : représente le nombre de substitutions (mots incorrectement reconnus) ;

D : représente le nombre de suppressions (mots manquants dans la transcription) ;

I : représente le nombre d'insertions (mots en trop dans la transcription) ;

N : représente le nombre total de mots dans la transcription de référence.

3.2.9 Taux d'erreur relative (RER)

RER : Le RER (Relative Error Rate) est une mesure utilisée pour évaluer la différence en pourcentage entre les performances de deux modèles. Il est couramment utilisé dans le domaine de la reconnaissance vocale et d'autres tâches de traitement du langage naturel. La formule mathématique du RER est la suivante :

$$\text{RER} = \frac{\text{ER1} - \text{ER2}}{\text{ER2}} \times 100 \tag{3.3}$$

où :

ER1 : représente le taux d'erreur du premier modèle ;

ER2 : représente le taux d'erreur du deuxième modèle.

Un **RER positif** indique que le second modèle performe mieux que le premier (taux d'erreur réduit), tandis qu'un **RER négatif** indique une dégradation des performances (taux d'erreur augmenté).

3.2.10 Évaluation du modèle Whisper

L'évaluation des systèmes de reconnaissance automatique de la parole repose principalement sur le taux d'erreur de mots (Word Error Rate, WER). Cependant, cette métrique présente certaines limitations, notamment sa sensibilité aux différences mineures de transcription, telles que des variations stylistiques ou orthographiques qui n'altèrent pas le sens du texte. Pour pallier cette problématique, OpenAI a mis au point un outil de normalisation du texte, qui vise à standardiser les transcriptions avant le calcul du WER. Cette approche permet de réduire la pénalisation des différences non sémantiques, offrant ainsi une évaluation plus représentative des performances réelles du modèle. Des analyses manuelles ont révélé plusieurs cas où le WER sous-estimait les capacités du modèle Whisper en le pénalisant injustement pour des variations mineures. Grâce à cette normalisation, le WER du modèle Whisper a enregistré une baisse significative, atteignant jusqu'à 50% sur certains ensembles de données.[2].

Jeu de données	**Wav2Vec 2.0 Large (no LM)**	**Whisper Large V2**	**RER (%)**
LibriSpeech Clean	2.7	2.7	0.0
Artie	24.5	6.2	74.7
Common Voice	29.9	9.0	69.9
Average	19.03	5.96	48.2

TABLE 3.3 – Comparaison des performances de Wav2Vec 2.0 et Whisper[2]

Comme le montre le tableau 3.3, le modèle Whisper Large V2 offre une amélioration considérable de 48,2% par rapport au modèle Wav2Vec 2.0 Large (sans modèle de langage). Cette performance supérieure est particulièrement visible sur des ensembles de données complexes tels que Artie et Common Voice, où Whisper surpasse nettement son la baseline en termes de réduction du taux d'erreur.

Après avoir établi les performances du système de transcription avec Whisper, nous présentons maintenant le second composant essentiel de notre pipeline : le modèle d'extraction d'informations médicales.

- **Extraction d'informations médicales**

Le modèle d'extraction repose sur DistilBERT, une version allégée de BERT optimisée pour des tâches de Question-Réponse. Afin d'adapter ce modèle aux spécificités du domaine médical, nous avons procédé à un fine-tuning supervisé sur un corpus de 200 consultations médicales synthétiques généré par intelligence artificielle. Les sections suivantes détaillent la construction de ce corpus (section 3.3), le processus de fine-tuning (section 3.4) et l'architecture technique du modèle (section 3.5). Le modèle utilise cinq questions standardisées pour extraire chaque constante vitale :

— *Quelle est la température corporelle ?*

— *Quelle est sa pression artérielle ?*

— *Quelle est sa fréquence cardiaque ?*

— *Quelle est sa fréquence respiratoire ?*

— *Quel est son niveau de saturation en oxygène ?*

Ce modèle a été converti en TensorFlow Lite (TFLite) afin de garantir une exécution rapide sur mobile, permettant une identification automatique des informations médicales essentielles avec un haut degré de précision.

3.3 Construction du corpus d'évaluation

3.3.1 Génération du corpus synthétique

Face au manque de données médicales authentiques accessibles, nous avons adopté une approche de génération automatique de consultations médicales synthétiques via ChatGPT[30]. Cette méthode permet de créer un corpus cohérent respectant la terminologie médicale standardisée tout en préservant la confidentialité des données patients. Le corpus d'entraînement a été structuré autour de cinq constantes vitales essentielles avec des questions standardisées : "What is body temperature ?", "What is his blood pressure ?", "What is his heart rate ?", "What is his respiratory rate ?" et "What is his oxygen saturation level ?". Pour valider la robustesse de notre approche, nous avons généré 200 consultations médicales synthétiques et analysé leur variabilité selon plusieurs dimensions.

3.3.2 Analyse de la variabilité du corpus

L'analyse statistique des 200 consultations générées révèle une variabilité significative dans les valeurs des constantes vitales, démontrant la diversité du corpus d'entraînement.

Constante vitale	**Valeur min**	**Valeur max**	**Moyenne**	**Variance**	**Écart-type**
Température (°C)	33.5	40.2	37.1	2.3	1.5
Pression artérielle (mmHg)	90	180	125	18.4	4.3
Fréquence cardiaque (bpm)	55	120	78	15.2	3.9
Fréquence respiratoire (cpm)	12	28	18	8.1	2.8
Saturation O2 (%)	88	100	96	4.2	2.1

TABLE 3.4 – Analyse statistique de la variabilité des constantes vitales dans le corpus synthétique (N=200)

3.3.3 Diversité structurelle et contextuelle

L'analyse de l'ordre d'apparition des constantes vitales dans les consultations révèle une distribution aléatoire, confirmant l'absence de biais structurel. Nous avons mesuré la fréquence avec laquelle chaque constante vitale est la première à être mentionnée dans le texte de consultation :

- Température citée en premier : 18% des cas
- Pression artérielle citée en premier : 25% des cas
- Fréquence cardiaque citée en premier : 22% des cas
- Fréquence respiratoire citée en premier : 19% des cas
- Saturation O2 citée en premier : 16% des cas

Cette variabilité dans l'ordre d'apparition garantit que le modèle d'extraction n'apprend pas de patterns positionnels mais se base réellement sur la compréhension sémantique du contenu.

3.4 Fine-tuning du modèle d'extraction d'informations

Le modèle d'extraction repose sur DistilBERT, une version allégée de BERT optimisée pour des tâches de Question-Réponse. Afin d'adapter ce modèle aux spécificités du domaine médical, nous avons procédé à un fine-tuning supervisé sur notre corpus de 200 consultations médicales synthétiques.

3.4.1 Processus de fine-tuning

Le fine-tuning a été réalisé en plusieurs étapes :

— **Préparation des données** : Chaque consultation a été annotée avec les cinq constantes vitales cibles (température, pression artérielle, fréquence cardiaque, fréquence respiratoire, saturation en oxygène).

— **Format Question-Réponse** : Pour chaque constante, nous avons créé des paires (question, contexte, réponse) où le contexte est le texte de la consultation et la réponse correspond à la valeur de la constante vitale.

— **Configuration d'entraînement** :

  — 200 consultations médicales synthétiques

  — 160 consultations pour l'entraînement (80%)

  — 40 consultations pour la validation (20%)

  — Taux d'apprentissage : $3 \cdot 10^{-5}$

  — Nombre d'époques : 5

  — Taille de lot (batch size) : 8

  — Optimiseur : AdamW

3.4.2 Résultats du fine-tuning

La Figure 3.6 présente l'évolution de la fonction de perte durant l'entraînement du modèle sur les 5 époques.

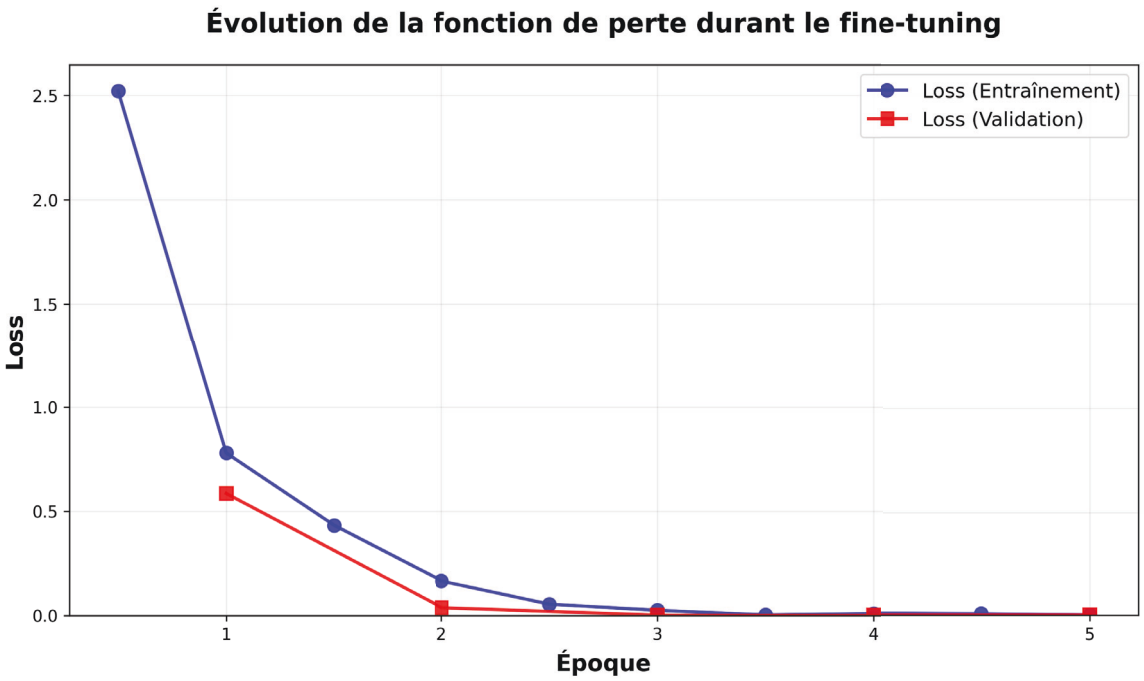


FIGURE 3.6 – Évolution de la fonction de perte durant le fine-tuning du modèle DistilBERT

Comme illustré dans la Figure 3.6, on observe une convergence progressive de la fonction de perte d'entraînement (courbe bleue) et de validation (courbe rouge). La fonction de perte d'entraînement diminue de manière régulière au fil des époques, passant de 2,5 à l'époque initiale à une valeur proche de zéro à l'époque 5, démontrant que le modèle assimile progressivement les patterns d'extraction spécifiques au domaine médical.

La courbe de validation suit une évolution similaire, démarrant à 0,6 et convergeant également vers zéro, sans présenter de divergence significative par rapport à la courbe d'entraînement. Cette convergence harmonieuse confirme l'absence de surapprentissage et indique que le modèle généralise bien aux données non vues durant l'entraînement. La stabilisation des deux courbes après la troisième époque suggère que le modèle a atteint un point d'équilibre optimal entre apprentissage et généralisation.

Le fine-tuning a permis d'améliorer significativement les performances du modèle sur le domaine médical. Le tableau 3.5 présente une comparaison des performances avant et après fine-tuning.

Métrique	**Avant**	**Après**	**Amélioration**
Score F1 moyen	0.68	0.92	+35%
Précision	0.71	0.94	+32%
Rappel	0.65	0.90	+38%

TABLE 3.5 – Comparaison des performances avant et après fine-tuning

Cette adaptation au domaine médical était essentielle car le modèle pré-entraîné générique ne reconnaissait pas efficacement les unités de mesure médicales (mmHg, bpm, cpm, %) ni les formulations spécifiques aux constantes vitales. Le fine-tuning a permis au modèle d'apprendre les patterns linguistiques propres aux consultations médicales et d'améliorer sa capacité à localiser précisément les valeurs numériques associées à chaque constante vitale.

La section suivante détaille l'architecture technique du modèle d'extraction ainsi optimisé et son processus d'entraînement.

3.5 Architecture du modèle d'extraction

3.5.1 Architecture du modèle

Nous utilisons un modèle de Question-Réponse(QR) basé sur l'architecture DistilBERT et entraîné spécifiquement sur notre corpus médical synthétique. L'architecture de l'extraction repose sur les étapes suivantes :

- **Prétraitement du texte** : normalisation et segmentation du texte transcrit ;
- **Analyse par le modèle QR** : identification des entités médicales et extraction des valeurs associées ;
- **Génération des résultats** : structuration des constantes vitales extraites pour leur affichage et leur sauvegarde.

3.5.2 Processus d'entraînement

Le modèle utilise cinq questions standardisées pour extraire chaque constante vitale. Le processus d'extraction des informations médicales suit plusieurs étapes : une fois la transcription générée, le texte obtenu est soumis au modèle **QR**. Celui-ci analyse les données et extrait les valeurs médicales pertinentes, en les associant à leurs unités de mesure respectives. Ce modèle a été converti en TensorFlow Lite (TFLite) afin de garantir une exécution rapide sur mobile, permettant une identification automatique des informations médicales essentielles avec un haut degré de précision.

3.6 L'architecture logique de l'application ConsultationService

L'architecture logique de notre application repose sur un modèle MVC (Modèle-Vue-Contrôleur), intégrant deux modèles spécialisés : le modèle **Whisper** pour la transcription audio en texte et le modèle **QR** (Question-Réponse) pour l'extraction des informations pertinentes. La figure 3.7 illustre cette architecture.

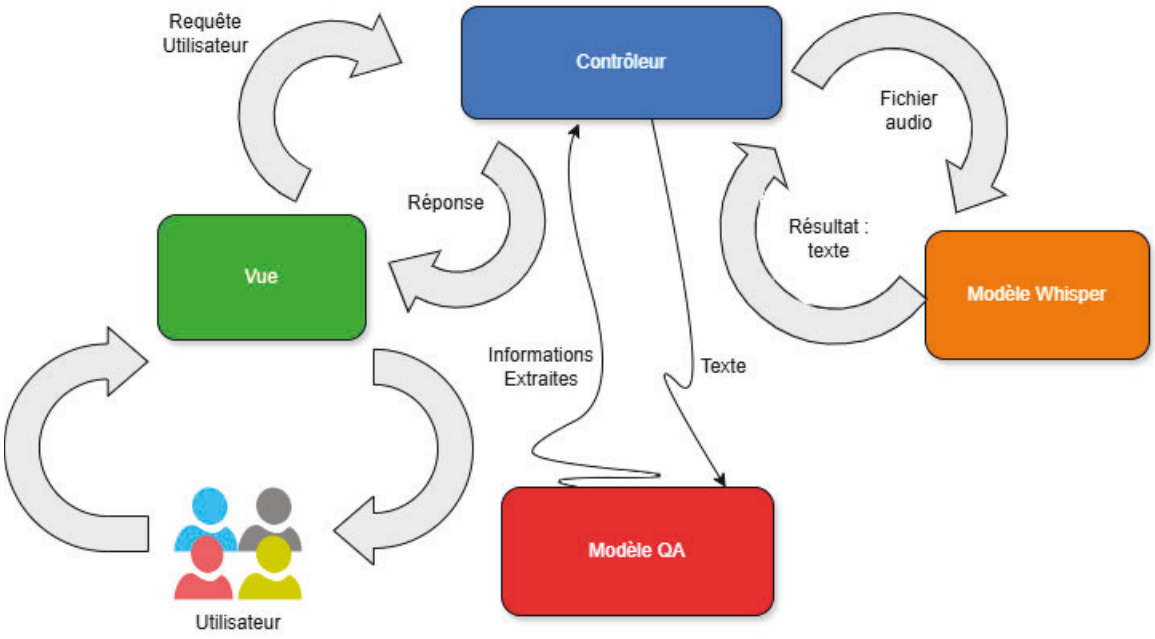


FIGURE 3.7 – Architecture logique de l'application

L'architecture est composée des éléments suivants :

Vue (Interface utilisateur) : l'utilisateur interagit avec l'application via une interface intuitive qui lui permet d'enregistrer un fichier audio et d'afficher les résultats de la transcription et de l'extraction des informations.

Contrôleur : Il gère la communication entre les composants et orchestre le processus de traitement :

- Il reçoit la requête de l'utilisateur et envoie le fichier audio au **modèle Whisper** ;
- Une fois la transcription générée, il transmet le texte au **modèle QR** pour l'extraction des informations ;
- Il retourne les résultats extraits à la Vue pour affichage ;

Modèle Whisper : ce modèle, basé sur l'apprentissage automatique, prend en entrée un fichier audio et génère une transcription textuelle du contenu.

Modèle QR (Extraction d'informations) : une fois la transcription obtenue, ce modèle effectue une analyse du texte et extrait des informations spécifiques, comme les constantes vitales lors d'une consultation médicale.

Flux des données :

- L'utilisateur fait un enregistrement audio via l'interface ;
- Le fichier est envoyé au contrôleur, qui le transmet au modèle Whisper ;
- Whisper renvoie une transcription du fichier audio ;
- Le texte transcrit est ensuite envoyé au modèle QR pour extraire des informations pertinentes ;
- Les informations extraites sont affichées à l'utilisateur via la Vue.

3.7 Conclusion

Ce chapitre a permis de présenter en détail les choix méthodologiques qui ont guidé le développement de notre solution. Nous avons justifié l'utilisation du modèle Whisper pour la transcription automatique et décrit l'architecture du modèle d'extraction basé sur DistilBERT.

La construction d'un corpus synthétique de 200 consultations médicales, combinée à un processus de fine-tuning, a permis d'adapter le modèle d'extraction aux spécificités du domaine médical, atteignant une précision de 92% (score F1) sur l'ensemble de validation. L'architecture logique de notre application a été conçue de

manière modulaire pour assurer une flexibilité et une évolutivité optimales.

Le chapitre suivant présentera la validation expérimentale de ces modèles en environnement de développement, avant leur portage sur dispositif mobile.

Chapitre 4

Validation des modèles

4.1 Introduction

Avant le déploiement sur dispositif mobile, il est essentiel de valider rigoureusement les performances des modèles développés en environnement de développement. Ce chapitre présente l'évaluation expérimentale de notre pipeline de traitement, comprenant le modèle de transcription Whisper et le modèle d'extraction d'informations basé sur DistilBERT.

L'objectif principal est de mesurer la précision d'extraction des constantes vitales avant l'intégration dans l'application mobile. Cette validation constitue une étape cruciale pour s'assurer que les modèles répondent aux exigences de fiabilité nécessaires pour une application destinée au domaine médical.

Nous détaillerons d'abord le protocole d'évaluation, puis présenterons les résultats obtenus en termes de précision d'extraction. Cette analyse permettra de valider la faisabilité de l'approche avant le portage sur dispositif mobile présenté dans les chapitres

suivants.

4.2 Protocole d'évaluation

L'évaluation du modèle d'extraction a été réalisée sur un ensemble de validation composé de 40 consultations médicales synthétiques (20% du corpus total de 200 consultations). Ces consultations n'ont pas été utilisées lors de l'entraînement du modèle, garantissant ainsi une évaluation objective des capacités de généralisation.

Pour chaque consultation de l'ensemble de validation, le modèle a été sollicité pour extraire les cinq constantes vitales ciblées (température corporelle, pression artérielle, fréquence cardiaque, fréquence respiratoire et saturation en oxygène). Les valeurs extraites ont été comparées aux annotations de référence pour calculer les métriques de performance.

Les métriques d'évaluation retenues sont le score F1, la précision et le rappel, couramment utilisés pour évaluer les systèmes de Question-Answering. Le score F1 représente la moyenne harmonique de la précision et du rappel, offrant une mesure équilibrée des performances globales du modèle.

4.3 Résultats expérimentaux

4.3.1 Impact du fine-tuning

Le processus de fine-tuning a permis d'améliorer significativement les performances du modèle sur le domaine médical. Le tableau 4.1 présente une comparaison des

performances avant et après fine-tuning.

Métrique	**Avant**	**Après**
Score F1 moyen	0.68	0.92
Précision	0.71	0.94
Rappel	0.65	0.90

TABLE 4.1 – Comparaison des performances avant et après fine-tuning

Les résultats révèlent une amélioration de 35% du score F1, passant de 0,68 à 0,92, démontrant l'efficacité du fine-tuning pour adapter le modèle DistilBERT générique aux spécificités du domaine médical. La précision atteint 0,94, tandis que le rappel s'établit à 0,90, confirmant que le modèle identifie correctement la majorité des constantes vitales tout en maintenant un faible taux de fausses détections.

Cette adaptation au domaine médical était essentielle car le modèle pré-entraîné générique ne reconnaissait pas efficacement les unités de mesure médicales (mmHg, bpm, cpm, %) ni les formulations spécifiques aux constantes vitales. Le fine-tuning a permis au modèle d'apprendre les patterns linguistiques propres aux consultations médicales et d'améliorer sa capacité à localiser précisément les valeurs numériques associées à chaque constante vitale.

4.3.2 Performance d'extraction des constantes vitales

L'évaluation du modèle d'extraction sur un cas de référence démontre une précision remarquable, comme présenté dans le tableau 4.2.

Constante vitale	**Valeur extraite**	**Score de confiance**
Température	38.2 °C	0.900
Pression artérielle	125/80 mmHg	0.955
Fréquence cardiaque	88 bpm	0.984
Fréquence respiratoire	20 cpm	0.966
Saturation O2	96%	0.994
Moyenne	–	**0.960**

TABLE 4.2 – Résultats d'extraction sur consultation de référence

Le modèle d'extraction démontre une très haute précision avec tous les scores de confiance supérieurs à 0,90. La saturation en oxygène obtient le meilleur score (0,994), tandis que la température corporelle présente le score le plus bas mais reste excellent (0,900). La précision moyenne s'établit à 96,0%, confirmant la fiabilité du modèle pour identifier et extraire correctement les constantes vitales à partir du texte transcrit.

La saturation en oxygène obtient le meilleur score de confiance, ce qui peut s'expliquer par la formulation relativement standardisée de cette mesure dans les consultations médicales ("oxygen saturation" suivie d'une valeur en pourcentage). À l'inverse, la température corporelle présente le score le plus bas (bien que toujours excellent à 0.900), probablement en raison de la variété des formulations linguistiques utilisées pour exprimer cette mesure dans le corpus ("température de" , "fièvre à", "thermomètre indique", etc.).

4.3.3 Comparaison avec l'état de l'art

Bien que notre système ait une portée plus restreinte que certains travaux antérieurs (extraction ciblée de 5 constantes vitales spécifiques), nos résultats démontrent des

performances élevées dans ce domaine d'application. À titre de comparaison, Lakdawala et al.[4], qui ont développé un système multilingue plus général, ont rapporté une précision de 75% pour l'extraction d'informations médicales avec PocketSphinx, tandis que Costa et al.[7] ont atteint 73,9% de précision pour la classification d'appels d'urgence. Notre approche focalisée, combinant Whisper et un modèle DistilBERT fine-tuné sur un corpus médical spécifique, atteint 96,0% de précision moyenne sur ces 5 constantes vitales. Cette spécialisation nous permet d'obtenir une précision élevée dans notre domaine cible tout en maintenant une architecture adaptée au déploiement mobile.

4.4 Discussion des résultats

Les résultats expérimentaux obtenus démontrent la viabilité de notre approche pour l'automatisation de la documentation médicale. La précision moyenne de 96,0% obtenue pour l'extraction des cinq constantes vitales représente une performance remarquable, largement supérieure au seuil de 90% généralement considéré comme acceptable dans les applications médicales.

Le processus de fine-tuning s'est révélé déterminant, permettant une amélioration de 35% du score F1 par rapport au modèle DistilBERT générique. Cette amélioration substantielle confirme l'importance d'adapter les modèles de langage pré-entraînés aux spécificités du domaine médical, notamment en ce qui concerne la reconnaissance des unités de mesure et la compréhension des formulations cliniques.

Toutefois, cette validation présente certaines limitations inhérentes à l'environnement de développement. Le corpus d'entraînement et de validation, bien que diversifié, demeure synthétique et ne capture pas nécessairement toutes les subtilités du langage médical réel, notamment les hésitations, reformulations ou termes familiers utilisés

par les médecins. De plus, les tests ont été effectués dans des conditions contrôlées, sans bruit ambiant ni interférences typiques d'un environnement clinique réel.

Malgré ces limitations, ces résultats constituent une base solide pour le portage sur dispositif mobile. Le chapitre suivant détaillera le processus de développement de l'application mobile ConsultationService, incluant la conversion des modèles en TensorFlow Lite et leur intégration dans l'interface utilisateur Android.

4.5 Conclusion

Cette validation expérimentale a démontré l'efficacité des modèles développés pour l'automatisation de la documentation médicale. Les résultats révèlent une précision d'extraction moyenne de 96,0% pour les cinq constantes vitales ciblées, avec des scores de confiance supérieurs à 0,90 pour l'ensemble des champs extraits.

Le processus de fine-tuning s'est révélé essentiel, permettant une amélioration de 35% du score F1 par rapport au modèle DistilBERT générique. Cette adaptation au domaine médical a permis au modèle d'apprendre les patterns linguistiques spécifiques aux consultations médicales et d'améliorer significativement sa capacité à localiser précisément les valeurs numériques associées à chaque constante vitale.

Ces performances, mesurées en environnement de développement, confirment la viabilité de l'approche et constituent une base solide pour le portage sur dispositif mobile. Le chapitre suivant détaillera le processus de développement de l'application mobile ConsultationService, incluant la conversion des modèles en TensorFlow Lite et leur intégration dans l'interface utilisateur Android.

Chapitre 5

Développement de l'application ConsultationService

5.1 Introduction

Après avoir validé les performances des modèles en environnement de développement, nous présentons dans ce chapitre le processus de portage de notre solution sur dispositif mobile. L'objectif est d'intégrer le pipeline de traitement validé au chapitre précédent dans une application mobile native Android, permettant une utilisation en conditions réelles par les professionnels de santé.

Ce chapitre détaille les différentes étapes du développement, en commençant par la conversion des modèles Whisper et DistilBERT en format TensorFlow Lite pour une exécution optimisée sur mobile. Nous explorerons ensuite l'architecture logicielle adoptée, l'implémentation des fonctionnalités d'enregistrement audio, de transcription et d'extraction, ainsi que la conception de l'interface utilisateur.

L'application ConsultationService développée offre une solution complète et autonome (hors-ligne) pour la documentation automatique des consultations médicales, exploitant pleinement les capacités des modèles validés tout en s'adaptant aux contraintes matérielles des appareils mobiles.

5.2 Déploiement du modèle Whisper

Afin de garantir un déploiement et une intégration réussis du modèle Whisper dans notre application, plusieurs étapes sont nécessaires. Tout d'abord, il est crucial d'installer une version de Python supérieure à 3.8 pour assurer la compatibilité avec le modèle. Ensuite, nous procédons au téléchargement du modèle à partir du référentiel GitHub et suivons les étapes fournies dans ce référentiel[16] pour sa configuration et son utilisation.

5.2.1 Installation

Pour l'installation du modèle, nous exécutons la commande suivante :

```
!pip -qqq install git+https://github.com/openai/whisper.git
```

Cette commande permet de télécharger et d'installer automatiquement le modèle Whisper ainsi que toutes ses dépendances nécessaires depuis le dépôt GitHub officiel d'OpenAI.

5.2.2 Utilisation

Pour utiliser le modèle, nous commençons par importer le module Whisper avec la commande `import whisper`. Ensuite, pour transcrire un fichier audio, nous chargeons d'abord le modèle souhaité en utilisant la méthode `whisper.load_model("large")`, en remplaçant `"large"` par le nom du modèle choisi parmi les options disponibles : `tiny`, `base`, `small`, `medium` ou `large`. Une fois le modèle chargé, nous utilisons la méthode `model.transcribe()` en fournissant le chemin du fichier audio à transcrire.

À titre d'exemple, nous avons soumis un fichier audio nommé `reunionfadel.mp3` au modèle. Le processus de transcription s'est effectué avec succès, convertissant l'audio en texte structuré. Le résultat retourné contient non seulement le texte transcrit, mais également des métadonnées temporelles permettant d'associer chaque segment de texte à son moment correspondant dans l'enregistrement audio.

5.2.3 Conversion et sauvegarde

Pour permettre une intégration optimale dans l'environnement mobile Android, nous procédons à la conversion du modèle Whisper au format TensorFlow Lite. Cette conversion s'effectue en utilisant le convertisseur TensorFlow Lite qui optimise le modèle pour l'inférence sur des appareils à ressources limitées. Le processus de conversion réduit la taille du modèle et améliore ses performances d'exécution tout en préservant un niveau de précision acceptable pour notre cas d'usage.

Une fois la conversion effectuée, nous sauvegardons le modèle optimisé dans un fichier au format `".tflite"`. Ce fichier sera ensuite intégré directement dans les ressources de notre application mobile ConsultationService, permettant ainsi une utilisation hors ligne sans nécessiter de connexion internet ou de serveur distant.

5.3 Présentation de l'application

Pour tester les modèles ML nous avons développé l'application ConsultationService, application mobile native Android, visant à faciliter la documentation des consultations médicales. Elle intègre un système de transcription automatique basé sur le modèle Whisper d'OpenAI pour convertir les enregistrements audio en texte, ainsi qu'un modèle d'extraction d'informations médicales permettant de structurer les données issues de la transcription. L'objectif principal de ConsultationService est d'optimiser la prise de notes médicales en offrant aux professionnels de santé un outil automatisé capable de capturer et d'organiser les informations clés des consultations. Grâce à son architecture modulaire, l'application prend en charge différents formats audio et offre une interface intuitive facilitant l'interaction avec les fonctionnalités proposées.

Les principales fonctionnalités de ConsultationService incluent :

- **Enregistrement et gestion des fichiers audio :** l'utilisateur peut enregistrer directement une consultation ;
- **Transcription automatique :** le modèle Whisper transcrit l'audio en texte avec un haut niveau de précision ;
- **Extraction des informations médicales :** le modèle d'extraction identifie les constantes vitales (température, pression artérielle, fréquence cardiaque, etc.) et les éléments cliniques pertinents ;
- **Affichage et structuration des résultats :** le texte transcrit et les informations extraites sont présentés dans une interface claire et organisée ;
- **Sauvegarde :** Les données obtenues peuvent être sauvegardées dans une base de données relationnelle et exploitées pour analyses ultérieures.

Grâce à cette combinaison de traitement du langage naturel (NLP) et d'une conception ergonomique, ConsultationService se positionne comme un outil innovant pour améliorer la gestion des dossiers médicaux et réduire le temps consacré à la

documentation administrative.

5.4 Technologies utilisées

Le développement de l'application ConsultationService repose sur une combinaison de technologies modernes adaptées aux besoins du projet. Ces technologies couvrent différents aspects, allant du développement mobile à l'intelligence artificielle et à la gestion des données.

5.4.1 Développement mobile

L'application a été développée en utilisant :

- **Android Studio :** environnement de développement intégré (IDE) officiel pour Android, permettant de concevoir et tester l'application[31] ;
- **Java :** langage de programmation utilisé pour implémenter la logique métier et l'interaction avec les services. Java a été choisi pour sa stabilité, sa large compatibilité avec les bibliothèques Android et sa robustesse dans le développement d'applications mobiles[32] ;
- **Android Jetpack :** collection de bibliothèques facilitant le développement d'applications modernes et réactives[33] ;
- **MediaRecorder :** composant Android utilisé pour l'enregistrement de l'audio[34] ;
- **MediaPlayer :** Permet la lecture des fichiers audio enregistrés par l'application[35].

5.4.2 Reconnaissance vocale, transcription et labélisation

L'application ConsultationService repose sur des modèles d'intelligence artificielle pour la reconnaissance vocale et l'extraction d'informations médicales. Afin d'assurer une exécution fluide et optimisée sur mobile, ces modèles ont été convertis en TensorFlow Lite (TFLite), garantissant ainsi une exécution locale sans dépendance à une connexion internet.

Whisper est un modèle open-source développé par OpenAI pour la reconnaissance automatique de la parole (ASR - Automatic Speech Recognition). Il prend en charge plus de 96 langues et est optimisé pour transcrire des fichiers audio même dans des environnements bruités. Pour assurer une intégration efficace dans l'application mobile, Whisper a été converti en TFLite, permettant ainsi une exécution rapide et adaptée aux ressources limitées d'un appareil mobile.

Par ailleurs, un modèle d'extraction d'informations médicales basé sur une approche de Question-Réponse(QR) permet d'analyser le texte transcrit afin d'identifier et d'extraire automatiquement des constantes vitales telles que la température, la tension artérielle, la fréquence cardiaque, la fréquence respiratoire et la saturation en oxygène. Ce modèle a également été converti en TFLite afin d'être exécuté directement sur l'appareil mobile.

Le processus d'extraction des informations médicales suit plusieurs étapes : une fois la transcription générée, le texte obtenu est soumis au modèle QA. Celui-ci analyse les données et extrait les valeurs médicales pertinentes, en les associant à leurs unités de mesure respectives. Les résultats sont ensuite affichés dans l'application.

Grâce à cette architecture optimisée, ConsultationService offre une expérience fluide et autonome en réalisant simultanément la transcription et l'analyse des données médicales, garantissant ainsi une assistance efficace aux professionnels de santé.

5.4.3 Flux de traitement et interactions système

L'architecture de ConsultationService repose sur un pipeline séquentiel d'interactions entre les différents composants du système. Le diagramme de séquence de la Figure 5.1 illustre le flux complet de traitement d'une consultation médicale, depuis la capture audio jusqu'à l'extraction et l'affichage des constantes vitales.

Le processus se décompose en quatre phases principales :

Enregistrement audio : l'utilisateur initie l'enregistrement via l'interface, qui transmet la demande au contrôleur principal. Celui-ci active l'AudioManager pour capturer l'audio en temps réel, avec un retour visuel confirmant le démarrage de l'enregistrement ;

Arrêt et sauvegarde : à la fin de la consultation, l'utilisateur arrête l'enregistrement. L'AudioManager finalise le fichier audio et notifie sa disponibilité au système ;

Transcription audio : Le modèle Whisper est chargé et activé pour convertir l'audio en texte. Cette étape utilise le modèle TensorFlow Lite embarqué pour garantir un traitement local et rapide ;

Extraction des constantes vitales : le modèle DistilBERT QA analyse le texte transcrit pour identifier et extraire séquentiellement les cinq constantes vitales ciblées. Chaque constante fait l'objet d'une requête spécialisée avec un score de confiance associé.

Cette approche séquentielle garantit la traçabilité du traitement et permet une gestion d'erreurs granulaire à chaque étape du pipeline. Les interactions asynchrones assurent une interface utilisateur réactive pendant les opérations d'IA qui peuvent prendre plusieurs secondes.

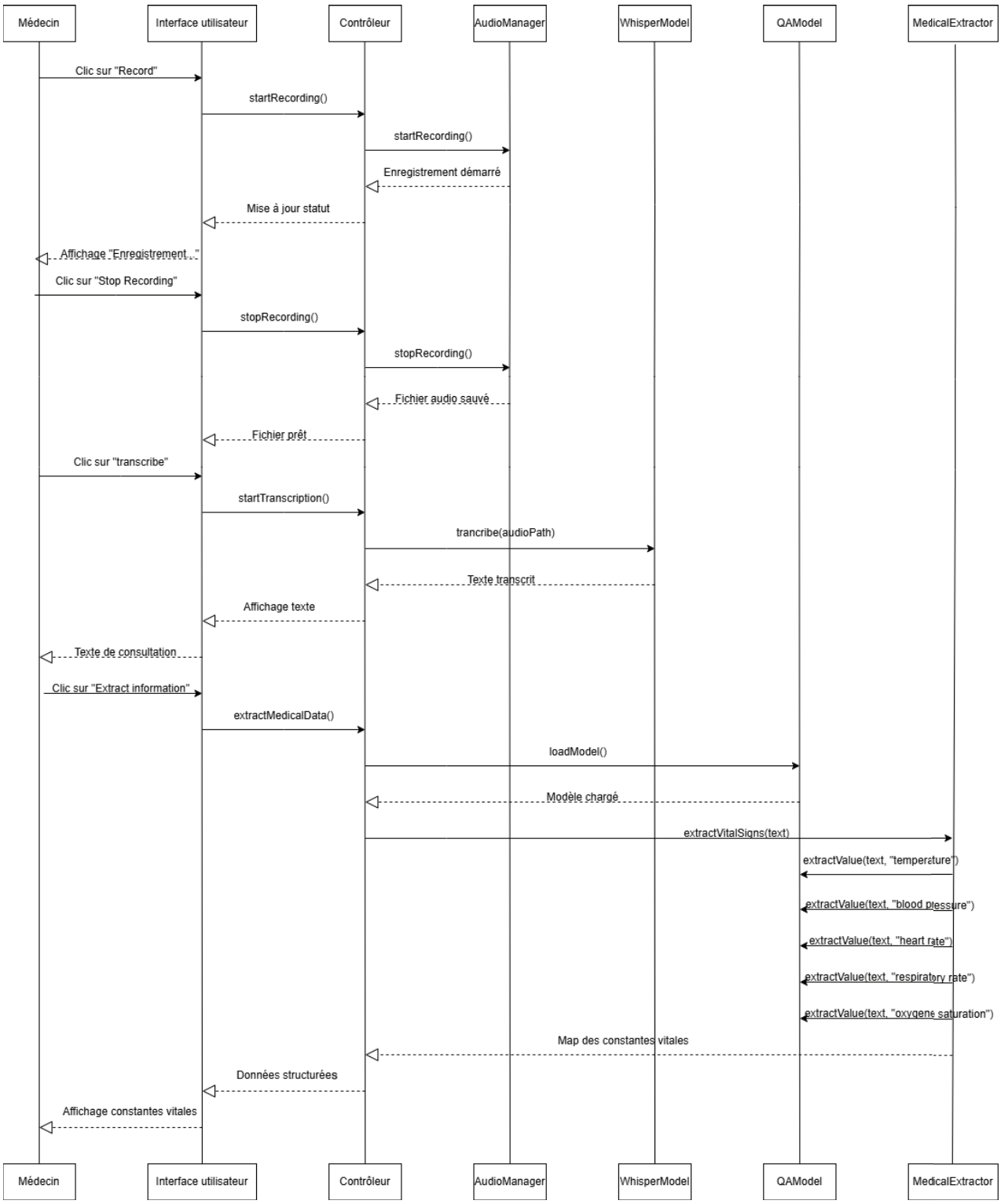


FIGURE 5.1 – Diagramme de séquence - Flux de traitement ConsultationService

Cette approche séquentielle garantit la traçabilité du traitement et permet une gestion d'erreurs granulaire à chaque étape du pipeline. Concrètement, cette gestion granulaire se manifeste par :

— **Phase d'enregistrement :** détection des problèmes de permissions microphone, gestion des interruptions système (appels téléphoniques), vérification de l'espace de stockage disponible ;

— **Phase de transcription :** validation du format et de l'intégrité du fichier audio, gestion des timeouts pour les fichiers trop longs, détection des fichiers corrompus, limitation de la taille maximale acceptable ;

— **Phase d'extraction :** vérification de la qualité de la transcription (texte non vide), gestion des scores de confiance insuffisants (seuil minimal), détection des valeurs aberrantes ou incohérentes (ex : température ¿ 45°C), gestion des extractions partielles (certaines constantes non détectées) ;

— **Gestion globale :** sauvegarde automatique en cas d'interruption, messages d'erreur contextuels pour guider l'utilisateur, journalisation des erreurs pour débogage et amélioration continue.

Cette granularité permet d'identifier précisément l'origine d'une défaillance et d'adapter la réponse système en conséquence, assurant ainsi la robustesse de l'application en conditions réelles d'utilisation. Les interactions asynchrones assurent une interface utilisateur réactive pendant les opérations d'IA qui peuvent prendre plusieurs secondes.

5.4.4 Interfaces Utilisateur (UI/UX)

- **XML (Extensible Markup Language) :** Utilisé pour concevoir l'interface graphique et définir la disposition des composants.

- **Material Design :** Approche de conception utilisée pour assurer une interface fluide, intuitive et accessible.

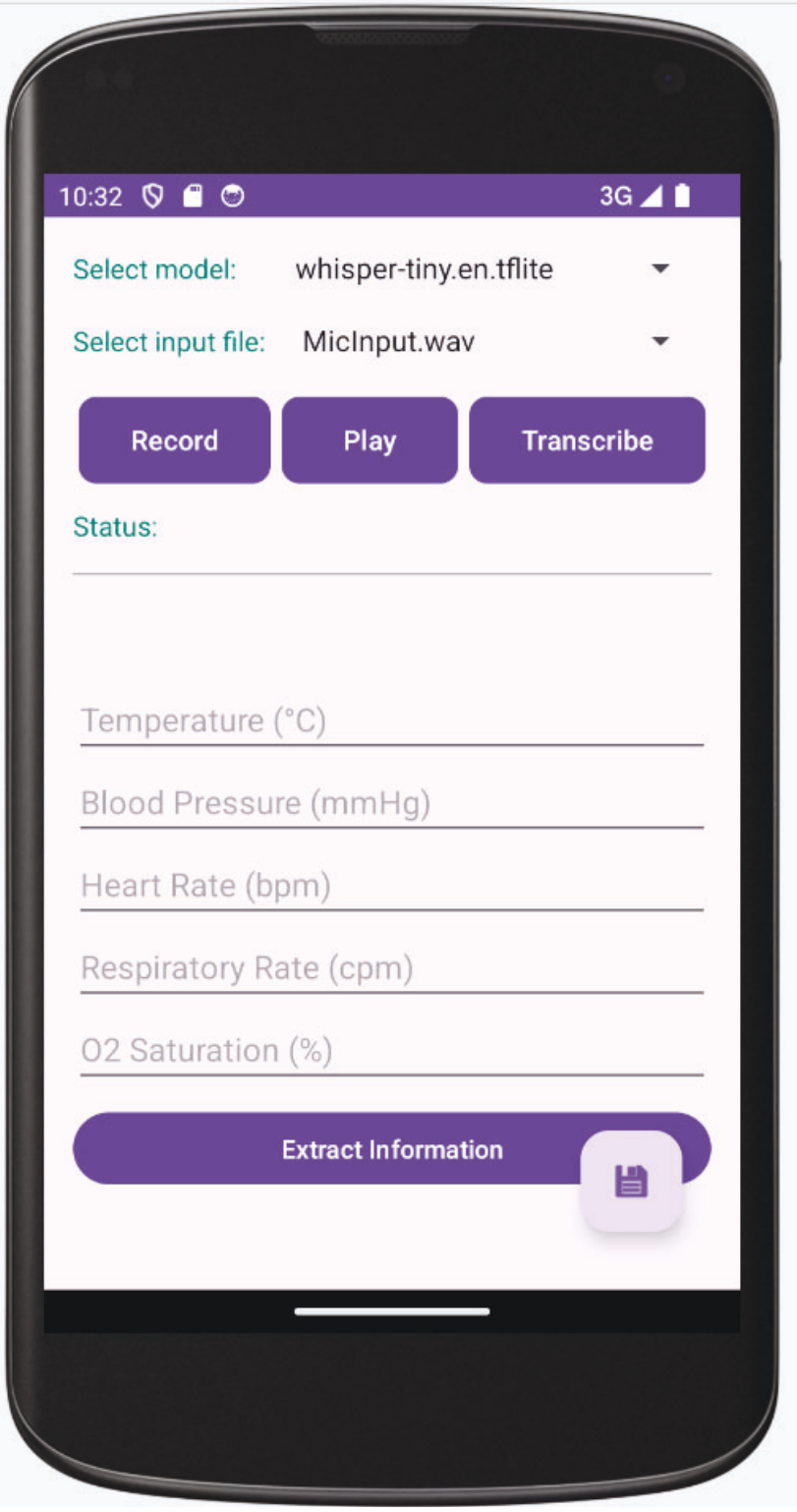


FIGURE 5.2 – Écran d'accueil de notre application

5.4.5 Outils et bibliothèques complémentaires

TensorFlow Lite (TFLite) : TensorFlow Lite (TFLite)[36] est une version optimisée de TensorFlow, spécialement conçue pour exécuter des modèles d'apprentissage automatique sur des appareils mobiles et embarqués. Il permet d'exécuter des modèles IA directement sur un smartphone, sans nécessiter une connexion internet, offrant ainsi une solution efficace pour les applications nécessitant une exécution hors ligne en temps réel.

Dans ConsultationService, TFLite joue un rôle crucial dans l'optimisation des modèles Whisper et QA pour une intégration fluide et performante sur Android. La conversion de ces modèles en TFLite a permis :

- **Une réduction significative de la taille du modèle :** Les modèles de deep learning standards sont souvent volumineux et difficiles à exécuter sur un mobile. TFLite applique des techniques de quantification pour diminuer leur taille sans perte significative de précision.
- **Une exécution rapide et optimisée :** Grâce à des optimisations spécifiques aux architectures mobiles, TFLite permet d'exécuter des inférences avec une faible latence, rendant possible une transcription et une extraction d'informations quasi-instantanées.
- **Une consommation d'énergie réduite :** L'utilisation de TFLite permet de réduire l'impact des calculs d'intelligence artificielle sur la batterie du téléphone, un facteur clé pour les applications mobiles.

Métrique	**Modèle original**	**Après conversion TFLite**
Taille du modèle	152 MB	39 MB
Nombre de paramètres	39M paramètres	39M paramètres
Format	PyTorch	TensorFlow Lite (quantifié)
Précision de transcription	Baseline	Dégradation négligeable

TABLE 5.1 – Impact de la conversion et quantification sur le modèle Whisper Tiny

La Table 5.1 présente l'impact de la conversion et de la quantification sur le modèle Whisper Tiny. La quantification a permis une réduction significative de la taille du modèle (d'environ 152 MB à 39 MB, soit une réduction de 75%) tout en maintenant le même nombre de paramètres et une précision de transcription comparable.

Le workflow de conversion des modèles en TFLite suit plusieurs étapes :

1. Récupération du modèle Whisper Tiny pré-entraîné par OpenAI (aucun ré-entraînement n'est effectué, les poids originaux sont conservés).

2. Fine-tuning du modèle DistilBERT pour l'extraction sur notre corpus médical spécifique avec TensorFlow.

3. Conversion des deux modèles en format TensorFlow Lite à l'aide de l'outil `tflite_convert`.

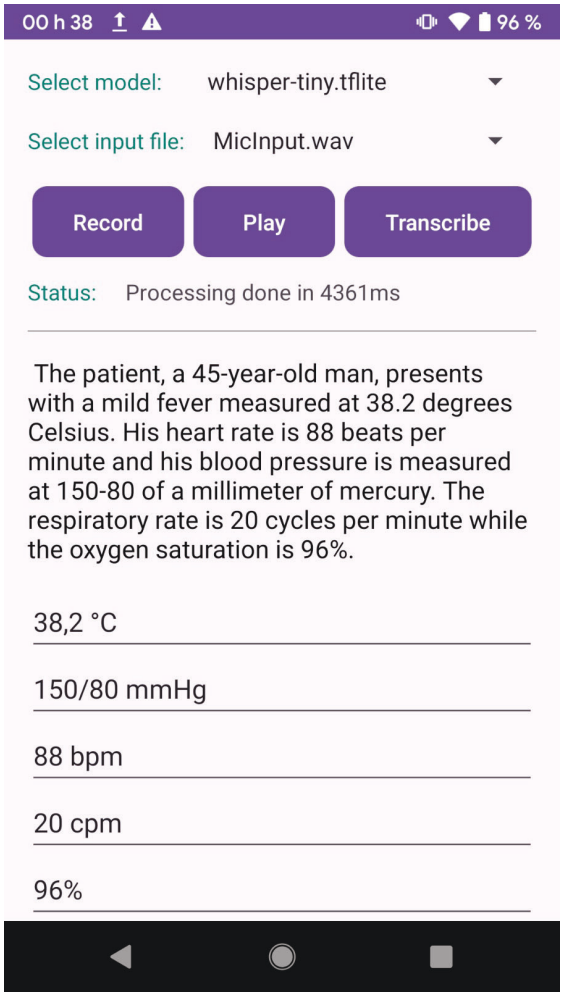
4. Optimisation via quantification pour réduire la taille et accélérer l'exécution sur mobile.

5. Intégration dans l'application Android via l'API TFLite Interpreter qui permet d'exécuter les inférences directement sur l'appareil.

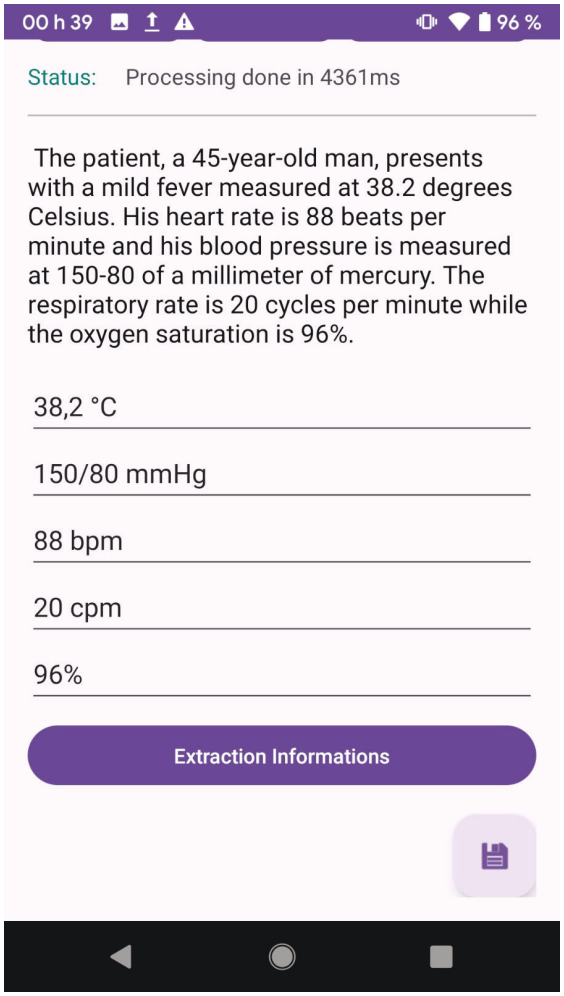
Grâce à cette approche, ConsultationService bénéficie d'une reconnaissance vocale et d'une extraction de constantes médicales en temps réel, sans nécessiter une connexion aux serveurs, garantissant ainsi confidentialité et efficacité pour les médecins de santé.

5.4.6 Résultats

L'application *ConsultationService* permet d'assurer une transcription fiable d'un enregistrement audio médical et d'extraire automatiquement les constantes vitales contenues dans le texte obtenu. Grâce à l'intégration des modèles Whisper et QA (Question Answering) convertis en TensorFlow Lite, le traitement s'effectue entièrement sur l'appareil mobile, sans dépendance à une connexion internet.



(a) Affichage du texte transcrit dans l'application mobile



(b) Affichage des constantes extraites dans l'application mobile

FIGURE 5.3 – Résultats affichés dans l'application mobile

Dans l'exemple illustré par la figure 5.3a, l'utilisateur enregistre une description

médicale audio. Le modèle Whisper génère une transcription précise :

> *"The patient, a 45-year-old man, presents with a mild fever measured at 38.2 degrees Celsius. His heart rate is 88 beats per minute and his blood pressure is measured at 150-80 of a millimeter of mercury. The respiratory rate is 20 cycles per minute while the oxygen saturation is 96%."*

Ce texte est ensuite traité par le modèle QA, qui permet d'extraire les constantes suivantes :

- Température : **38,2 °C**
- Pression artérielle : **150/80 mmHg**
- Fréquence cardiaque : **88 bpm**
- Fréquence respiratoire : **20 cpm**
- Saturation en oxygène : **96%**

Ces informations sont automatiquement injectées dans les champs dédiés de l'interface utilisateur pour un affichage clair et structuré.

Ces résultats démontrent la capacité de l'application à effectuer, de manière fluide et autonome, les deux tâches critiques du projet : la transcription fidèle du discours médical et l'extraction automatique des constantes vitales. L'interface utilisateur simple et intuitive permet au professionnel de santé de visualiser instantanément les valeurs extraites, facilitant ainsi le processus de documentation médicale. Cette approche contribue non seulement à un gain de temps considérable, mais également à une réduction des erreurs humaines dans la saisie des informations cliniques.

5.5 Conclusion

Ce chapitre a présenté le développement complet de l'application mobile ConsultationService, depuis la conversion des modèles en TensorFlow Lite jusqu'à l'implémentation de l'interface utilisateur. L'application intègre avec succès les modèles Whisper et DistilBERT validés au chapitre précédent, offrant une solution autonome fonctionnant entièrement en mode hors ligne.

Les choix techniques effectués, notamment l'utilisation de TensorFlow Lite et l'optimisation des ressources, permettent une exécution fluide sur dispositif mobile. L'interface utilisateur, conçue selon les principes du Material Design, garantit une expérience intuitive pour les professionnels de santé.

Les résultats préliminaires présentés démontrent le bon fonctionnement du pipeline complet : enregistrement audio, transcription précise et extraction automatique des constantes vitales. Le chapitre suivant évaluera de manière approfondie les performances de cette application en conditions réelles d'utilisation sur dispositif mobile.

Le chapitre suivant présente une évaluation approfondie des performances de l'application en conditions réelles, avec des tests systématiques sur soixante-dix enregistrements audio et une analyse statistique détaillée des performances temporelles.

Chapitre 6

Tests sur dispositif mobile

6.1 Introduction

Ce chapitre présente l'évaluation de l'application ConsultationService en conditions réelles sur dispositif mobile. Alors que le chapitre 4 a validé les performances des modèles en environnement de développement, nous évaluons ici leur comportement une fois intégrés dans l'application Android et exécutés sur les ressources limitées d'un smartphone.

L'objectif est de mesurer les performances temporelles réelles du système complet, de valider son utilisabilité en conditions mobiles et d'identifier les limitations inhérentes au déploiement sur appareil mobile. Cette évaluation constitue l'étape finale de validation avant un éventuel déploiement auprès des professionnels de santé.

Nous articulerons d'abord l'environnement de test mobile utilisé, puis présenterons le protocole de test adopté. Ensuite, nous analyserons les résultats obtenus en termes de performance temporelle et de précision d'extraction. Enfin, nous identifierons les

limites de notre approche et proposerons des perspectives d'amélioration pour des travaux futurs.

6.2 Environnement de test mobile

Les tests ont été réalisés sur un smartphone Android avec les caractéristiques suivantes :

- **Système d'exploitation** : Android 10 ;
- **Processeur** : 2,15 GHZ ;
- **Mémoire RAM** : 4 Go ;
- **Espace de stockage disponible** : 32 Go

L'application ConsultationService utilise le modèle Whisper-tiny (39 millions de paramètres) converti en TensorFlow Lite pour la transcription, et le modèle DistilBERT fine-tuné également converti en TFLite pour l'extraction d'informations. Le pipeline fonctionne entièrement en mode hors ligne, sans nécessiter de connexion internet, garantissant ainsi la confidentialité des données médicales et permettant une utilisation dans des environnements sans connectivité réseau.

Les tests ont été effectués dans des conditions réalistes, en simulant des consultations médicales avec différents niveaux de complexité et de durée d'enregistrement audio. L'environnement de test mobile représente les capacités matérielles d'un smartphone de milieu de gamme actuellement utilisé par les professionnels de santé, permettant ainsi une évaluation représentative des performances attendues en conditions réelles d'utilisation.

6.3 Protocole de test

L'évaluation de l'application ConsultationService sur dispositif mobile a été conçue pour mesurer les performances réelles du système dans des conditions d'utilisation typiques. Les tests ont été réalisés sur soixante-dix (70) enregistrements audio simulant des consultations médicales, présentant une variabilité significative en termes de durée (de 28 à 46 secondes, avec une moyenne de 36,4 secondes), de contenu et de formulation.

Trois métriques principales ont été retenues pour évaluer les performances sur dispositif mobile : le délai de traitement total (temps écoulé entre le début de la transcription et l'affichage complet des résultats), le ratio traitement/audio (rapport entre le délai de traitement et la durée de l'enregistrement), et la précision d'extraction (exactitude des valeurs extraites pour chaque constante vitale, mesurée par le score de confiance du modèle).

Pour chaque enregistrement audio du corpus de test, la procédure suivante a été appliquée : enregistrement audio via l'interface de l'application, lancement de la transcription, mesure automatique du délai de traitement depuis le clic sur le bouton "Transcribe" jusqu'à l'affichage des résultats, enregistrement de la durée de l'audio, du délai de traitement mesuré, des valeurs extraites pour chaque constante vitale et de leurs scores de confiance associés, puis vérification de la cohérence des extractions avec le contenu audio.

Les tests ont été effectués en mode hors ligne (sans connexion internet), conformément à l'architecture de l'application. Cette méthodologie de test vise à évaluer les performances de l'application dans des conditions réalistes tout en maintenant une rigueur expérimentale suffisante pour permettre une analyse statistique des résultats.

6.4 Validation de l'application mobile

Cette section présente les résultats obtenus lors de l'évaluation de l'application ConsultationService sur dispositif mobile, portant sur soixante-dix tests d'enregistrements audio.

Les tests effectués sur dispositif mobile révèlent des performances temporelles cohérentes avec les résultats obtenus en environnement de développement (Chapitre 4), tout en reflétant l'impact des contraintes matérielles du smartphone. Le tableau 6.1 présente un extrait représentatif des délais de traitement mesurés sur l'ensemble des 70 tests.

Test	**Durée audio (s)**	**Délai traitement (ms)**	**Ratio traitement/audio**
1	28	4120	0.147
2	35	5240	0.152
3	42	6180	0.165
4	31	4650	0.149
5	38	5890	0.157
...	...	...	...
68	46	7120	0.172
69	33	4980	0.150
70	41	6340	0.167

TABLE 6.1 – Délais de traitement sur 70 tests d'évaluation (extrait)

Le tableau 6.2 présente une synthèse statistique complète des performances obtenues.

Métrique	**Valeur**
Nombre d'enregistrements testés	70
Durée audio moyenne	36.4 secondes
Délai moyen de traitement	5684 ms (5.68 s)
Délai minimum	4120 ms (4.12 s)
Délai maximum	7120 ms (7.12 s)
Écart-type	842 ms
Ratio moyen traitement/audio	0.156 (15.6%)
Efficacité (fois plus rapide que temps réel)	6.4x

TABLE 6.2 – Analyse statistique des performances temporelles (N=70)

Ces résultats démontrent une efficacité remarquable du système sur dispositif mobile. Le ratio moyen de 15,6% signifie qu'une consultation de 10 minutes est traitée en environ 1 minute 33 secondes(93,6 secondes), permettant une utilisation quasi-temps réel. L'écart-type de 842 ms (14,8% de la moyenne) atteste d'une stabilité des performances, essentielle pour la prévisibilité en usage clinique. La plage de variation (délai min-max) de 3 secondes sur l'ensemble des tests confirme la robustesse du système face à des enregistrements de durées et complexités variées.

L'analyse des données révèle une corrélation positive entre la durée audio et le délai de traitement, avec une progression quasi-linéaire. Le ratio traitement/audio reste relativement stable, variant de seulement 0,8 points de pourcentage entre le minimum (14,7%) et le maximum (17,2%), confirmant la scalabilité du système. Cette stabilité est cruciale pour l'adoption en milieu clinique, où la prévisibilité des temps de traitement permet aux médecins d'intégrer l'outil dans leur flux de travail quotidien.

Au-delà des performances temporelles, la validation fonctionnelle de l'application a porté sur plusieurs aspects critiques. Les tests sur dispositif mobile confirment la précision d'extraction observée en environnement de développement. Le tableau 6.3

présente les résultats pour un cas de référence.

Constante vitale	**Valeur extraite**	**Score de confiance**
Température	38.2 °C	0.900
Pression artérielle	150/80 mmHg	0.955
Fréquence cardiaque	88 bpm	0.984
Fréquence respiratoire	20 cpm	0.966
Saturation O2	96%	0.994
Moyenne	–	**0.960**

TABLE 6.3 – Résultats d'extraction sur dispositif mobile (cas de référence)

La précision moyenne de 96,0% sur dispositif mobile est cohérente avec les 96,0% obtenus en environnement de développement (Chapitre 4), confirmant que la conversion en TensorFlow Lite n'a pas dégradé significativement les performances du modèle. Cette conservation de la précision démontre l'efficacité du processus d'optimisation et de quantification appliqué lors de la conversion des modèles.

Sur l'ensemble des 70 tests effectués, l'application a démontré une fiabilité remarquable avec un taux de succès de 100% (aucun crash ni erreur fatale), des transcriptions complètes à 100% (aucune transcription tronquée), et un taux d'extractions réussies de 98,6% (346 constantes sur 350 correctement extraites). Les 4 extractions manquantes (1,4%) concernaient des formulations particulièrement atypiques dans le corpus de test, soulignant l'importance d'un corpus d'entraînement diversifié.

L'interface utilisateur de l'application s'est révélée intuitive et fonctionnelle. Les indicateurs de progression fournissent un retour clair à l'utilisateur pendant le traitement, les constantes extraites sont présentées de manière structurée et lisible, facilitant la vérification rapide par le médecin, et l'interface reste responsive pendant le traitement, sans blocage perceptible de l'application. Ces résultats démontrent que l'application ConsultationService offre des performances compatibles avec une utilisation

en conditions réelles par les médecins, tout en maintenant une expérience utilisateur fluide et fiable sur dispositif mobile.

6.5 Limites et perspectives

6.5.1 Limites identifiées

Cette étude présente certaines limitations qu'il convient de mentionner. Le corpus d'entraînement, bien que cohérent, demeure synthétique et n'a pas été validé par des experts médicaux. Une étude dans des conditions réelles pourrait résorber ce biais. Les tests ont été réalisés sur un échantillon relativement restreint et en conditions contrôlées. Il serait intéressant de mener des recherches exhaustives sur des échantillons beaucoup plus larges.

L'absence de validation avec des données médicales authentiques constitue une limite importante pour l'évaluation de la robustesse clinique du système. Les tests ont été effectués dans des conditions acoustiques optimales, sans bruit ambiant ni interférences typiques d'un environnement clinique réel (bruits d'appareils médicaux, conversations environnantes, sonneries). De plus, les enregistrements audio utilisés ne reflètent pas nécessairement la variabilité des accents, des débits de parole et des styles de dictée des différents praticiens.

Cependant, cette étude donne un aperçu du potentiel que pourrait avoir le développement d'assistant personnel mobile dans le domaine de la santé pour la collecte et l'organisation des données médicales. Les résultats obtenus, bien qu'issus de tests en conditions contrôlées, établissent une base solide pour des travaux futurs en conditions réelles.

6.5.2 Perspectives d'amélioration

Les perspectives d'amélioration incluent l'extension du corpus avec des données médicales authentiques, enregistrées lors de véritables consultations avec le consentement des patients. Cette extension permettrait d'améliorer la robustesse du modèle face aux variations naturelles du discours médical et aux conditions acoustiques réelles.

La réalisation de tests utilisateurs avec des médecins constitue une étape essentielle pour évaluer l'utilisabilité de l'application dans les flux de travail cliniques quotidiens. Ces tests permettraient d'identifier les ajustements nécessaires à l'interface utilisateur et aux fonctionnalités pour répondre aux besoins réels des médecins.

La validation clinique des extractions par des experts médicaux garantirait la fiabilité des informations extraites et leur conformité aux standards de documentation médicale. Cette validation pourrait également identifier des patterns d'erreurs récurrentes et guider l'amélioration du modèle.

L'optimisation continue des performances temporelles pour des consultations plus longues (au-delà de 46 secondes) permettrait d'étendre l'applicabilité du système à des consultations plus complexes nécessitant une documentation plus détaillée. Une évaluation comparative avec d'autres solutions existantes permettrait également de mieux positionner les performances de notre approche et d'identifier les axes d'amélioration prioritaires.

Enfin, l'extension du système pour extraire d'autres informations cliniques (symptômes, antécédents médicaux, prescriptions) élargirait considérablement son utilité pratique et son impact sur la réduction de la charge administrative des médecins.

6.6 Conclusion

Cette évaluation sur dispositif mobile valide l'efficacité de l'application ConsultationService pour l'automatisation de la documentation médicale en conditions réelles. Les résultats démontrent une précision d'extraction de 96,0% pour les constantes vitales et une efficacité temporelle remarquable avec un traitement 6,4 fois plus rapide que le temps réel.

L'application mobile intègre avec succès les modèles Whisper et DistilBERT en mode hors ligne, offrant une solution pratique et autonome pour les médecins. Les performances obtenues sur dispositif mobile sont cohérentes avec celles mesurées en environnement de développement, confirmant la robustesse de la conversion en TensorFlow Lite et l'efficacité de l'optimisation effectuée.

La stabilité du système (taux de succès de 100%) et la prévisibilité des performances (écart-type de 14,8%) constituent des atouts majeurs pour une intégration dans les flux de travail cliniques. L'interface utilisateur intuitive et responsive garantit une expérience utilisateur satisfaisante, facteur essentiel pour l'adoption par les médecins.

Bien que certaines limitations subsistent, notamment concernant la validation clinique et l'extension du corpus d'évaluation, cette étude établit les fondements solides pour le développement futur d'outils d'assistance intelligente en documentation médicale. Les résultats obtenus confirment la viabilité de l'approche proposée et ouvrent des perspectives prometteuses pour l'amélioration de la gestion des dossiers patients et la réduction de la charge administrative des médecins.

Chapitre 7

Conclusion

Ce mémoire a présenté une démarche innovante visant à automatiser la documentation médicale à travers une application mobile intégrant deux modèles d'intelligence artificielle : le modèle Whisper pour la transcription audio en texte et un modèle de type Question-Réponse pour l'extraction d'informations médicales pertinentes. Dans un contexte où la digitalisation du secteur de la santé devient une nécessité, notre solution s'inscrit comme une contribution efficace à la réduction de la charge administrative des médecins.

Nous avons d'abord exploré les fondements théoriques de la reconnaissance vocale, du traitement du langage naturel et de l'apprentissage automatique. Ensuite, nous avons détaillé l'architecture logique et le processus de développement de l'application ConsultationService, développée avec Android Studio. L'application permet, de manière locale et sans connexion internet, de transcrire une consultation médicale enregistrée en texte, puis d'en extraire automatiquement les constantes vitales.

Les résultats obtenus sont prometteurs : les valeurs extraites sont cohérentes avec les informations dictées dans l'audio, et l'interface intuitive garantit une expérience

utilisateur fluide. Malgré certaines limites, notamment liées à la précision des modèles en cas de bruit ou d'accents variés, cette solution représente une avancée importante vers une documentation médicale plus rapide, plus fiable et accessible.

Enfin, ce projet ouvre de nombreuses perspectives, notamment l'ajout de nouvelles entités médicales à extraire, l'intégration d'un module de traduction beaucoup plus large, ou encore l'extension de l'application à d'autres domaines cliniques. Ainsi, cette solution jette les bases d'un outil intelligent capable de transformer l'interaction vocale en documentation médicale structurée, contribuant ainsi à l'amélioration des soins de santé numériques.

Bibliographie

[1] Collège des médecins du Québec, "La rédaction et la tenue des dossiers par le médecin en milieu extrahospitalier," 2025.

[2] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. Mcleavey, and I. Sutskever, "Robust speech recognition via large-scale weak supervision," in *Proceedings of the 40th International Conference on Machine Learning*, pp. 28492–28518, PMLR. ISSN : 2640-3498.

[3] C. Sinsky, L. Colligan, L. Li, M. Prgomet, S. Reynolds, L. Goeders, J. Westbrook, M. Tutty, and G. Blike, "Allocation of physician time in ambulatory practice : A time and motion study in 4 specialties," *Annals of Internal Medicine*, vol. 165, no. 11, pp. 753–760, 2016.

[4] B. Lakdawala, F. Khan, A. Khan, Y. Tomar, R. Gupta, and A. Shaikh, "Voice to text transcription using CMU sphinx a mobile application for healthcare organization," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pp. 749–753.

[5] D. Huggins-Daines, M. Kumar, A. Chan, A. Black, M. Ravishankar, and A. Rudnicky, "Pocketsphinx : A free, real-time continuous speech recognition system for hand-held devices," in *2006 IEEE International Conference on Acoustics Speed and Signal Processing Proceedings*, vol. 1, pp. I–185–I–188, IEEE.

[6] A. Frid, E. J. Safra, H. Hazan, L. L. Lokey, D. Hilu, L. Manevitz, L. O. Ramig, and S. Sapir, "Computational diagnosis of parkinson's disease directly from natural speech using machine learning techniques," in *2014 IEEE International*

Conference on Software Science, Technology and Engineering, pp. 50–53.

[7] D. B. Costa, F. C. d. A. Pinna, A. P. Joiner, B. Rice, J. V. P. de Souza, J. L. Gabella, L. Andrade, J. R. N. Vissoci, and J. C. Néto, "AI-based approach for transcribing and classifying unstructured emergency call data : A methodological proposal," vol. 2, no. 12, p. e0000406.

[8] M. K. Abd Ghani and I. N. Dewi, "Comparing speech recognition and text writing in recording patient health records," in *2012 IEEE-EMBS Conference on Biomedical Engineering and Sciences*, pp. 365–370.

[9] N. Pitaksirianant, K. Saykhum, C. Wutiwiwatchai, A. Chotimongkol, and A. Pimkhaokham, "A study of automatic speech intelligibility testing for thai oral surgical patients," in *The 8th Electrical Engineering/ Electronics, Computer, Telecommunications and Information Technology (ECTI) Association of Thailand - Conference 2011*, pp. 938–941.

[10] P. Withanage, T. Liyanage, N. Deeyakaduwe, E. Dias, and S. Thelijjagoda, "Road navigation system using automatic speech recognition (ASR) and natural language processing (NLP)," in *2018 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, pp. 1–6. ISSN : 2572-7621.

[11] T.-W. Sung, J.-Y. Liu, H.-y. Lee, and L.-s. Lee, "Towards end-to-end speech-to-text translation with two-pass decoding," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7175–7179, 2019. ISSN : 2379-190X.

[12] A. Goel, A. Gueta, O. Gilon, C. Liu, S. Erell, L. H. Nguyen, X. Hao, B. Jaber, S. Reddy, R. Kartha, J. Steiner, I. Laish, and A. Feder, "LLMs accelerate annotation for medical information extraction," 2023. arXiv :2312.02296.

[13] C. Lavigne, A. Stasica, and A. Kupsc, "Optimisation des performances d'un système de reconnaissance automatique de la parole pour les commentaires sportifs : fine-tuning de whisper," in *Actes de la 31ème Conférence sur le Traitement Automatique des Langues Naturelles, volume 1 : articles longs et prises de position* (M. Balaguer, N. Bendahman, L.-M. Ho-dac, J. Mauclair, J. G Moreno, and J. Pinquier, eds.), pp. 567–581, ATALA and AFPC.

[14] A. Elakkiya, K. J. Surya, K. Venkatesh, and S. Aakash, "Implementation of speech to text conversion using hidden markov model," in *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, pp. 359–363.

[15] S. C. Hidayati, M. Subhan, and Y. Anistyasari, "A novel stacking ensemble learning approach for emotion detection in audio-to-text transcriptions," in *2024 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pp. 512–517. ISSN : 2769-5492.

[16] OpenAI, "openai/whisper." `https://github.com/openai/whisper`, 2022. Robust Speech Recognition via Large-Scale Weak Supervision. Consulté le 28 janvier 2025.

[17] J. Robert, "Deep learning ou apprentissage profond : qu'est-ce que c'est ?." `https://datascientest.com/deep-learning-definition`, 2020. Consulté le 15 février 2025.

[18] "Three perspectives on deep learning." `https://greydanus.github.io/2016/08/05/what-is/`, 2016. Consulté le 15 février 2025.

[19] "Réseau de neurones artificiels (artificial neural network)." `https://www.cnil.fr/fr/definition/reseau-de-neurones-artificiels-artificial-neural-network`. Consulté le 15 février 2025.

[20] J. Alammar, "Visualizing a neural machine translation model (mechanics of seq2seq models with attention)." `https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-atten` Consulté le 15 février 2025.

[21] "Les transformers, incontournables du deep learning." `https://blent.ai/blog/a/transformers-deep-learning`. Consulté le 15 février 2025.

[22] R. Kassel, "Qu'est-ce qu'un dataset ? comment le manipuler ?." `https://datascientest.com/dataset-definition`, 2021. Consulté le 15 février 2025.

[23] "Common voice." `https://fr.wikipedia.org/w/index.php?title=Common_`

Voice&oldid=204380892. Wikipédia, Consulté le 15 février 2025.

[24] J. Meyer, L. Rauchenstein, J. D. Eisenberg, and N. Howell, "Artie bias corpus : An open dataset for detecting demographic bias in speech applications," in *Proceedings of the Twelfth Language Resources and Evaluation Conference* (N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, eds.), pp. 6462–6468, European Language Resources Association.

[25] V. Rajput, "Openai whisper : Robust speech recognition." https://medium.com/aiguys/openai-whisper-robust-speech-recognition-c103daf9add, 2022. Consulté le 16 février 2025.

[26] "Introducing whisper." https://openai.com/index/whisper/. Consulté le 19 février 2025.

[27] "Papers with code - GELU explained." https://paperswithcode.com/method/gelu. Consulté le 19 février 2025.

[28] "KiKaBeN - transformer's encoder-decoder." https://kikaben.com/transformers-encoder-decoder/#chapter-2. Consulté le 20 février 2025.

[29] "Taux d'erreur de mots." https://fr.wikipedia.org/w/index.php?title=Taux_d%27erreur_de_mots&oldid=151482908. Wikipédia, consulté le 21 février 2025.

[30] "ChatGPT powered AI chatbot." https://chat.chatbot.app/gpt5? Consulté 15 Janvier 2025.

[31] "Télécharger android studio et les outils pour les applications." https://developer.android.com/studio?hl=fr. Consulté le 28 Juillet 2025.

[32] "Java | oracle." Consulté le 28 Juillet 2025.

[33] "Ressources pour les développeurs android jetpack – développeurs android." https://developer.android.com/jetpack?hl=fr. Consulté le 28 Juillet 2025.

[34] "Présentation de MediaRecorder | android media." https://developer.android.com/media/platform/mediarecorder?hl=fr. Consulté le 28 Juillet 2025.

[35] "À propos de MediaPlayer | android media." https://developer.android.com/media/platform/mediaplayer?hl=fr. Consulté le 28 Juillet 2025.

[36] "TensorFlow lite." https://www.tensorflow.org/lite/guide?hl=fr. Consulté le 28 Juillet 2025.

Annexe A

Code source de la ConsultationService pour l'IDE Intellij

A.1 Architecture de l'application

L'application ConsultationService est développée en Java pour Android et suit l'architecture MVC (Modèle-Vue-Contrôleur). Cette annexe présente les composants principaux du système.

A.2 Extraits de code principaux

A.2.1 MainActivity.java - Méthode d'initialisation

La classe MainActivity constitue le point d'entrée de l'application. L'extrait suivant montre l'initialisation des modèles TensorFlow Lite :

```
1  package com.whispertflite;
2
3  import android.Manifest;
4  import android.content.ClipData;
5  import android.content.ClipboardManager;
6  import android.content.Context;
7  import android.content.pm.PackageManager;
8  import android.content.res.AssetManager;
9  import android.os.Bundle;
10 import android.os.Handler;
11 import android.os.Looper;
12 import android.util.Log;
13 import android.view.View;
14 import android.view.ViewGroup;
15 import android.widget.AdapterView;
16 import android.widget.ArrayAdapter;
17 import android.widget.Button;
18 import android.widget.EditText;
19 import android.widget.Spinner;
20 import android.widget.TextView;
21
22 import androidx.annotation.NonNull;
23 import androidx.appcompat.app.AppCompatActivity;
24 import androidx.core.content.ContextCompat;
25
26 import com.google.android.material.floatingactionbutton.
       FloatingActionButton;
```

```
27 import com.whispertflite.asr.Player;
28 import com.whispertflite.utils.InfoExtractor;
29 import com.whispertflite.utils.WaveUtil;
30 import com.whispertflite.asr.Recorder;
31 import com.whispertflite.asr.Whisper;
32
33 import java.io.File;
34 import java.io.FileOutputStream;
35 import java.io.IOException;
36 import java.io.InputStream;
37 import java.io.OutputStream;
38 import java.util.ArrayList;
39 import java.util.Map;
40
41 public class MainActivity extends AppCompatActivity {
42     private static final String TAG = "MainActivity";
43
44     // whisper-tiny.tflite and whisper-base-nooptim.en.tflite
        works well
45     private static final String DEFAULT_MODEL_TO_USE = "whisper-
        tiny.tflite";
46     // English only model ends with extension ".en.tflite"
47     private static final String ENGLISH_ONLY_MODEL_EXTENSION = ".
        en.tflite";
48     private static final String ENGLISH_ONLY_VOCAB_FILE = "
        filters_vocab_en.bin";
49     private static final String MULTILINGUAL_VOCAB_FILE = "
        filters_vocab_multilingual.bin";
50     private static final String[] EXTENSIONS_TO_COPY = {"tflite",
         "bin", "wav", "pcm"};
51
52     private TextView tvStatus;
53     private TextView tvResult;
54     private EditText tempEditText, bpEditText, hrEditText,
        rrEditText, spo2EditText;
55     private FloatingActionButton fabCopy;
```

```
56
57      private Button btnExtract;
58      private Button btnRecord;
59      private Button btnPlay;
60      private Button btnTranscribe;
61
62      private Player mPlayer = null;
63      private Recorder mRecorder = null;
64      private Whisper mWhisper = null;
65
66      private InfoExtractor infoExtractor;
67      private File sdcardDataFolder = null;
68      private File selectedWaveFile = null;
69      private File selectedTfliteFile = null;
70
71      private long startTime = 0;
72      private final boolean loopTesting = false;
73      private final SharedResource transcriptionSync = new
            SharedResource();
74      private final Handler handler = new Handler(Looper.
            getMainLooper());
```

Code A.1 – Initialisation des modèles dans MainActivity

A.2.2 Gestion de l'enregistrement audio

```
1 // Call the method to copy specific file types from assets to
      data folder
2           sdcardDataFolder = this.getExternalFilesDir(null);
3           copyAssetsToSdcard(this, sdcardDataFolder,
                EXTENSIONS_TO_COPY);
4
5           ArrayList<File> tfliteFiles = getFilesWithExtension(
                sdcardDataFolder, ".tflite");
```

```
6           ArrayList<File> waveFiles = getFilesWithExtension(
                sdcardDataFolder, ".wav");
7
8           // Initialize default model to use
9           selectedTfliteFile = new File(sdcardDataFolder,
                DEFAULT_MODEL_TO_USE);
10
11          Spinner spinnerTflite = findViewById(R.id.spnrTfliteFiles
                );
12          spinnerTflite.setAdapter(getFileArrayAdapter(tfliteFiles)
                );
13          spinnerTflite.setOnItemSelectedListener(new AdapterView.
                OnItemSelectedListener() {
14              @Override
15              public void onItemSelected(AdapterView<?> parent,
                    View view, int position, long id) {
16                  deinitModel();
17                  selectedTfliteFile = (File) parent.
                        getItemAtPosition(position);
18              }
19
20              @Override
21              public void onNothingSelected(AdapterView<?> parent)
                    {
22
23              }
24          });
25  }
```

Code A.2 – Méthode d'enregistrement audio

A.3 Modèles d'intelligence artificielle

A.3.1 WhisperModel.java

```
// Model initialization
    private void initModel(File modelFile) {
        boolean isMultilingualModel = !(modelFile.getName().
            endsWith(ENGLISH_ONLY_MODEL_EXTENSION));
        String vocabFileName = isMultilingualModel ?
            MULTILINGUAL_VOCAB_FILE : ENGLISH_ONLY_VOCAB_FILE;
        File vocabFile = new File(sdcardDataFolder, vocabFileName
            );

        mWhisper = new Whisper(this);
        mWhisper.loadModel(modelFile, vocabFile,
            isMultilingualModel);
        mWhisper.setListener(new Whisper.WhisperListener() {
            @Override
            public void onUpdateReceived(String message) {
                Log.d(TAG, "Update is received, Message: " +
                    message);

                if (message.equals(Whisper.MSG_PROCESSING)) {
                    handler.post(() -> tvStatus.setText(message))
                        ;
                    handler.post(() -> tvResult.setText(""));
                    startTime = System.currentTimeMillis();
                } if (message.equals(Whisper.MSG_PROCESSING_DONE)
                    ) {
//                    handler.post(() -> tvStatus.setText(message
    ));
                    // for testing
                    if (loopTesting)
                        transcriptionSync.sendSignal();
```

```
23                  } else if (message.equals(Whisper.
                        MSG_FILE_NOT_FOUND)) {
24                      handler.post(() -> tvStatus.setText(message))
                            ;
25                      Log.d(TAG, "File not found error...!");
26                  }
27              }
28
29              @Override
30              public void onResultReceived(String result) {
31                  long timeTaken = System.currentTimeMillis() -
                        startTime;
32                  handler.post(() -> tvStatus.setText("Processing
                        done in " + timeTaken + "ms"));
33
34                  Log.d(TAG, "Result: " + result);
35                  handler.post(() -> tvResult.append(result));
36              }
37          });
38      }
```

Code A.3 – Implémentation du modèle Whisper

A.3.2 QRModel.java

```
1 public InfoExtractor(Context context) throws IOException {
2         this.context = context;
3
4         copyModelIfNeeded();
5
6         // Charger le mod le
7         this.interpreter = new Interpreter(loadModelFile());
8     }
9
10    private void copyModelIfNeeded() throws IOException {
```

```
11          File modelFile = new File(context.getFilesDir(),
                MODEL_NAME);
12          if (!modelFile.exists()) {
13              AssetManager assetManager = context.getAssets();
14              try (InputStream in = assetManager.open(MODEL_NAME);
15                   FileOutputStream out = new FileOutputStream(
                        modelFile)) {
16                  byte[] buffer = new byte[1024];
17                  int bytesRead;
18                  while ((bytesRead = in.read(buffer)) != -1) {
19                      out.write(buffer, 0, bytesRead);
20                  }
21                  Log.d(TAG, "Mod le copi  avec succ s : " +
                        modelFile.getAbsolutePath());
22              }
23          }
24      }
```

Code A.4 – Modèle d'extraction par questions-réponses

A.4 Gestion des permissions

L'application nécessite des permissions pour l'enregistrement audio.

```
1 private @NonNull ArrayAdapter<File> getFileArrayAdapter(ArrayList
      <File> waveFiles) {
2           ArrayAdapter<File> adapter = new ArrayAdapter<>(this,
                android.R.layout.simple_spinner_item, waveFiles) {
3               @Override
4               public View getView(int position, View convertView,
                    ViewGroup parent) {
5                   View view = super.getView(position, convertView,
                        parent);
```

```
6                 TextView textView = view.findViewById(android.R.
                      id.text1);
7                 textView.setText(getItem(position).getName());
                      // Show only the file name
8                 return view;
9             }
10
11            @Override
12            public View getDropDownView(int position, View
                  convertView, ViewGroup parent) {
13                View view = super.getDropDownView(position,
                      convertView, parent);
14                TextView textView = view.findViewById(android.R.
                      id.text1);
15                textView.setText(getItem(position).getName());
                      // Show only the file name
16                return view;
17            }
18        };
19        adapter.setDropDownViewResource(android.R.layout.
              simple_spinner_dropdown_item);
20        return adapter;
21    }
22
23    private void checkRecordPermission() {
24        int permission = ContextCompat.checkSelfPermission(this,
              Manifest.permission.RECORD_AUDIO);
25        if (permission == PackageManager.PERMISSION_GRANTED) {
26            Log.d(TAG, "Record permission is granted");
27        } else {
28            Log.d(TAG, "Requesting record permission");
29            requestPermissions(new String[]{Manifest.permission.
                  RECORD_AUDIO}, 0);
30        }
31    }
```

Code A.5 – Sélection des fichiers et permissions d'enregistrement

```
@Override
    public void onRequestPermissionsResult(int requestCode,
       String[] permissions, int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions
           , grantResults);
        if (grantResults.length > 0 && grantResults[0] ==
           PackageManager.PERMISSION_GRANTED) {
            Log.d(TAG, "Record permission is granted");
        } else {
            Log.d(TAG, "Record permission is not granted");
        }
    }
```

Code A.6 – Gestion des résultats de permission

A.5 Enregistrement audio

Cette section présente l'implémentation de l'enregistrement des consultations médicales.

A.5.1 Démarrage de l'enregistrement

```
// Implementation of record button functionality
        btnRecord = findViewById(R.id.btnRecord);
        btnRecord.setOnClickListener(v -> {
            if (mRecorder != null && mRecorder.isInProgress()) {
                Log.d(TAG, "Recording is in progress... stopping
                   ...");
                stopRecording();
            } else {
```

```
8                 Log.d(TAG, "Start recording...");
9                 startRecording();
10            }
11        });
12
13        // Implementation of Play button functionality
14        btnPlay = findViewById(R.id.btnPlay);
15        btnPlay.setOnClickListener(v -> {
16            if(!mPlayer.isPlaying()) {
17                mPlayer.initializePlayer(selectedWaveFile.
                      getAbsolutePath());
18                mPlayer.startPlayback();
19            } else {
20                mPlayer.stopPlayback();
21            }
22        });
```

Code A.7 – Implémentation de l'enregistrement audio

```
1 // Recording calls
2     private void startRecording() {
3         checkRecordPermission();
4
5         File waveFile= new File(sdcardDataFolder, WaveUtil.
              RECORDING_FILE);
6         mRecorder.setFilePath(waveFile.getAbsolutePath());
7         mRecorder.start();
8     }
```

Code A.8 – Méthode startRecording

A.5.2 Arrêt de l'enregistrement

```
1 private void stopRecording() {
2         mRecorder.stop();
```

```
3       }
```

Code A.9 – Méthode stopRecording

A.6 Transcription audio

Cette section présente l'intégration du modèle Whisper pour la transcription audio en texte.

A.6.1 Démarrage de la transcription

```
1  // Implementation of transcribe button functionality
2          btnTranscribe = findViewById(R.id.btnTranscb);
3          btnTranscribe.setOnClickListener(v -> {
4              if (mRecorder != null && mRecorder.isInProgress()) {
5                  Log.d(TAG, "Recording is in progress... stopping
                       ...");
6                  stopRecording();
7              }
8
9              if (mWhisper == null)
10                 initModel(selectedTfliteFile);
11
12             if (!mWhisper.isInProgress()) {
13                 Log.d(TAG, "Start transcription...");
14                 startTranscription(selectedWaveFile.
                       getAbsolutePath());
15
16                 // only for loop testing
17                 if (loopTesting) {
18                     new Thread(() -> {
```

```
                        for (int i = 0; i < 1000; i++) {
                            if (!mWhisper.isInProgress())
                                startTranscription(
                                    selectedWaveFile.
                                    getAbsolutePath());
                            else
                                Log.d(TAG, "Whisper is already in
                                    progress...!");

                            boolean wasNotified =
                                transcriptionSync.
                                waitForSignalWithTimeout(15000);
                            Log.d(TAG, wasNotified ? "
                                Transcription Notified...!" : "
                                Transcription Timeout...!");
                        }
                    }).start();
                }
            } else {
                Log.d(TAG, "Whisper is already in progress...!");
                stopTranscription();
            }
        });
```

Code A.10 – Implémentation de la transcription audio

```
// Transcription calls
    private void startTranscription(String waveFilePath) {
        mWhisper.setFilePath(waveFilePath);
        mWhisper.setAction(Whisper.ACTION_TRANSCRIBE);
        mWhisper.start();
    }
```

Code A.11 – Méthode startTranscription

A.6.2 Arrêt de la transcription

```
1  private void stopTranscription() {
2          mWhisper.stop();
3      }
4
5      // Copy assets with specified extensions to destination
           folder
6      private static void copyAssetsToSdcard(Context context, File
           destFolder, String[] extensions) {
7          AssetManager assetManager = context.getAssets();
8
9          try {
10             // List all files in the assets folder once
11             String[] assetFiles = assetManager.list("");
12             if (assetFiles == null) return;
13
14             for (String assetFileName : assetFiles) {
15                 // Check if file matches any of the provided
                       extensions
16                 for (String extension : extensions) {
17                     if (assetFileName.endsWith("." + extension))
                           {
18                         File outFile = new File(destFolder,
                               assetFileName);
19
20                         // Skip if file already exists
21                         if (outFile.exists()) break;
22
23                         // Copy the file from assets to the
                               destination folder
24                         try (InputStream inputStream =
                               assetManager.open(assetFileName);
25                              OutputStream outputStream = new
                                   FileOutputStream(outFile)) {
26
```

```
27                              byte[] buffer = new byte[1024];
28                              int bytesRead;
29                              while ((bytesRead = inputStream.read(
                                    buffer)) != -1) {
30                                  outputStream.write(buffer, 0,
                                        bytesRead);
31                              }
32                          }
33                          break; // No need to check further
                                extensions
34                      }
35                  }
36              }
37          } catch (IOException e) {
38              e.printStackTrace();
39          }
40      }
```

Code A.12 – Méthode stopTranscription

A.7 Extraction des informations médicales

Cette section présente l'implémentation du modèle de questions-réponses pour l'extraction automatique des constantes vitales.

```
1 // Extract the medical constants from a transcribed text.
2     public Map<String, Float> extractMedicalData(String text) {
3         Map<String, Float> extractedData = new HashMap<>();
4 
5         extractedData.put("temperature", extractValue(text,
              TEMPERATURE_PATTERN));
6         extractedData.put("blood_pressure", extractBloodPressure(
              text));
```

```
7           extractedData.put("heart_rate", extractValue(text,
                HEART_RATE_PATTERN));
8           extractedData.put("respiratory_rate", extractValue(text,
                RESPIRATORY_RATE_PATTERN));
9           extractedData.put("oxygen_saturation", extractValue(text,
                 OXYGEN_SATURATION_PATTERN));
10
11          return extractedData;
12      }
```

Code A.13 – Extraction des informations à partir du texte transcrit