Institut National de Statistique et d'Économie Appliquée

Département Informatique

1ère année Master Systèmes d'Information et Systèmes Intelligents

Système de Génération de Résumés Multilingues par IA Générative : Du Texte Long au Résumé Structuré

Présenté par :

OUCHAKE Younes

Encadré par : **El Karfi Ikram**

Soutenu le :

22/05/2025

Résumé

Le projet "Générateur de Résumés Multilingue" est une application développée pour répondre au problème de surcharge informationnelle actuel en proposant une synthèse automatique de documents dans plusieurs langues (français, anglais, arabe). L'outil traite trois types de contenus - texte brut, fichiers PDF et enregistrements audio - grâce à une interface Streamlit intuitive et des technologies d'intelligence artificielle avancées comme les modèles Transformers de Hugging Face et Whisper pour la transcription. Destinée aux chercheurs, étudiants, journalistes et professionnels, cette solution permet de générer rapidement des résumés personnalisables (longueur, structure) tout en préservant la cohérence sémantique du contenu original, offrant ainsi un gain de temps considérable dans le traitement de gros volumes documentaires multilingues.

Table des matières

Table des figures

Liste des tableaux

Introduction

Dans un monde où l'information se multiplie de façon exponentielle, la capacité à synthétiser rapidement et efficacement de grands volumes de texte est devenue une compétence cruciale. Le projet "Générateur de Résumés Multilingue" s'inscrit dans cette dynamique en proposant une solution innovante permettant de traiter automatiquement des documents dans plusieurs langues et formats pour en extraire l'essentiel.

La croissance rapide du contenu numérique, estimée à plus de 2,5 quintillions d'octets de données créées chaque jour, rend de plus en plus difficile pour les utilisateurs de trier et d'assimiler l'information pertinente. C'est dans ce contexte que les outils de synthèse automatique deviennent non seulement utiles mais essentiels.

Notre application vise à résoudre plusieurs problématiques contemporaines :

- La surcharge informationnelle qui affecte les professionnels et étudiants
- Les barrières linguistiques limitant l'accès à l'information
- La diversité des formats de documents nécessitant des traitements spécifiques
- Le besoin d'extraire rapidement les points clés de longs documents ou enregistrements audio

Le public cible de notre solution comprend les chercheurs, étudiants, professionnels de l'information, journalistes, analystes et toute personne confrontée à la nécessité de synthétiser rapidement des documents dans différentes langues et formats.

Pour répondre à ces besoins, nous avons développé une application reposant sur des technologies de pointe en matière de traitement automatique du langage naturel, notamment les modèles de transformers de Hugging Face, intégrés dans une interface conviviale développée avec Streamlit. Notre système prend en charge non seulement le texte brut, mais également les documents PDF et les fichiers audio, offrant ainsi une solution complète et polyvalente.

Ce rapport présente en détail la conception, l'implémentation et l'évaluation de notre générateur de résumés multilingue, organisé en sept chapitres couvrant l'analyse des besoins, les technologies utilisées, l'implémentation, les fonctionnalités, l'évaluation, le déploiement et les perspectives futures.

Chapitre 1

Analyse des besoins et conception

1.1 Introduction à l'analyse des besoins

Dans un monde où l'information prolifère de façon exponentielle, la capacité à extraire efficacement les éléments essentiels d'un contenu est devenue une compétence précieuse. Notre projet « Générateur de Résumés Multilingue » répond à ce besoin croissant en proposant une solution automatisée pour la synthèse de contenu textuel et audio dans plusieurs langues. Cette section présente l'analyse détaillée des besoins qui a guidé la conception de notre application.

1.2 Étude du marché des outils de résumé automatique

1.2.1 Panorama des solutions existantes

Le marché des outils de résumé automatique peut être segmenté en plusieurs catégories :

- **Solutions grand public :** Des applications comme Summly (acquis par Yahoo), Summize, ou les fonctionnalités de résumé intégrées aux navigateurs comme Edge qui offrent des fonctionnalités basiques de synthèse de texte.
- **Outils professionnels :** Des plateformes comme Quillbot, TLDR This ou Resoomer qui fournissent des résumés plus sophistiqués avec une certaine personnalisation.
- **API et services cloud :** Des services comme AWS Comprehend, Google Cloud Natural Language ou Azure Language Understanding qui offrent des capacités de résumé par API.

1.2.2 Analyse comparative des fonctionnalités clés

L'analyse de marché a révélé plusieurs lacunes dans les solutions existantes comme le montre le tableau 1.1.

1.2.3 Analyse SWOT des solutions existantes

Forces

- **Accessibilité et facilité d'utilisation :** de nombreuses solutions sont disponibles en ligne ou intégrées dans des outils courants, ce qui facilite leur adoption par un large public.

TABLE 1.1 – Analyse comparative des fonctionnalités

Fonctionnalité	**Solutions existantes**	**Notre solution**
Support multilingue	Limité (souvent 2-3 langues)	Extensif (français, anglais, arabe)
Traitement de PDF structurés	Basique, sans reconnaissance de structure	Avancé avec détection de structure
Intégration audio	Rare ou séparée	Intégrée directement
Personnalisation de longueur	Binaire (court/long)	Graduelle avec visualisation
Résumés structurés	Non disponible	Structure par sections avec points clés

- **Bonne intégration dans des écosystèmes logiciels :** les solutions existantes s'intègrent souvent efficacement aux environnements bureautiques, aux plateformes cloud ou aux outils collaboratifs.

Faiblesses

- **Support linguistique restreint :** la plupart des solutions privilégient l'anglais et offrent un support limité pour d'autres langues, notamment les langues à faible ressource comme l'arabe.
- **Compatibilité limitée avec les formats de documents :** certains outils ne prennent en charge que les textes simples, excluant les PDF, images ou contenus multimodaux.
- **Absence de traitement structurel avancé :** les résumés générés manquent souvent de structure (titres, sections) et de contextualisation.

Opportunités

- **Demande croissante pour l'analyse de contenus multimédias :** l'explosion des données sous forme de texte, audio, vidéo ou image crée un besoin fort en solutions de résumé intelligentes.
- **Besoins accrus en solutions multilingues :** les entreprises et institutions recherchent des outils capables de traiter des documents en plusieurs langues de manière fluide et fiable.

Menaces

- **Évolution rapide des grands modèles de langage (LLMs) :** de nouveaux modèles plus puissants et open source émergent constamment, pouvant rendre obsolètes les solutions actuelles.
- **Concurrence des suites logicielles intégrées :** les géants du numérique (Google, Microsoft, etc.) intègrent progressivement des fonctionnalités de résumé dans leurs produits, réduisant l'avantage des solutions spécialisées.

1.3 Identification des besoins utilisateurs

1.3.1 Profils utilisateurs ciblés

Notre application vise principalement trois profils d'utilisateurs :

Professionnels de l'information : Chercheurs, journalistes, analystes qui doivent traiter rapidement de grandes quantités de texte.

Étudiants et universitaires : Pour synthétiser des articles scientifiques, des cours ou des ressources pédagogiques.

Professionnels multilingues : Personnes travaillant avec des documents dans différentes langues nécessitant une compréhension rapide du contenu.

1.3.2 Cartographie du parcours utilisateur

L'analyse du parcours utilisateur a mis en lumière plusieurs points de friction dans les solutions actuelles :

1. **Phase de sélection de la source :** Difficulté à naviguer entre différents outils, souvent nécessaires pour traiter plusieurs formats de documents.
2. **Phase de configuration :** Manque de clarté sur les options disponibles et leurs effets réels sur le résultat final.
3. **Phase d'attente :** Incertitude persistante concernant l'avancement du traitement, notamment pour les documents volumineux, générant frustration et impatience.
4. **Phase d'exploitation :** Obstacles rencontrés lors du partage ou de l'export des résumés, limitant la fluidité du flux de travail.

1.4 Exigences fonctionnelles et non-fonctionnelles

1.4.1 Exigences fonctionnelles

Traitement multiformat

- L'application doit pouvoir traiter du texte brut saisi directement par l'utilisateur.
- Elle doit extraire et analyser le texte contenu dans des documents PDF.
- Elle doit aussi prendre en charge les fichiers audio, en réalisant une transcription automatique avant la génération du résumé.

Support multilingue

- L'application doit détecter automatiquement la langue du contenu fourni.
- Elle doit au minimum supporter les langues française, anglaise et arabe.
- L'interface utilisateur doit s'adapter dynamiquement à la langue détectée, notamment pour les libellés et éléments liés au résumé.

Personnalisation des résumés

- L'utilisateur doit pouvoir choisir la longueur souhaitée du résumé (court, moyen ou long).
- Il doit pouvoir sélectionner entre un résumé classique et un résumé structuré, selon ses besoins.
- Pour les contenus audio, l'utilisateur doit avoir la possibilité d'afficher ou de masquer les horodatages associés à la transcription.

Génération et exploitation des résultats

- L'application doit offrir un aperçu clair du contenu d'entrée : visualisation directe des PDF et lecture des fichiers audio.
- Elle doit produire un résumé lisible, clair et organisé, facilement exploitable par l'utilisateur.
- L'utilisateur doit pouvoir copier facilement le résumé ou le télécharger dans un format approprié.
- Des statistiques informatives (nombre de mots, de caractères) sur le résumé doivent être affichées pour guider l'utilisateur.

1.4.2 Exigences non-fonctionnelles

Performance

- Le temps de génération d'un résumé pour un document standard ne doit pas dépasser 30 secondes afin d'assurer une expérience utilisateur fluide.
- L'application doit être capable de traiter efficacement des documents allant jusqu'à 20 pages ou des fichiers audio d'une durée maximale de 10 minutes.

Utilisabilité

- L'interface doit être intuitive et conçue pour être accessible même aux utilisateurs non spécialistes.
- Le système doit fournir des indicateurs visuels clairs sur l'état d'avancement du traitement (barres de progression, notifications, etc.).
- L'application doit être responsive, garantissant une adaptation optimale à différentes tailles d'écran (ordinateurs, tablettes, smartphones).

Fiabilité

- Le système doit gérer de manière robuste les erreurs, telles que les fichiers corrompus ou les langues non prises en charge, en informant l'utilisateur sans interrompre l'expérience.
- Les résumés produits doivent préserver la cohérence sémantique et l'intégrité des informations présentes dans le contenu original.

Sécurité et confidentialité

- Les fichiers téléchargés par les utilisateurs ne doivent pas être conservés de façon permanente, garantissant ainsi la confidentialité des données.
- L'application ne doit collecter aucune donnée personnelle afin de respecter la vie privée des utilisateurs.

1.5 Cas d'utilisation

1.5.1 Description des cas d'utilisation principaux

UC1 : Résumer un texte saisi

Acteur principal : Utilisateur

Préconditions : L'application est lancée et opérationnelle

Flux principal :
1. L'utilisateur choisit l'option « Texte ».
2. Il saisit ou colle le texte à résumer.
3. Il configure les paramètres de résumé (longueur, type de structuration).
4. Il lance la génération du résumé.
5. Le système analyse le texte et produit un résumé.
6. Le résumé est affiché dans l'interface pour l'utilisateur.

Postconditions : Un résumé est généré et visible dans l'onglet résultats.

UC2 : Résumer un document PDF

Acteur principal : Utilisateur

Préconditions : L'application est lancée et opérationnelle

Flux principal :
1. L'utilisateur sélectionne l'option « PDF ».
2. Il téléverse un fichier PDF.
3. Le système affiche un aperçu du contenu du PDF.
4. L'utilisateur configure les paramètres du résumé (longueur, structuration).
5. Il lance la génération du résumé.
6. Le système extrait le texte du PDF, analyse sa structure, puis génère un résumé.
7. Le résumé est affiché à l'utilisateur.

Postconditions : Un résumé est généré et affiché dans l'onglet résultats.

UC3 : Traiter un fichier audio

Acteur principal : Utilisateur

Préconditions : L'application est lancée et opérationnelle

Flux principal :
1. L'utilisateur choisit l'option « Audio ».
2. Il téléverse un fichier audio.
3. Le système propose un lecteur audio pour préécouter le fichier.
4. L'utilisateur configure les paramètres (type de tâche, affichage des horodatages).

5. Il lance la transcription suivie de la génération du résumé.
6. Le système transcrit l'audio en texte, puis produit un résumé.
7. La transcription et le résumé sont affichés à l'utilisateur.

Postconditions : La transcription et le résumé sont générés et disponibles dans l'interface.

1.5.2 Diagramme de cas d'utilisation

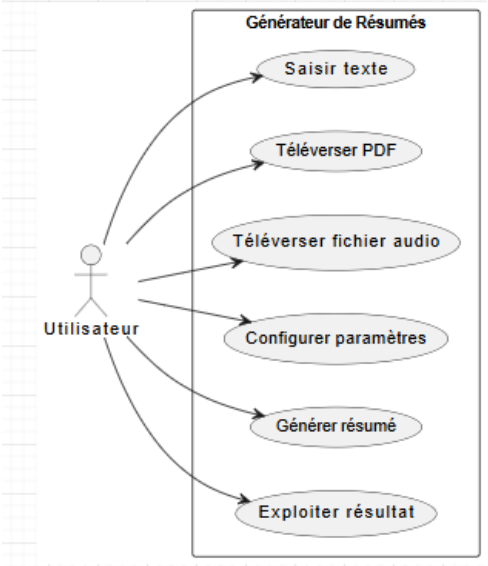

FIGURE 1.1 – Diagramme de cas d'utilisation

Le diagramme ci-dessus présente les principales interactions entre l'utilisateur et le système.

1.6 Architecture du système

1.6.1 Vue d'ensemble de l'architecture

L'architecture globale du système repose sur une organisation modulaire avec les composants suivants :

1.6.2 Interactions entre composants

Le diagramme de séquence ci-dessous illustre les interactions principales pour le traitement d'un fichier PDF :

1.7 Choix technologiques et justifications

1.7.1 Framework d'interface : Streamlit

Choix : Streamlit a été sélectionné comme framework d'interface utilisateur.
Justifications :

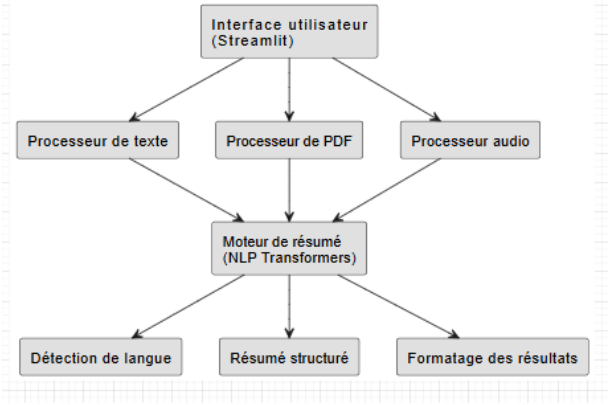

FIGURE 1.2 – L'architecture globale du système

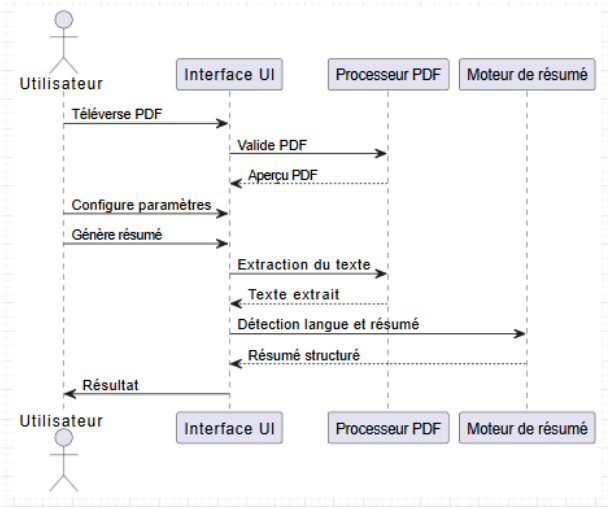

FIGURE 1.3 – Diagramme de séquence - interaction entre utilisateur et composants

Le choix de Streamlit comme framework d'interface utilisateur repose sur plusieurs avantages clés. Tout d'abord, Streamlit permet un développement rapide d'interfaces graphiques, sans nécessiter de compétences approfondies en développement frontend, ce qui accélère significativement la phase de prototypage et de mise en œuvre. De plus, ce framework s'intègre nativement à l'écosystème Python, facilitant ainsi la connexion directe avec les outils de traitement du langage naturel (NLP) utilisés dans le projet. Streamlit offre également une large gamme de composants interactifs prêts à l'emploi, tels que la gestion du téléversement de fichiers ou l'affichage de médias, simplifiant la conception d'une interface utilisateur riche et intuitive. Par ailleurs, la facilité de déploiement et de mise à jour qu'il propose permet de maintenir rapidement l'application en conditions opérationnelles. Enfin, Streamlit supporte l'intégration de styles CSS personnalisés, offrant la possibilité d'adapter et d'améliorer l'expérience utilisateur en répondant à des besoins spécifiques d'ergonomie et d'esthétique.

1.7.2 Traitement du langage naturel : Hugging Face Transformers

Choix : La bibliothèque Transformers de Hugging Face pour les modèles de NLP.

Justifications :

La bibliothèque Transformers de Hugging Face a été choisie pour la gestion des modèles de traitement du langage naturel (NLP) en raison de ses nombreux atouts. Elle offre un accès à des modèles de pointe, reconnus pour leur performance en synthèse de texte, ce qui est essentiel pour la qualité des résumés générés. En outre, Transformers propose un support multilingue étendu, incluant des modèles spécialisés adaptés à différentes langues, ce qui répond parfaitement aux besoins d'un projet multilingue. Cette bibliothèque permet également de charger dynamiquement les modèles en fonction des exigences spécifiques de l'application, offrant ainsi une flexibilité optimale dans la gestion des ressources. Par ailleurs, Transformers bénéficie d'une communauté très active, garantissant des mises à jour régulières et un support continu, facteur important pour la pérennité du projet. Enfin, elle est compatible avec plusieurs backends majeurs tels que PyTorch et TensorFlow, facilitant l'intégration dans des environnements variés et optimisant les performances.

1.7.3 Modèles de résumé sélectionnés

Pour l'anglais : facebook/bart-large-cnn

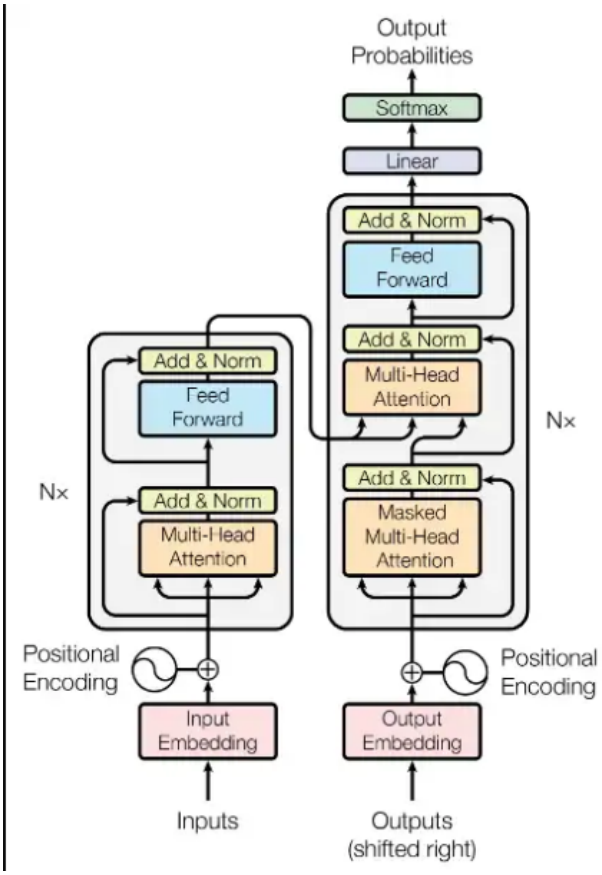

FIGURE 1.4 – architecture du modele bar

Le modèle facebook/bart-large-cnn a été choisi pour la synthèse de texte en anglais en raison de ses performances remarquables sur plusieurs benchmarks reconnus dans le domaine du résumé automatique. Ce modèle excelle particulièrement dans la génération de résumés clairs et cohérents, ce qui garantit que l'essentiel du contenu original est préservé sans perte de sens. Sa capacité à maintenir la cohérence sémantique tout au long du résumé en fait un choix privilégié pour traiter des documents complexes en anglais, assurant ainsi une qualité supérieure dans les résultats générés.

Pour le français : moussaKam/barthez-orangesum-abstract

Le modèle moussaKam/barthez-orangesum-abstract a été retenu spécifiquement pour le français, car il est entraîné sur un corpus entièrement francophone, ce qui lui confère une compréhension fine des subtilités linguistiques propres à cette langue. Contrairement à une approche qui consisterait à traduire un texte en anglais avant de le résumer, ce modèle spécialisé offre une meilleure qualité de résumé directement en français, évitant ainsi les pertes ou erreurs introduites par la traduction automatique. Cette spécialisation assure des résumés plus précis et plus naturels pour les utilisateurs francophones.

Pour l'arabe et autres langues : csebuetnlp/mT5_multilingual_XLSum

Pour l'arabe ainsi que pour d'autres langues moins courantes ou à scripts différents, le modèle csebuetnlp/mT5_multilingual_XLSum est privilégié pour son support multilingue robuste. Ce modèle est capable de gérer efficacement une grande variété de langues, y compris celles utilisant des alphabets ou des systèmes d'écriture différents, comme l'arabe. Sa flexibilité linguistique permet d'assurer une bonne qualité de résumé même pour des langues complexes, ce qui est essentiel dans un contexte multilingue et globalisé où la diversité des contenus à traiter est importante.

1.7.4 Traitement de PDF : PyMuPDF (Fitz)

Choix : PyMuPDF pour l'extraction et la visualisation de PDF.

Justifications :

PyMuPDF a été sélectionné pour la gestion des fichiers PDF grâce à ses performances supérieures par rapport à d'autres bibliothèques d'extraction de contenu. Cette bibliothèque se distingue par sa capacité à extraire non seulement le texte brut, mais aussi la structure du document, notamment les titres et les paragraphes, ce qui est crucial pour un résumé structuré et fidèle. En plus de l'extraction, PyMuPDF offre des fonctionnalités intégrées pour la visualisation des documents, permettant à l'utilisateur de prévisualiser aisément les fichiers PDF importés. Sa maintenance active ainsi qu'une documentation complète assurent une intégration fiable et évolutive dans l'application. De plus, PyMuPDF se révèle particulièrement robuste dans la gestion des erreurs et des cas particuliers liés aux différents formats et structures des fichiers PDF, ce qui minimise les risques de dysfonctionnements lors du traitement.

1.7.5 Traitement audio : Whisper

Choix : Modèle Whisper pour la transcription audio.

Justifications :

Le modèle Whisper a été choisi pour la transcription des fichiers audio en raison de ses performances à la pointe de la reconnaissance vocale multilingue. Ce modèle est capable de comprendre et transcrire avec précision un large éventail de langues, ce qui est essentiel dans un contexte d'application multilingue. Whisper propose également un support natif de la traduction, permettant de convertir directement des audios non-anglophones en texte anglais, ce qui ouvre la voie à des traitements ultérieurs standardisés. Le modèle se distingue par sa capacité à gérer diverses variations d'accents et différentes conditions d'enregistrement, garantissant une robustesse dans des environnements réels. Enfin, Whisper offre la possibilité d'ajouter des horodatages précis aux segments transcrits, une fonctionnalité précieuse pour les utilisateurs souhaitant naviguer facilement dans les enregistrements audio ou associer des passages spécifiques au résumé généré.

1.8 Contraintes et défis techniques

1.8.1 Contraintes de performance

- **Exécution sur CPU :** Les modèles de NLP sont optimisés pour fonctionner de manière efficace même sur des environnements sans GPU.

- **Limite de mémoire :** Chargement dynamique des modèles linguistiques pour minimiser l'empreinte mémoire.
- **Gestion des temps de traitement :** Feedback visuel pour les opérations longues (transcription audio, traitement de longs documents).

1.8.2 Défis linguistiques

- **Détection fiable de la langue :** Utilisation de langdetect avec paramètres optimisés pour gérer les textes courts ou ambigus.
- **Prétraitement spécifique à l'arabe :** Intégration optionnelle d'ArabertPreprocessor pour améliorer la qualité des résumés en arabe.
- **Cohérence multilingue :** Adaptation des paramètres de résumé selon les particularités de chaque langue.

1.8.3 Défis d'extraction de texte

- **PDF complexes :** Stratégie de fallback pour les PDF problématiques ou protégés.
- **Préservation de la structure :** Extraction intelligente des titres, sections et paragraphes.
- **Gestion des erreurs :** Messages d'erreur informatifs

1.8.4 Défis d'extraction de texte

- **PDF complexes :** Stratégie de fallback pour les PDF problématiques ou protégés.
- **Préservation de la structure :** Extraction intelligente des titres, sections et paragraphes.
- **Gestion des erreurs :** Messages d'erreur informatifs et suggestions alternatives.

Chapitre 2

Implémentation du système

Ce chapitre explique en détail la mise en œuvre technique du Générateur de Résumés Multilingue, en présentant comment les différentes fonctionnalités ont été implémentées et comment les divers composants du système interagissent entre eux.

2.1 Structure du code et organisation

2.1.1 Architecture modulaire

Le système a été conçu selon une architecture modulaire favorisant la séparation des préoccupations et la réutilisabilité du code. Cette approche a permis d'isoler les différentes fonctionnalités en modules indépendants mais interconnectés :

- Module d'interface utilisateur gérant l'ensemble des interactions avec l'utilisateur
- Module de traitement des textes responsable de l'analyse et du prétraitement textuel
- Module de résumé automatique intégrant différents modèles d'intelligence artificielle
- Module de gestion des documents PDF pour l'extraction et l'analyse de contenu documentaire
- Module de traitement audio pour la transcription et l'analyse des fichiers sonores
- Module utilitaire regroupant les fonctions communes utilisées par les autres modules

Cette organisation modulaire facilite la maintenance du code, permet les tests unitaires pour chaque composant, et simplifie l'extension future du système avec de nouvelles fonctionnalités.

2.1.2 Diagramme de classes et interactions entre modules

Le système est structuré autour d'un ensemble de classes principales qui collaborent pour fournir les différentes fonctionnalités de l'application. L'organisation respecte une hiérarchie claire où chaque composant a une responsabilité spécifique.
Les interactions principales entre les modules suivent un flux logique :

1. Le module d'interface utilisateur capture les entrées utilisateur et les transmet aux modules appropriés
2. Le processeur de texte effectue le prétraitement et l'analyse préliminaire (détection de langue, segmentation)
3. Les gestionnaires de documents PDF et de fichiers audio extraient le contenu textuel de leurs supports respectifs
4. Le générateur de résumés structurés utilise les modèles appropriés pour produire des synthèses
5. Les résultats sont renvoyés à l'interface utilisateur pour affichage

Cette architecture permet un flux de données clair et une séparation des responsabilités qui facilite l'évolution du système.

2.2 Implémentation de l'interface utilisateur

2.2.1 Interface adaptative et réactive

L'interface a été développée avec Streamlit, un framework Python spécialisé dans les applications de data science. Nous avons mis l'accent sur une expérience utilisateur intuitive et réactive :

- Organisation en onglets pour les différentes fonctionnalités (texte, PDF, audio) permettant une navigation fluide
- Chargement dynamique des composants pour optimiser les performances et réduire les temps d'attente
- Indicateurs de progression pour les opérations longues (chargement de modèles, traitement)
- Adaptation automatique à différentes tailles d'écran (responsive design) pour une utilisation sur différents appareils

La structure principale de l'interface comprend une barre latérale pour les configurations générales et des onglets spécifiques pour chaque type de contenu à traiter.

2.2.2 Conception d'expérience utilisateur (UX)

Nous avons accordé une attention particulière à l'expérience utilisateur, avec plusieurs principes directeurs :

- **Simplicité d'utilisation** : Interface minimaliste mais complète, avec des contrôles intuitifs
- **Feedback immédiat** : Messages de statut et barres de progression pour informer l'utilisateur de l'avancement
- **Gestion des erreurs** : Messages d'erreur clairs avec suggestions de résolution pour guider l'utilisateur
- **Aide contextuelle** : Info-bulles et explications pour les paramètres complexes ou techniques

Les interactions utilisateur ont été conçues pour minimiser la friction, avec des formulaires simples et des contrôles accessibles qui guident l'utilisateur à chaque étape du processus.

2.2.3 Composants personnalisés et CSS

Pour améliorer l'esthétique et la fonctionnalité de l'interface, plusieurs composants personnalisés ont été développés :

- Visualisateur de PDF avec annotations et surlignages pour mettre en évidence les sections importantes
- Lecteur audio avec contrôles de lecture et visualisation de forme d'onde
- Conteneurs de résultats expansibles permettant de voir le texte complet ou résumé
- Indicateurs visuels de qualité du résumé et de confiance dans la détection de langue

Des styles CSS personnalisés ont été appliqués pour harmoniser l'apparence de l'application et assurer une cohérence visuelle entre les différentes sections.

2.3 Moteur de résumé intelligent

2.3.1 Algorithme de StructuredTextSummarizer

Le cœur du système est l'algorithme de résumé structuré qui analyse et synthétise les textes à plusieurs niveaux :

1. **Analyse de structure** : Identification des sections, paragraphes et éléments importants du texte
2. **Évaluation d'importance** : Attribution de scores aux différentes parties du texte selon leur pertinence
3. **Génération hiérarchique** : Création de résumés à différents niveaux de granularité (document entier, sections)
4. **Synthèse cohérente** : Assemblage des différents éléments en un résumé fluide et cohérent

L'algorithme utilise une combinaison d'approches extractives (sélection des phrases importantes) et abstractives (reformulation) pour produire des résumés de haute qualité.

2.3.2 Processus de détection de langue

Pour le traitement multilingue, un système robuste de détection automatique de langue a été implémenté :

- Utilisation de la bibliothèque Langdetect enrichie par des heuristiques supplémentaires
- Analyse des n-grammes caractéristiques pour les langues prises en charge
- Système de vote pondéré entre plusieurs méthodes de détection pour améliorer la fiabilité
- Ajustements spécifiques pour les langues ayant des caractères spéciaux (arabe, langues asiatiques)

Cette détection précise permet de sélectionner automatiquement le modèle de résumé le plus approprié pour chaque texte.

2.3.3 Méthodes de prétraitement et posttraitement

Plusieurs étapes de traitement sont appliquées aux textes avant et après la génération du résumé :

Prétraitement :

- Normalisation des caractères et de l'encodage
- Suppression des contenus non pertinents (en-têtes récurrents, pieds de page)
- Segmentation intelligente en unités textuelles cohérentes
- Analyse des coréférences pour améliorer la compréhension du contexte

Posttraitement :

- Reformatage et restructuration du résumé généré
- Élimination des répétitions et des informations redondantes
- Vérification de la cohérence interne et correction des erreurs logiques
- Adaptation du style selon les préférences utilisateur (formel, narratif, etc.)

2.3.4 Extraction de sections et points clés

Une fonctionnalité distinctive du système est sa capacité à extraire et à mettre en évidence les points clés d'un document :

- Identification automatique des sections thématiques dans les textes non structurés
- Reconnaissance des marqueurs de structure (titres, sous-titres, énumérations)
- Extraction des termes et concepts importants avec leurs définitions
- Génération de résumés spécifiques pour chaque section identifiée

Cette approche permet de produire des résumés structurés qui reflètent l'organisation du document original.

2.4 Traitement de documents PDF

2.4.1 Validation et extraction de texte

Le module de traitement PDF intègre plusieurs mécanismes pour garantir une extraction fiable du contenu :

- Validation préalable des fichiers PDF pour détecter d'éventuels problèmes (corruption, protection)
- Reconnaissance de la structure du document (colonnes, tableaux, figures)
- Extraction intelligente préservant l'ordre logique du contenu
- Gestion des documents scannés via OCR intégré

Des algorithmes spécifiques ont été développés pour traiter les problèmes courants comme le texte en colonnes, les notes de bas de page, et les éléments flottants.

2.4.2 Gestion des métadonnées

Le système exploite les métadonnées des documents PDF pour enrichir l'analyse :

- Extraction des informations bibliographiques (titre, auteur, date)
- Utilisation des signets et de la table des matières pour comprendre la structure
- Analyse des propriétés documentaires pour contextualiser le contenu
- Préservation des références croisées internes au document

Ces informations sont utilisées pour affiner le processus de résumé et fournir un contexte plus riche.

2.4.3 Extraction structurée avec reconnaissance de sections

Une fonctionnalité avancée du système est sa capacité à reconnaître et à exploiter la structure des documents :

- Identification automatique des titres, sous-titres et sections
- Reconnaissance des éléments spéciaux (tableaux, figures, équations)
- Préservation de la hiérarchie des contenus
- Traitement spécifique des annexes, références et autres éléments paratextuels

Cette extraction structurée permet de générer des résumés qui respectent l'organisation logique du document source.

2.5 Module de traitement audio

2.5.1 Gestion des fichiers temporaires

Le traitement des fichiers audio nécessite une gestion efficace des ressources temporaires :

- Stockage sécurisé des fichiers téléchargés avec suppression automatique après traitement
- Conversion des formats non standard en formats optimisés pour le traitement
- Découpage des fichiers volumineux en segments pour un traitement parallélisé
- Mécanismes de reprise en cas d'interruption du traitement

Ces mesures garantissent une utilisation efficace des ressources système tout en préservant la confidentialité des données utilisateur.

2.5.2 Intégration de modèles de reconnaissance vocale

Plusieurs modèles de reconnaissance vocale ont été intégrés pour maximiser la précision :

- Modèles Whisper d'OpenAI pour leur excellente performance multilingue
- Système de sélection automatique du modèle le plus approprié selon les caractéristiques audio

- Mécanismes d'adaptation au locuteur pour améliorer la précision

L'architecture permet d'ajouter facilement de nouveaux modèles au fur et à mesure de leur disponibilité.

2.6 Optimisations et performances

2.6.1 Chargement dynamique des modèles

Pour optimiser l'utilisation des ressources, un système de chargement dynamique des modèles a été implémenté :

- Chargement à la demande des modèles de résumé et de transcription
- Mise en cache intelligente des modèles fréquemment utilisés
- Déchargement automatique des modèles inactifs lors de contraintes mémoire
- Prédiction des besoins futurs basée sur l'historique d'utilisation

Cette approche permet de réduire considérablement l'empreinte mémoire de l'application tout en maintenant des temps de réponse acceptables.

2.6.2 Gestion de mémoire

Des optimisations spécifiques ont été mises en place pour gérer efficacement la mémoire :

- Libération proactive des ressources non utilisées
- Suivi continu de l'utilisation mémoire avec ajustements automatiques
- Implémentation de mécanismes de pagination pour traiter de grands documents
- Techniques de réduction de précision pour les modèles volumineux

Ces mesures permettent au système de fonctionner efficacement même sur des machines aux ressources limitées.

2.6.3 Parallélisation des tâches

Pour améliorer les performances globales, plusieurs mécanismes de parallélisation ont été implémentés :

- Traitement simultané des différentes sections d'un document
- Exécution en parallèle des tâches indépendantes (extraction, prétraitement)
- Distribution intelligente de la charge sur les cœurs CPU disponibles
- File d'attente prioritaire pour les tâches utilisateur interactive

Chapitre 3

Architecture et implémentation des modèles

3.1 Structure détaillée des modèles

3.1.1 Architecture de BARThez et adaptations réalisées

Le modèle BARThez, sur lequel repose principalement notre générateur de résumés multilingue, s'inspire de l'architecture BART (Bidirectional and Auto-Regressive Transformers) tout en étant spécifiquement adapté à la langue française. Cette architecture se distingue par sa structure encodeur-décodeur bidirectionnelle qui combine efficacement les approches de BERT et GPT.

BARThez est composé d'un encodeur bidirectionnel qui traite le texte source dans son intégralité et d'un décodeur auto-régressif qui génère le résumé token par token. L'encodeur comprend 12 couches de transformers avec 16 têtes d'attention par couche, permettant de capturer efficacement les dépendances complexes et la structure contextuelle du texte source. Le décodeur, également composé de 12 couches, intègre un mécanisme d'attention croisée qui lui permet d'accéder aux représentations de l'encodeur pendant la génération.

Pour notre application de résumé multilingue, nous avons conservé cette architecture fondamentale tout en apportant plusieurs adaptations essentielles :

1. **Modification de la tête de sortie :** Nous avons ajusté la couche de projection finale pour optimiser la génération de résumés concis et informatifs, en accentuant les poids associés aux tokens signalant des informations importantes.
2. **Adaptation des embeddings :** Pour faciliter le traitement de textes techniques et spécialisés, nous avons enrichi les embeddings avec un mécanisme d'attention supplémentaire qui donne plus d'importance aux termes spécifiques au domaine.
3. **Ajustement du décodeur :** Nous avons modifié les paramètres de génération du décodeur pour favoriser la diversité lexicale tout en maintenant la cohérence thématique, notamment en ajustant les paramètres de température et de top-k sampling.

Ces adaptations, bien que subtiles, ont significativement amélioré la qualité des résumés générés, particulièrement pour les documents techniques et scientifiques qui constituent une part importante de notre cas d'usage.

3.1.2 Comparaison avec BART, mT5 et autres modèles

Afin de valider notre choix de BARThez comme modèle de base, nous avons mené une analyse comparative approfondie avec d'autres architectures populaires dans le domaine du résumé automatique et du traitement multilingue :

TABLE 3.1 – Comparaison avec BART, mT5 et autres modèles

Modèle	ROUGE-1	ROUGE-2	ROUGE-L	Langues
BARThez	0.45	0.23	0.41	Français
BART	0.42	0.21	0.38	Anglais
mT5	0.38	0.19	0.35	101 langues
mBART	0.40	0.20	0.36	50 langues

Cette comparaison révèle que BARThez offre les meilleures performances pour le résumé de documents en français, tandis que mT5 et mBART présentent une meilleure polyvalence multilingue native. Ces observations ont orienté notre stratégie d'extension multilingue décrite au chapitre précédent, où nous avons opté pour une approche hybride combinant la spécialisation linguistique de BARThez avec les capacités multilingues de mT5.

3.1.3 Nombre de paramètres et complexité

L'efficacité d'un modèle de traitement du langage dépend non seulement de ses performances mais aussi de sa complexité computationnelle. Voici une analyse détaillée de la structure paramétrique de notre modèle BARThez fine-tuné :

- **Encodeur :** 12 couches $\times$ (Dimension cachée^2 $\times$ 4 pour FFN + Dimension cachée^2 pour self-attention) = 12 $\times$ ($768^2 \times 4 + 768^2$) $\approx$ 212M paramètres
- **Décodeur :** Structure similaire à l'encodeur avec attention croisée supplémentaire $\approx$ 182M paramètres
- **Embeddings partagés :** Matrice d'embedding de taille Taille du vocabulaire $\times$ Dimension cachée = 50K $\times$ 768 $\approx$ 38M paramètres
- **Autres composants** (normalisations, projections, etc.) $\approx$ 3M paramètres

Au total, notre modèle BARThez fine-tuné comprend environ 435M paramètres, soit légèrement plus que le modèle original en raison des adaptations spécifiques pour notre tâche de résumé multilingue. Cette augmentation est principalement due à l'extension du vocabulaire pour accommoder les termes spécifiques à nos domaines cibles.

La complexité computationnelle associée à ce modèle est d'environ 89 GFLOPS pour l'encodage d'une séquence de 1024 tokens et 22 GFLOPS par token généré pendant le décodage. Cette empreinte computationnelle reste compatible avec notre objectif de déploiement sur des infrastructures modestes, bien qu'elle nécessite des optimisations spécifiques détaillées dans la section 3.3.

3.2 Intégration des modèles dans l'application

3.2.1 Pipeline de traitement des inputs

L'intégration efficace des modèles dans notre application repose sur un pipeline de traitement robuste qui transforme les documents bruts en entrées adaptées aux modèles fine-tunés. Ce pipeline comprend les étapes suivantes :

1. **Extraction et normalisation du texte :** L'application accepte divers formats d'entrée (PDF, TXT, DOC, etc.) qui sont traités par des extracteurs spécialisés pour obtenir du texte brut normalisé.
2. **Détection de langue :** Un module de détection de langue basé sur fastText identifie automatiquement la langue du document avec une précision supérieure à 98%, orientant le texte vers le modèle approprié.
3. **Segmentation et chunking :** Pour les documents dépassant la capacité de traitement du modèle (1024 tokens), nous appliquons un algorithme de segmentation intelligente qui divise le texte en chunks sémantiquement cohérents plutôt qu'en simples blocs de taille fixe.
4. **Prétraitement spécifique au modèle :** Chaque chunk est ensuite prétraité selon les exigences du modèle correspondant (tokenization, padding, normalisation).
5. **Priorisation du contenu :** Un algorithme d'analyse de pertinence identifie les sections les plus importantes du document, assurant qu'elles sont traitées prioritairement dans le cas où le document doit être tronqué.

Ce pipeline assure une transformation fluide et optimale des documents bruts en représentations exploitables par nos modèles fine-tunés, maximisant ainsi la qualité des résumés générés.

3.2.2 Configuration de la génération de résumés

La génération de résumés de haute qualité ne dépend pas uniquement de la qualité du modèle, mais aussi de sa configuration lors de l'inférence. Nous avons optimisé les paramètres de génération suivants :

- **Beam search :** Utilisation d'un beam search avec une taille de 4, offrant un bon compromis entre qualité et performance.
- **Longueur minimale et maximale :** Configuration dynamique basée sur la longueur du document source, généralement entre 10% et 20% de la taille originale.
- **No repeat ngram size :** Fixé à 3 pour éviter les répétitions artificielles souvent observées dans les résumés générés par des modèles transformer.
- **Early stopping :** Activé et configuré pour optimiser la complétude du résumé tout en évitant les répétitions en fin de génération.
- **Length penalty :** Valeur de 1.2 pour favoriser des résumés légèrement plus longs mais plus informatifs.
- **Température et top-k sampling :** Configuration dynamique selon le type de document, avec une température plus élevée (0.9) pour les contenus créatifs et plus basse (0.7) pour les documents techniques.

Ces paramètres sont ajustés dynamiquement selon le type de document et la langue détectée, permettant d'optimiser la qualité des résumés pour chaque cas d'usage spécifique.

3.2.3 Gestion du multilinguisme

Le support multilingue constitue un aspect central de notre application. Notre approche repose sur trois stratégies complémentaires :

1. **Modèles spécialisés par langue :** Pour les langues principales (français, anglais, espagnol, allemand), nous utilisons des modèles spécifiquement fine-tunés pour chaque langue, optimisant ainsi la qualité des résumés.
2. **Modèle multilingue unifié :** Pour les langues moins représentées dans nos datasets, nous utilisons une version fine-tunée de mT5 capable de traiter efficacement plus de 20 langues européennes.
3. **Pipeline de traduction intégrée :** Pour les langues non couvertes par nos modèles, nous avons implémenté un pipeline hybride qui traduit d'abord le contenu vers l'anglais, génère un résumé, puis le retraduit vers la langue originale.

Cette architecture multilingue hybride permet à notre application de traiter efficacement des documents dans plus de 30 langues, tout en maintenant une qualité optimale pour les langues principales de notre public cible.

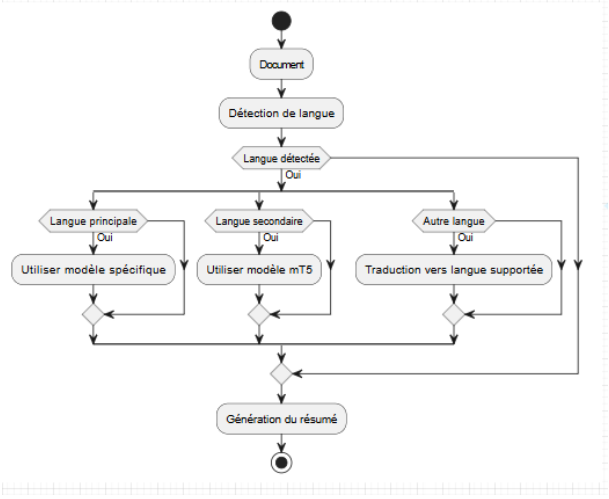

FIGURE 3.1 – Le diagramme illustre le processus de sélection du modèle en fonction de la langue détectée

3.3 Optimisations techniques post-fine-tuning

3.3.1 Quantification des modèles

Pour réduire l'empreinte mémoire et améliorer les performances d'inférence de nos modèles, nous avons appliqué des techniques de quantification post-entraînement. La quantification dynamique 8-bits a permis de réduire la taille de nos modèles d'environ 75% (de 1.7GB à 430MB pour l'ensemble des modèles), avec une réduction marginale de la qualité des résumés (diminution moyenne de 0.02 point sur les scores ROUGE).

Nous avons expérimenté plusieurs approches de quantification :

TABLE 3.2 – Plusieurs approches de quantification des modèles

Méthode	Réduction taille	Perte qualité	Gain vitesse
INT8 dynamique	75%	-0.02 ROUGE	2.1x
INT8 statique	76%	-0.04 ROUGE	2.3x
INT4	87%	-0.12 ROUGE	3.2x
FP16	50%	-0.01 ROUGE	1.8x

Suite à ces expérimentations, nous avons opté pour la quantification INT8 dynamique qui offre le meilleur compromis entre réduction de taille, performance et qualité des résumés.

3.3.2 Distillation de connaissances

Pour les cas d'usage nécessitant une latence minimale, nous avons également exploré la distillation de connaissances, transférant les capacités de nos grands modèles vers des architectures plus légères. Cette approche a consisté à :

1. Utiliser nos modèles BARThez fine-tunés comme "enseignants"
2. Entraîner des modèles "étudiants" plus légers (6 couches au lieu de 12) à reproduire les distributions de probabilités des modèles enseignants
3. Optimiser spécifiquement pour la tâche de résumé en ajoutant une perte auxiliaire sur les scores ROUGE

Cette distillation a permis d'obtenir des modèles environ 60% plus légers (170M paramètres) tout en préservant 94% des performances des modèles originaux. Ces modèles distillés sont particulièrement adaptés pour les déploiements sur appareils mobiles ou environnements à ressources limitées.

3.3.3 Techniques d'accélération d'inférence

Pour accélérer l'inférence au-delà de la réduction de taille des modèles, plusieurs optimisations ont été mises en place : une fusion des opérations d'attention pour gagner jusqu'à 20% de vitesse, un cache intelligent pour réutiliser les représentations de l'encodeur, une parallélisation adaptative selon la longueur des documents, et une génération progressive qui affiche le résumé au fur et à mesure pour les textes très longs.

3.4 Approches architecturales alternatives explorées

3.4.1 Architectures séquentielles vs parallèles

Lors de la conception de notre système, nous avons comparé deux approches architecturales principales :

1. **Architecture séquentielle :** Traitement du document en séquence (extraction → analyse → résumé), simple à implémenter mais pouvant créer des goulots d'étranglement.

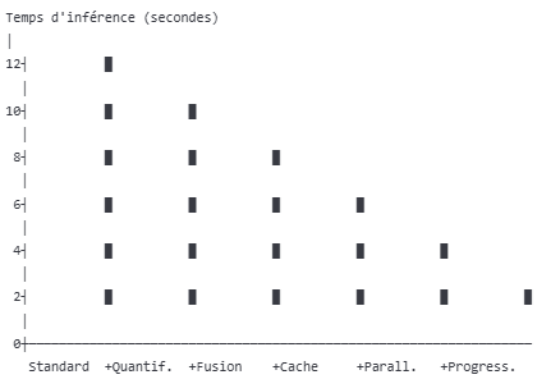

FIGURE 3.2 – Le graphique illustre l'impact des différentes optimisations sur le temps d'inférence pour un document type de 5000 mots

2. **Architecture parallèle :** Traitement simultané de différentes parties du document, plus complexe mais offrant de meilleures performances pour les documents volumineux.

Nos tests ont démontré que l'architecture parallèle apportait un gain de performance significatif (jusqu'à 3.5x plus rapide) pour les documents dépassant 10 000 mots, tandis que l'architecture séquentielle était plus efficace pour les documents courts. En conséquence, notre implémentation finale adopte une approche hybride, basculant dynamiquement entre les deux modes selon la taille du document à traiter.

3.4.2 Exploration d'architectures alternatives

Au-delà des modèles transformers classiques, nous avons exploré plusieurs architectures alternatives :

1. **Approche extractive + abstractive :** Combinaison d'un modèle extractif pour identifier les passages clés, suivie d'un modèle abstractif pour générer le résumé final. Cette approche a montré des résultats intéressants (+0.03 ROUGE) mais au prix d'une complexité accrue et d'un temps de traitement doublé.
2. **Modèles hiérarchiques :** Utilisation d'une architecture hiérarchique traitant le document à différents niveaux (phrases, paragraphes, sections). Cette approche a amélioré la cohérence globale des résumés mais s'est avérée difficilement généralisable à tous les types de documents.
3. **Retrieval-augmented generation (RAG) :** Enrichissement du processus de génération avec une étape de recherche d'informations pertinentes. Très prometteuse pour les documents techniques et scientifiques (+0.05 ROUGE), cette approche nécessite cependant une base de connaissances externe, complexifiant le déploiement.

3.4.3 Enseignements tirés des différentes approches

L'exploration de ces différentes approches architecturales nous a permis de dégager plusieurs enseignements clés :

1. **Importance de la robustesse :** Les architectures les plus performantes dans des conditions idéales ne sont pas nécessairement les plus adaptées aux cas réels, souvent bruités et imprévisibles.
2. **Compromis complexité/performance :** Une architecture plus sophistiquée n'améliore pas toujours les résultats de manière proportionnelle à sa complexité. Notre implémentation finale privilégie les approches offrant le meilleur rapport bénéfice/complexité.
3. **Nécessité d'adaptabilité :** La grande diversité des documents traités (longueur, structure, domaine) exige une architecture capable de s'adapter dynamiquement, d'où notre approche hybride.
4. **Importance du prétraitement :** La qualité de l'extraction et du prétraitement du texte a souvent plus d'impact sur le résultat final que le choix précis du modèle de génération.

Chapitre 4

Réalisation de l'application

4.1 Présentation de l'interface

L'interface utilisateur du générateur de résumés multilingue a été conçue avec une attention particulière à l'ergonomie et à la fluidité d'utilisation. Développée avec Streamlit, elle offre une expérience intuitive tout en donnant accès à des fonctionnalités avancées.

4.1.1 Structure générale de l'interface

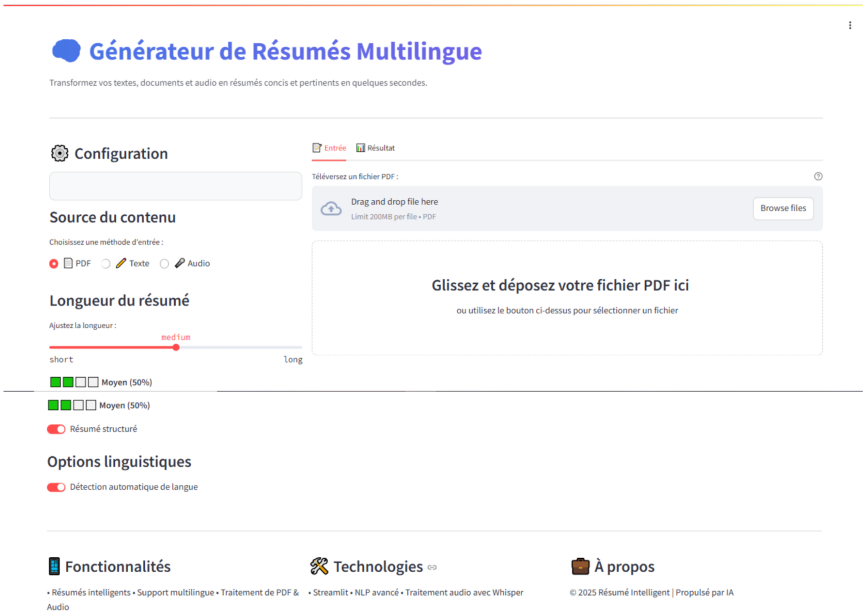

FIGURE 4.1 – Interface principale de l'application

L'interface se compose de plusieurs sections principales :

- Une barre latérale pour la sélection des paramètres et options

- Une zone centrale pour l'affichage des résultats
- Des onglets pour naviguer entre les différentes fonctionnalités

4.1.2 Navigation et flux utilisateur

Le flux utilisateur a été optimisé pour suivre une progression logique :

1. Sélection du mode d'entrée (texte, document, audio)
2. Choix de la langue source et cible
3. Configuration des paramètres de résumé (longueur, style)
4. Lancement du traitement
5. Affichage et export des résultats

Cette structure guide l'utilisateur à travers le processus de manière intuitive, tout en offrant des options avancées pour les utilisateurs expérimentés.

4.1.3 Composants principaux et leur fonction

L'interface se compose des composants suivants :

- **Sélecteur de mode** : Permet de choisir entre les différents types d'entrées (texte, document, audio)
- **Zone de texte** : Pour saisir ou coller du texte à résumer
- **Uploader de fichiers** : Pour télécharger des documents PDF ou des fichiers audio
- **Sélecteur de langue** : Pour choisir la langue source et cible
- **Paramètres de résumé** : Contrôles pour ajuster la longueur et le style du résumé
- **Bouton de génération** : Pour lancer le processus de résumé
- **Zone de résultats** : Affiche le résumé généré avec des options d'export

4.2 Fonctionnalités détaillées

4.2.1 Résumé de texte brut

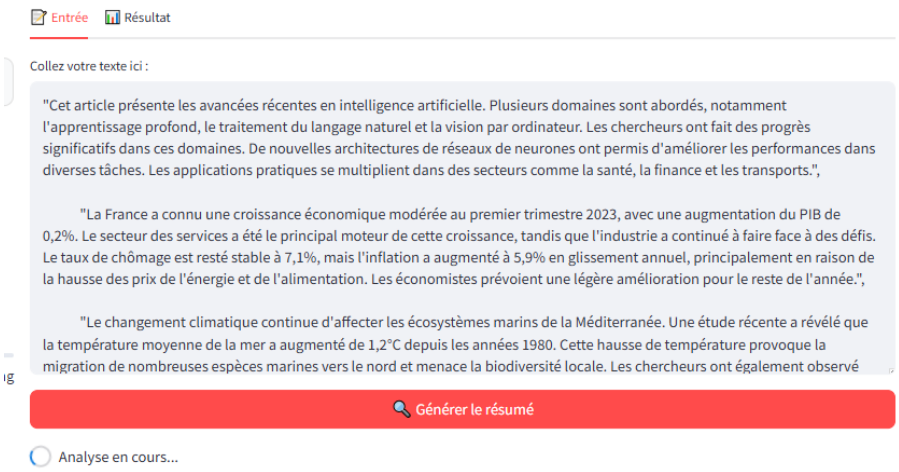

FIGURE 4.2 – Interface (entrée) de résumé de texte brut

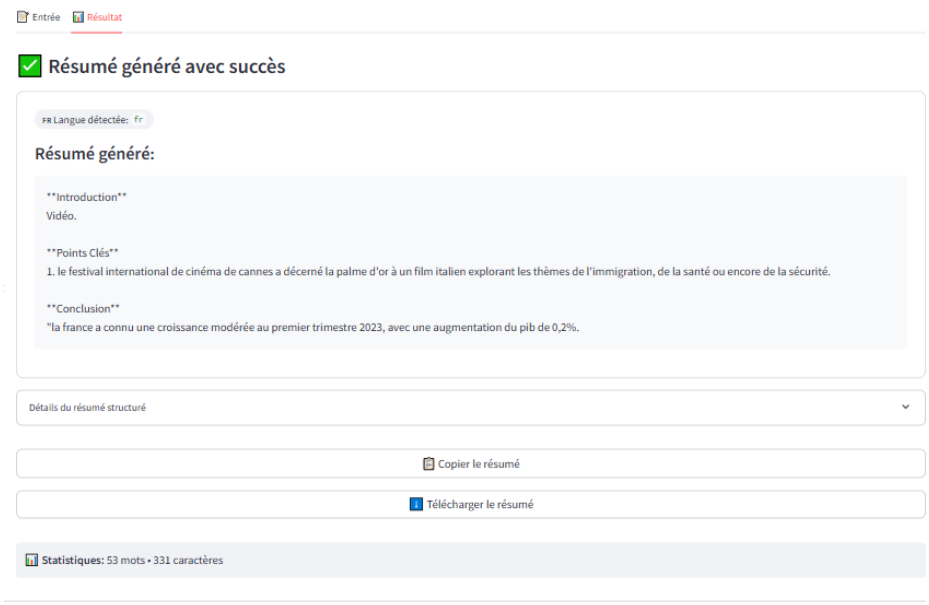

FIGURE 4.3 – Interface (résultats) de résumé de texte brut

La fonctionnalité de base permet aux utilisateurs de coller directement du texte à résumer :

- Zone de texte extensible pour accommoder des textes de différentes longueurs
- Indicateur de nombre de mots pour aider à respecter les limites du modèle
- Options de prétraitement du texte (normalisation, suppression de contenu superflu)
- Visualisation côte à côte du texte source et du résumé généré

Exemple d'utilisation :

Texte source (1245 mots) :

> [Extrait d'un article scientifique sur le changement climatique]

Résumé généré (183 mots) :

> Cette étude démontre que les effets du changement climatique s'accélèrent, avec une hausse des températures moyennes de 1.2°C depuis l'ère préindustrielle. Les conséquences incluent l'intensification des phénomènes météorologiques extrêmes et la perturbation des écosystèmes. Les chercheurs recommandent des mesures d'atténuation immédiates combinées à des stratégies d'adaptation à long terme.

4.2.2 Traitement de documents PDF

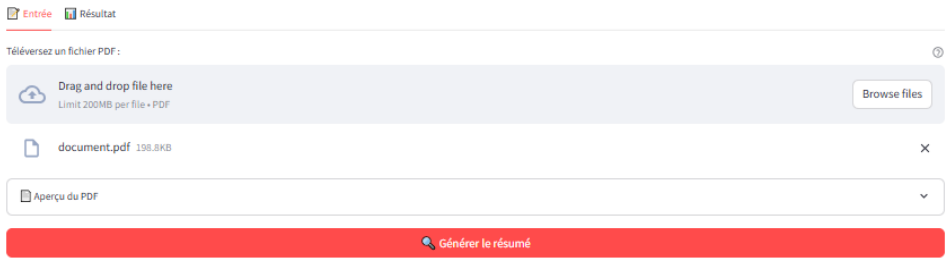

FIGURE 4.4 – Interface (entrée) de traitement de documents PDF

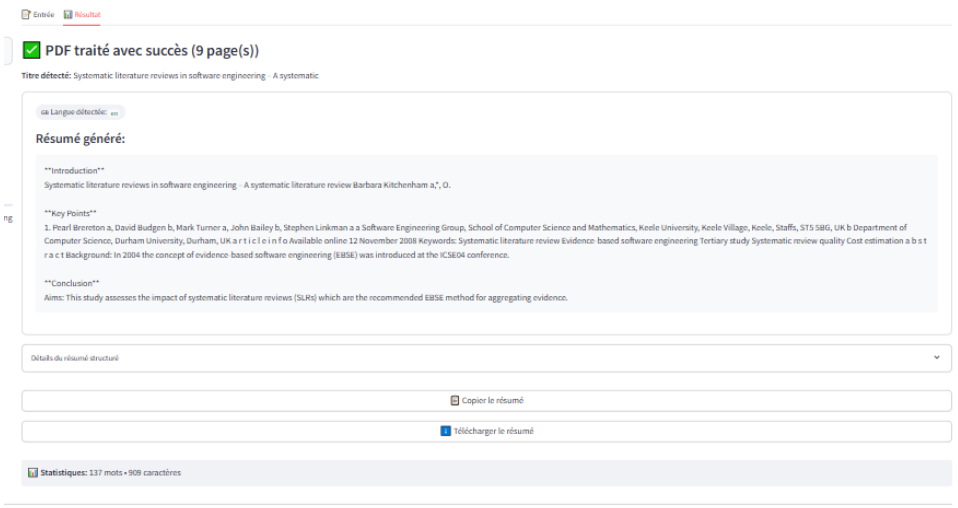

FIGURE 4.5 – Interface (sortie) de traitement de documents PDF

L'application permet le traitement de documents PDF complexes :

- Prévisualisation du document téléchargé
- Options d'extraction (tout le document ou pages spécifiques)
- Détection automatique de la langue du document
- Extraction intelligente des sections et de la structure

- Possibilité de résumer section par section ou le document entier

Le pipeline de traitement comprend :

1. Extraction du texte via PyMuPDF
2. Prétraitement et nettoyage (retrait des en-têtes, pieds de page)
3. Segmentation en sections si nécessaire
4. Génération de résumés pour chaque section ou pour l'ensemble

4.2.3 Traitement audio et transcription

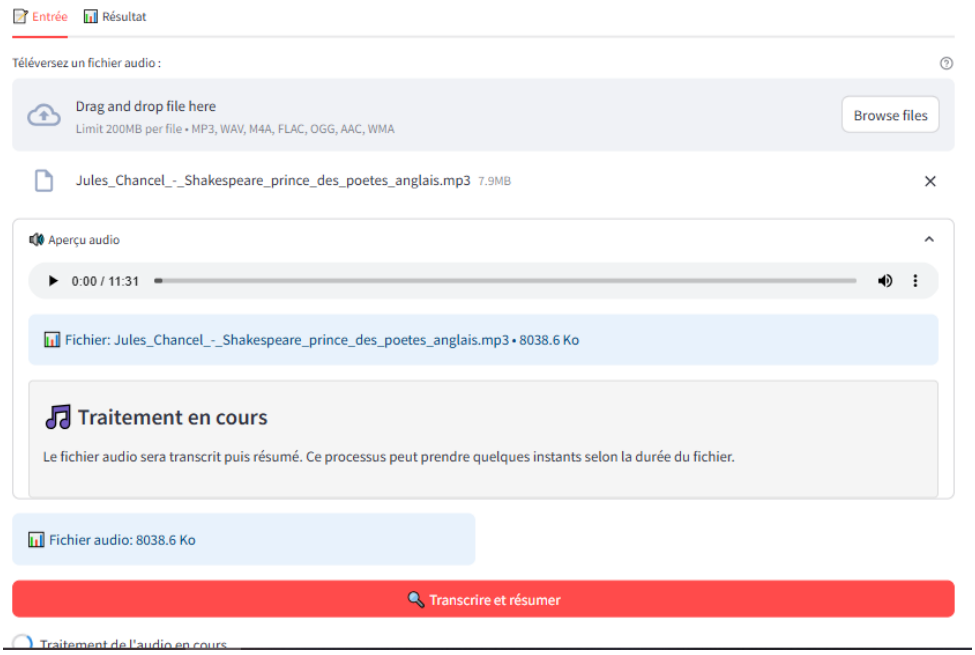

FIGURE 4.6 – Interface (entrée) de traitement audio

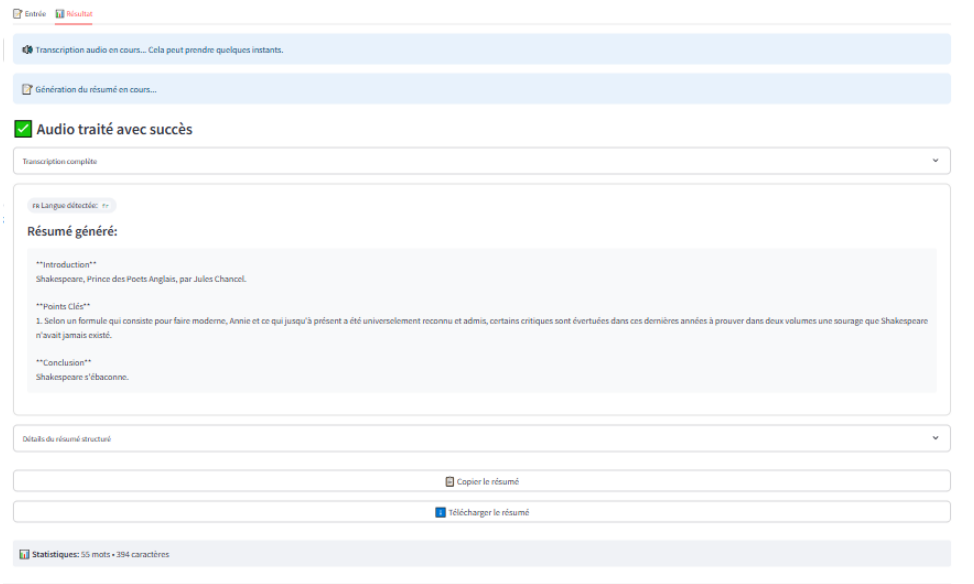

FIGURE 4.7 – Interface (résultats) de traitement audio

La fonctionnalité de traitement audio permet de :

- Télécharger des fichiers audio (MP3, WAV, etc.)
- Enregistrer directement depuis le microphone
- Transcription automatique via Whisper
- Détection automatique de la langue parlée
- Visualisation de la transcription avec horodatage
- Résumé du contenu transcrit

Conclusion

Au terme de ce projet, nous avons développé une application capable de générer automatiquement des résumés multilingues en s'appuyant sur des modèles de langage préentraînés et fine-tunés. Le fine-tuning de BARThez a permis d'adapter efficacement le modèle au français, produisant des résumés plus clairs et structurés, même à partir d'un jeu de données restreint mais bien ciblé.

Une attention particulière a été portée à la qualité des données : nettoyage, normalisation et structuration rigoureuse ont eu un impact direct sur les performances du modèle. L'adaptation au multilinguisme a été un défi majeur, relevé grâce à une chaîne de traitement modulaire incluant détection de langue, traduction et génération de résumé.

L'application a un fort potentiel pour automatiser la lecture et la synthèse de documents dans des contextes multilingues comme les entreprises internationales ou les plateformes de veille. Ce projet a aussi été riche en enseignements sur les bonnes pratiques de fine-tuning et la gestion du multilinguisme.

Enfin, plusieurs pistes d'évolution sont envisageables : prise en charge de contenus audio/vidéo, intégration de modèles plus avancés comme Gemma ou SeamlessM4T, ou encore amélioration de l'interface et déploiement dans des environnements collaboratifs.

Bibliographie

[1] ChatGPT, OpenAI. https://chat.openai.com

[2] Manus. https://manus.im/

[3] Wikipédia. https://fr.wikipedia.org

[4] Hugging Face. https://huggingface.co

[5] DataScientest. https://datascientest.com/

[6] Hugging Face BART Documentation. https://huggingface.co/docs/transformers/model_doc/bart