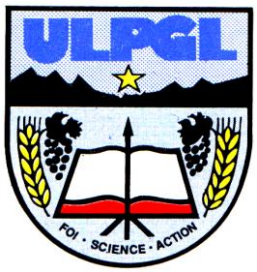
UNIVERSITE LIBRE DES PAYS DES GRANDS LACS

FACULTE DE SCIENCES ET TECHNOLOGIES APPLIQUEES

DEPARTEMENT DE GENIE ELECTRIQUE ET INFORMATIQUE



BP. 368 GOMA

www.ulpgl.net

CONCEPTION ET REALISATION D'UN GILET D'INTERPRETE DU LANGAGE NATUREL

Par : **BAHATI NGANDU Marc**

Travail présenté en vue de l'obtention du Diplôme d'ingénieur civil en génie électrique et informatique

Option : **Génie Informatique**

Directeur : **Prof BARAKA MUSHAGE Olivier**

Encadreur : **David KRAME KADURHA**

ANNEE ACADEMIQUE 2023 - 2024

Epigraphe

« Si vous parlez à un homme dans une langue qu'il comprend, vous parlez à sa tête. Si vous lui parlez dans sa langue, vous parlez à son cœur. »

Nelson Mandela

Dédicace

À ma chère mère, NDASIMWA MUNGU Sarah.

Remerciements

En premier lieu, nous exprimons notre profonde gratitude à Dieu, source infinie de lumière et de sagesse, pour la guidance précieuse qu'Il ne cesse de nous accorder. En second lieu, nous adressons nos remerciements les plus sincères à notre vénérable institution, l'Université Libre des Pays des Grands Lacs (ULPGL), qui a su nous encadrer avec bienveillance tout au long de cette étape cruciale de notre quête de connaissance. Nous tenons également à saluer nos mentors éminents: le Professeur BARAKA MUSHAGE Olivier, dont l'expertise et le dévouement ont éclairé notre cheminement académique, ainsi qu'au Master KRAME KADHURA David, dont le suivi humble, patient et attentif a enrichi notre expérience d'apprentissage. Nos pensées affectueuses vont à notre mère, NDASIMWA MUNGU Sarah, et à notre beau-frère Elie HANGI Shadrak, pour le soutien moral, matériel et affectif inestimable qu'ils nous apportent sans relâche. Nous remercions également USHINDI NGANDU Philippe pour son aide précieuse. Nous n'oublions pas ceux dont la présence contribue significativement à notre bien-être : Annah NGANDU Francine, KUBUYA NGANDU Dorcas, Frederik NGANDU et BAENI NGANDU. Enfin, nous adressons nos salutations chaleureuses à nos amis et camarades, en particulier à KAVIRA THASI Divine et HABIB LUNGA, ainsi qu'à tous les bienfaiteurs qui nous ont soutenus et que nous n'avons pas pu citer.

Résumé

Dans ce travail, nous présentons **GILAN**, un gilet d'interprétation du langage naturel, associé à une plateforme de chat en ligne dédiée à la traduction instantanée. Ce dispositif innovant permet de traduire en temps réel la voix d'un locuteur d'une langue à une autre, rendant ainsi les échanges interlinguistiques plus accessibles et compréhensibles pour tous.

L'objectif de **GILAN** est de transcender les barrières linguistiques grâce à l'utilisation de la technique de *Speech-to-Speech Translation* (S2ST), qui repose sur l'intelligence artificielle pour effectuer des traductions directes sans recourir à des ressources écrites abondantes. Contrairement aux approches traditionnelles, cette méthode est particulièrement prometteuse pour les langues moins documentées, permettant une interprétation fluide et contextuelle qui ouvre de nouvelles perspectives pour la communication interculturelle. La plateforme de chat associée vient compléter ce système en offrant une solution de traduction écrite, où chaque utilisateur peut envoyer un message dans sa propre langue, que le destinataire reçoit dans la langue de son choix, facilitant ainsi des échanges écrits adaptés aux préférences linguistiques de chacun.

Les résultats de nos recherches montrent que **GILAN** non seulement améliore la qualité des interactions linguistiques, mais également réduit les malentendus culturels, créant ainsi un environnement propice à l'harmonie et à la collaboration entre les utilisateurs. En conclusion, ce travail a pour ambition de promouvoir l'utilisation des solutions technologiques avancées pour la traduction, de valoriser l'apprentissage des langues et de soutenir l'ouverture à la diversité linguistique et culturelle, contribuant à des interactions plus riches et à une compréhension plus profonde entre les sociétés.

Mots clés: Barrières linguistiques, Intelligence artificielle, *Speech-to-Speech Translation (S2ST)*, Traduction instantanée.

Abstract

In this work, we introduce **GILAN,** a natural language interpretation vest, coupled with an online chat platform dedicated to instant translation. This innovative device enables real-time voice translation from one language to another, making interlinguistic exchanges more accessible and comprehensible for everyone.

The goal of **GILAN** is to transcend linguistic barriers through the use of Speech-to-Speech Translation (S2ST) technology, which leverages artificial intelligence to perform direct translations without relying on extensive written resources. Unlike traditional approaches, this method is particularly promising for under-documented languages, providing a fluid and contextual interpretation that opens new perspectives for intercultural communication. The accompanying chat platform complements this system by offering a written translation solution, where each user can send a message in their own language, and the recipient receives it in the language of their choice, thus facilitating written exchanges tailored to individual linguistic preferences.

Our research findings show that GILAN not only enhances the quality of linguistic interactions but also reduces cultural misunderstandings, creating a conducive environment for harmony and collaboration among users. In conclusion, this work aims to promote the use of advanced technological solutions for translation, to support language learning, and to advocate for openness to linguistic and cultural diversity, thereby contributing to richer interactions and a deeper understanding between societies.

Keywords: Language barriers, Artificial intelligence, Speech-to-Speech Translation (S2ST), Instant translation.

Table des matières

Liste des abréviations

ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
BLUE	BiLingual Evaluation Understudy
GILAN	Gilet d'interprétation du langage naturel
GRU	Gated Recurrent Unit
HuBERT	Hidden-Unit BERT
LSTM	Long Short Term Memory
MT	Machine Translation
NLP	Natural Language Processing
POS	Part Of Speech tagging
RNN	Recurrent Neural Network
S2ST	Speech-to-speech translation
S2UT	Speech-to-unit translation
Seq2Seq	Sequence-to-Sequence
TNN	Transformers neural network
WAV	Waveform Audio File Format

Liste des tableaux

Liste des figures

0. Introduction générale

0.1. Contexte

La communication est un élément fondamental dans toute interaction humaine, permettant aux individus de partager leurs pensées, idées et sentiments, tissant ainsi des liens profonds et favorisant la compréhension mutuelle. Pour relever ces défis et favoriser une communication efficace et harmonieuse, il est impératif de promouvoir l'apprentissage des langues, de recourir à des outils de traduction de qualité et de cultiver une attitude d'ouverture envers la diversité linguistique et culturelle. En surmontant ces obstacles, la diversité linguistique devient une source de richesse, permettant des échanges plus enrichissants, des collaborations fructueuses et une compréhension mutuelle approfondie entre individus et sociétés à l'échelle mondiale. En transcendant les barrières linguistiques, la communication se révèle comme un vecteur essentiel d'interaction humaine, favorisant l'harmonie et la coopération à travers le monde, ouvrant ainsi la voie à des échanges et des collaborations d'une valeur inestimable [1].

Dans ce contexte, notre approche innovante repose sur la technique de traduction *direct Speech-to-Speech Translation (S2ST)*, qui ne dépend pas de ressources écrites abondantes mais utilise l'intelligence artificielle pour traduire directement la parole d'une langue à une autre. Cette méthode se révèle particulièrement prometteuse pour les langues moins documentées, en permettant une traduction instantanée et fluide qui transcende les limitations des techniques traditionnelles

0.2. Identification et formulation du problème

Comme présenté dans la section précédente, la barrière linguistique constitue un défi significatif dans plusieurs domaines. Au niveau de la communication interpersonnelle, elle engendre des difficultés lors des interactions entre individus de cultures différentes lors de

voyages, d'événements internationaux ou dans la vie quotidienne. Dans le domaine du commerce et des affaires, elle entrave la collaboration internationale, les négociations commerciales et l'expansion vers de nouveaux marchés. En éducation et en recherche, elle limite la participation à des conférences internationales, l'échange de connaissances et la collaboration en recherche. Ces défis sont d'autant plus complexes en République démocratique du Congo (RDC), un pays qui compte une multitude des langues parlées mais souvent à faible ressource en matière de traduction [2]. **Ainsi, il serait intéressant de mettre au point un système révolutionnaire qui vise à briser ces barrières en offrant une traduction instantanée et transparente**, permettant ainsi de faciliter la communication et la collaboration à l'échelle mondiale de manière fluide et efficace. L'outil que nous développerons au cours de ce travail, pour résoudre le problème exposé ici, sera nommé **GILAN (**Gilet d'interprète du langage naturel**)**.

0.3. Questions de recherche

Face à ces défis, une question se pose :

Est-il possible de mettre au point un système capable de traduire instantanément la parole d'une personne, dite dans une langue comme le swahili, le français, l'anglais ou le lingala, vers une autre langue de manière plus fluide et dans un faible délai ?

Cette question de recherche centrale se décline en plusieurs sous-questions :

- Comment des systèmes de traduction tels que celui de *Speech-to-Speech Translation (S2ST)*, qui sont basés sur l'intelligence artificielle, pourraient être utilisés pour garantir une traduction précise et fluide en temps réel, répondant ainsi à notre objectif ?
- Comment la conception d'un dispositif portable pour la traduction instantanée et l'intégration de cette fonctionnalité dans une plateforme de chat en ligne pourraient-elles améliorer la performance et l'efficacité du système ?
- Quelle serait l'architecture globale la plus adaptée pour concevoir un système de traduction du langage naturel efficace et performant ?

0.4. Formulation des hypothèses

En se basant sur les questions soulevées, il est possible d'avancer que :

- Des systèmes de traduction basés sur l'intelligence artificielle comme celui de *Speech-to-Speech Translation (S2ST)* joueraient un rôle crucial dans la recherche de solutions de traduction instantanée performantes.
- L'utilisation d'un dispositif portable pour la traduction instantanée pourrait renforcer la portée et l'accessibilité de la communication multilingue, en permettant aux individus de bénéficier de traductions rapides et précises lors de leurs déplacements. De plus, l'intégration de la traduction instantanée dans une plateforme de chat en ligne serait une fonctionnalité très utile pour faciliter la communication à grande distance entre locuteurs des différentes langues.
- L'architecture idéale pour un système de traduction performant combinerait des techniques de *Sequence-to-Sequence (Seq2Seq)* avec des règles linguistiques. Cette combinaison constituerait une approche optimale pour un système de traduction performant. Elle permettrait d'exploiter les avantages de chaque méthode pour obtenir des traductions plus précises et plus adaptées aux nuances linguistiques.

0.5. Justification du choix du sujet et motivations

Le choix du sujet de ce travail de fin d'études découle d'une conviction profonde quant à l'importance cruciale de la communication dans les interactions humaines et des défis majeurs posés par les barrières linguistiques. Cette thématique émerge naturellement de la nécessité fondamentale de favoriser la compréhension mutuelle et la collaboration à l'échelle mondiale. Notre intérêt pour ce sujet est intrinsèquement lié à notre passion pour l'innovation technologique et son potentiel à transformer la manière dont les individus interagissent malgré les différences linguistiques. En explorant la faisabilité d'un dispositif révolutionnaire tel que le Gilet d'Interprète du langage naturel (GILAN), nous cherchons à contribuer à l'avancement des connaissances en matière de communication interculturelle et à apporter des solutions concrètes aux défis pratiques rencontrés dans des domaines variés tels que le commerce,

l'éducation et les relations internationales. Sur le plan scientifique, ce projet s'inscrit dans un domaine de recherche en constante évolution, offrant des perspectives novatrices pour surmonter les obstacles linguistiques et favoriser une communication fluide et efficace à l'échelle mondiale.

En montrant comment le GILAN peut révolutionner la manière dont les individus interagissent et coopèrent malgré les différences linguistiques, cette recherche contribue à l'avancement des connaissances et à la résolution d'un problème majeur dans le domaine de la communication interculturelle surtout en République démocratique du Congo (RDC), un pays qui compte une multitude des langues parlées [2] .Ce projet revêt une importance particulière en offrant des perspectives novatrices pour promouvoir l'inclusion, la diversité et la compréhension entre les individus et les sociétés à l'échelle mondiale. En mettant en lumière les enjeux sociétaux majeurs liés aux barrières linguistiques et en proposant des solutions innovantes, ce travail de recherche vise à répondre aux besoins des décideurs, des praticiens et du grand public en matière de communication interculturelle et de coopération internationale.

0.6. Énoncé des objectifs de recherche

0.6.1. L'objectif général

L'objectif général de cette recherche est de concevoir et réaliser un système intégré comprenant un gilet d'interprète du langage naturel (GILAN) et une plateforme web de chat en ligne avec traduction automatique, afin de surmonter les barrières linguistiques et de favoriser une communication fluide et efficace à l'échelle mondiale.

0.6.2. Les objectifs spécifiques

Pour mener à bien notre projet, nous prévoyons :

- Étudier en détail les défis et les enjeux liés aux barrières linguistiques dans divers domaines tels que le commerce, l'éducation et les relations internationales, et identifier les besoins des utilisateurs potentiels.
- Définir les spécifications techniques et les fonctionnalités clés du GILAN et de la plateforme web de chat en ligne, en s'appuyant sur les avancées récentes dans les

domaines de la traduction automatique, de la reconnaissance vocale et de l'intelligence artificielle.

- Concevoir et développer un prototype fonctionnel du système intégré, en mettant l'accent sur l'ergonomie, l'efficacité, la fiabilité et l'interopérabilité des différents composants.
- Analyser les implications sociales, économiques et éthiques de l'utilisation du système, en identifiant les avantages potentiels ainsi que les défis à relever pour une adoption à grande échelle.
- Comparer les résultats obtenus avec les solutions existantes dans le domaine de la communication multilingue, afin de mettre en évidence les innovations et les apports spécifiques de cette recherche.
- Formuler des recommandations et des perspectives d'amélioration pour optimiser l'utilisation du système intégré et favoriser son déploiement à l'échelle mondiale

0.7. Méthodologie et délimitation du travail

Dans le cadre de ce travail, les méthodes d'analyse utilisées comprennent une approche documentaire où une revue de littérature est effectuée pour étudier les techniques de traduction automatique, de reconnaissance vocale et de communication multilingue. Le système prend en entrée le français et l'anglais, et génère en sortie des traductions en français, anglais, swahili, coréen, chinois et lingala. À la fin, le résultat attendu est un système intégré, performant et adapté aux besoins de communication multilingue. Cette analyse documentaire permet d'identifier les méthodes existantes et les possibles améliorations pour le système. En complément, des techniques expérimentales sont mises en place pour évaluer l'adéquation du système aux besoins des utilisateurs, en mesurant la qualité de traduction, la fluidité de la communication et la satisfaction des utilisateurs. Ce travail combine ainsi des techniques documentaires et expérimentales pour une approche holistique. Il s'agit d'une recherche de type exploratoire, car elle vise à explorer et à améliorer les solutions existantes pour développer un système intégré novateur et performant.

0.8. Subdivision du travail

Au-delà de l'introduction et de la conclusion générale, le travail se structure comme suit :

1. Au premier chapitre, **Généralités sur le Traitement Automatique du Langage Naturel,** il s'agira de l'exploration des bases théoriques essentielles du traitement automatique du langage naturel, mettant en lumière les techniques clés telles que la traduction automatique, la reconnaissance vocale, *direct Speech-to-Speech Translation (S2ST),* ainsi que les technologies existantes à ce sujet.

2. Au second chapitre, **Conception Intégrée du Système,** nous y présenterons deux sections :

    - Dans la première section, **Conception du Gilet d'Interprète du Langage Naturel** (GILAN), nous procéderons à une analyse approfondie des besoins et des spécifications techniques du GILAN, suivi de la conception détaillée de son architecture et du choix des technologies et composants adéquats.

    - Dans la seconde section, **Conception de la Plateforme Web de Chat,** cette partie se concentrera sur l'analyse des besoins spécifiques et des exigences techniques de la plateforme web, la conception de son architecture, et l'intégration du module de traduction automatique pour assurer une communication multilingue fluide.

3. Au troisième chapitre, **Réalisation et Tests**, nous procéderons à la concrétisation de la conception établie, en mettant en œuvre les éléments clés identifiés. En nous appuyant sur la conception préalablement réalisée, nous finaliserons le développement du gilet d'interprète du langage naturel et de la plateforme de chat en ligne avec système de traduction. Nous expliquerons en détail les choix d'implémentation effectués pour assurer l'interopérabilité, la précision de la traduction, et la convivialité de l'ensemble. Enfin, nous présenterons les résultats des tests effectués pour valider le bon

fonctionnement de la solution, en mettant en lumière les performances et l'efficacité de notre projet.

Chapitre 1
Généralités sur le traitement automatique du langage naturel

Dans ce chapitre, nous allons présenter une vue d'ensemble du traitement du langage naturel (NLP). Nous allons commencer par définir le NLP, tout en soulignant les défis liés à la complexité et à la flexibilité du langage naturel face au langage formel. Nous allons ensuite examiner l'évolution du NLP, en mettant l'accent sur l'importance de l'apprentissage profond (*deep learning*) pour cette discipline. Nous allons également démontrer l'importance du prétraitement des données en NLP, en passant en revue et en introduisant les étapes clés du prétraitement. Enfin, nous allons lister les différents domaines dans lesquels le NLP intervient, ce qui nous permettra de nous rapprocher progressivement de l'objectif principal de notre travail.

1.1 Présentation et définitions

Le traitement du langage naturel ou *Natural Language Processing* (NLP) en anglais est une branche de l'intelligence artificielle qui permet aux programmes informatiques de comprendre et d'interagir avec le langage humain [3]. En tant que composante de l'IA (intelligence artificielle), le NLP traite le langage naturel, qu'il soit parlé ou écrit, en utilisant des techniques de traitement de l'information pour analyser et interpréter le sens des phrases et des textes. Dans le monde réel, le NLP a des nombreuses applications dans divers domaines, tels que la reconnaissance vocale, la traduction automatique, la classification des textes, la génération des textes, et bien d'autres [4].

Comme nous l'avons dit précédemment, grâce au *Natural Language Processing* (NLP), les appareils informatiques peuvent comprendre et traiter le langage humain. Le traitement du langage naturel se porte au centre de l'interaction entre un humain et un ordinateur. En d'autres termes c'est le point d'intersection entre l'informatique, l'intelligence artificielle et la

linguistique [3]. Cependant, il est légitime de se demander pourquoi on parle de traitement automatique du **langage naturel** (voire ce qui différencie un langage naturel des autres langages)

Le langage naturel est le terme précis pour désigner le langage humain, dont l'étude est la linguistique. Cette discipline se concentre sur l'analyse et la compréhension des règles, des structures et des usages du langage parlé et écrit par les humains. Elle couvre une variété de domaines, tels que la phonologie, la morphologie, la syntaxe, la sémantique et la pragmatique. Par opposition au langage naturel, on parle de langage formel pour désigner les systèmes d'écritures bien définis, comme les écritures mathématiques ou le code informatique. Ce type de langage est caractérisé par une structure rigide et systématique, contrairement au langage naturel qui est plus flexible et souple [4].

Les langages formels sont souvent utilisés dans les domaines des sciences exactes, de l'informatique et de l'ingénierie pour communiquer des informations précises. Concernant les langues humaines usuellement utilisées, on ne peut pas dire, sans être démenti, qu'elles sont dénuées d'imprécisions. Elles regorgent en général une grande richesse, ce qui a pour conséquence d'introduire très souvent une grande ambiguïté. Cette ambiguïté rend le langage humain complexe, car les mots et les phrases peuvent avoir plusieurs significations selon le contexte, les nuances culturelles et les intentions de l'interlocuteur. Par exemple, la phrase "Il fait chaud ici" peut simplement signifier que la température ambiante est élevée, mais selon le contexte et l'intonation, elle peut aussi être une plainte déguisée ou une invitation implicite à ouvrir une fenêtre. La capacité à interpréter correctement ces multiples facettes du langage humain repose sur une compréhension approfondie des conventions sociales, des tournures de phrases particulières et des significations cachées basées sur le contexte [5].

1.2 Nécessité du NLP par le deep learning

Le domaine du Traitement Automatique du Langage Naturel (NLP) a connu une évolution significative avant l'avènement *du deep learning*. Avant cette ère, les approches traditionnelles en NLP reposaient principalement sur des techniques de traitement symbolique et statistique,

comme la détection des spams, l'analyse des sentiments et le POS (*Part Of Speech tagging*) [6]. Ces méthodes incluaient des règles linguistiques, ainsi que d'autres méthodes basées sur des caractéristiques manuellement extraites pour traiter le langage. Cependant, ces approches étaient souvent limitées par leur capacité à capturer la complexité et la variabilité du langage naturel de manière exhaustive. Elles nécessitaient souvent une ingénierie de fonctionnalités intensive et n'étaient pas toujours capables de généraliser efficacement sur des nouvelles tâches ou des nouveaux types des données.

L'essor du *deep learning* a radicalement transformé le domaine du NLP, notamment en surmontant les limitations des approches traditionnelles [6]. L'utilisation de réseaux de neurones profonds a permis de surpasser ces limitations en offrant une plus grande capacité de modélisation des relations complexes au sein des données linguistiques. Contrairement aux approches traditionnelles, *le deep learning* permet aux modèles d'apprendre directement à partir des données brutes, sans nécessiter une extraction manuelle de caractéristiques. Grâce à leur architecture hiérarchique, les réseaux de neurones profonds sont capables de capturer des modèles complexes et abstraits dans les données, ce qui est particulièrement adapté pour la modélisation des structures linguistiques et sémantiques. Par exemple, les modèles tels que les *transformers* comme BERT et GPT ont montré des performances remarquables sur une variété de tâches NLP, telles que la traduction automatique, la génération de texte et l'analyse de sentiments.

1.3 Evolution du NLP à l'ère du deep learning

Dans cette partie, nous allons présenter l'intérêt du *deep learning* dans l'évolution du traitement automatique du langage naturel (NLP). Pour y aboutir, nous allons d'abord présenter brièvement un réseau de neurones artificiels (ANN), en suivant son évolution au fil du temps, jusqu'aux réseaux de neurones récurrents (RNN) ainsi que les *transformers*, qui représentent notre principal sujet d'intérêt dans le domaine du traitement du langage naturel (NLP).

1.3.1 Réseaux de neurones artificiels (ANN)

Les réseaux de neurones artificiels sont des systèmes informatiques inspirés du fonctionnement du cerveau humain. Ils sont composés d'une couche d'entrée qui reçoit les données, d'une ou plusieurs couches cachées qui effectuent des transformations non linéaires sur ces données, et d'une couche de sortie qui produit le résultat final [6].

Les neurones artificiels, qui constituent les unités de base du réseau, sont reliés entre eux par des connexions dont les poids sont ajustés au cours d'un processus d'apprentissage sur des données d'entraînement. Ainsi, le réseau est capable d'apprendre à résoudre des problèmes complexes, tels que la reconnaissance de formes ou le traitement du langage naturel, sans avoir besoin d'être explicitement programmé. Cette architecture multicouche permet au réseau de neurones d'extraire progressivement des représentations de plus en plus abstraites et pertinentes à partir des données brutes d'entrée. La couche d'entrée reçoit les informations, les couches cachées effectuent des transformations non linéaires successives, et la couche de sortie produit le résultat final sous la forme d'une prédiction, d'une classification ou d'une décision [7].

Voici une figure qui illustre un réseau de neurones artificiel :

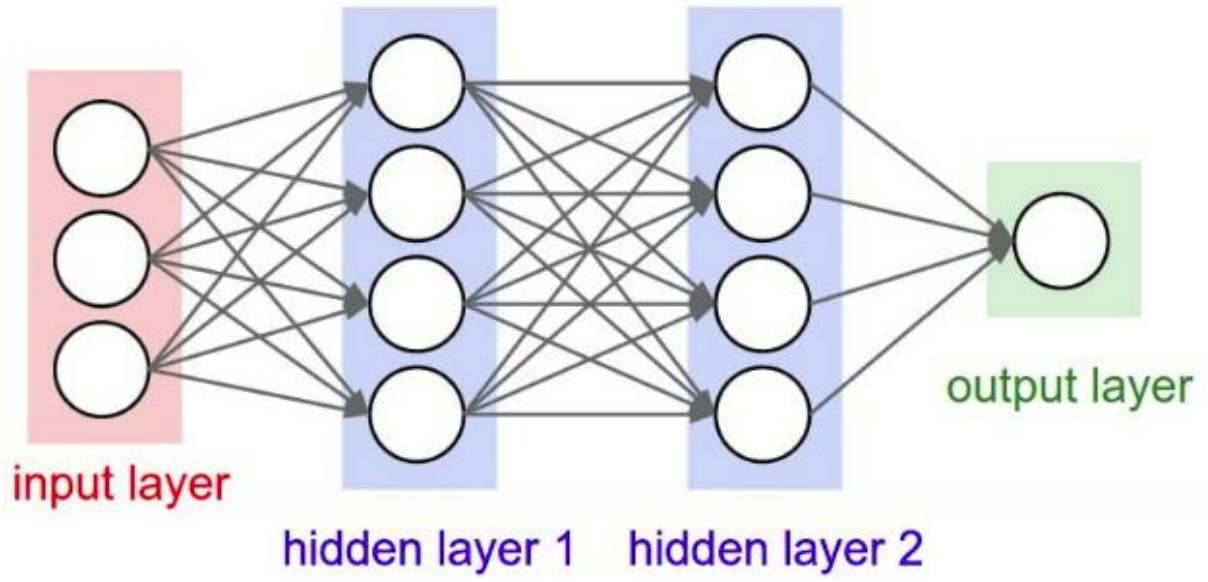


Figure 1-1 Réseau de neurones artificiel simple [7]

1.3.2 Les réseaux de neurones récurrents (RNN)

Les réseaux de neurones récurrents (RNN) sont un type de réseau de neurones particulièrement adaptés au traitement de données séquentielles, telles que le texte, l'audio, et d'autres types de données similaires. Les RNN fonctionnent en maintenant un état caché qui est mis à jour à mesure que chaque élément de la séquence est traité. Cet état caché permet au RNN de mémoriser des informations sur les éléments précédents de la séquence, ce qui peut être utile pour des tâches telles que la modélisation du langage, la traduction automatique, la reconnaissance vocale et autre [8].

L'image ci-dessous illustre la structure fonctionnelle typique des réseaux de neurones récurrents :

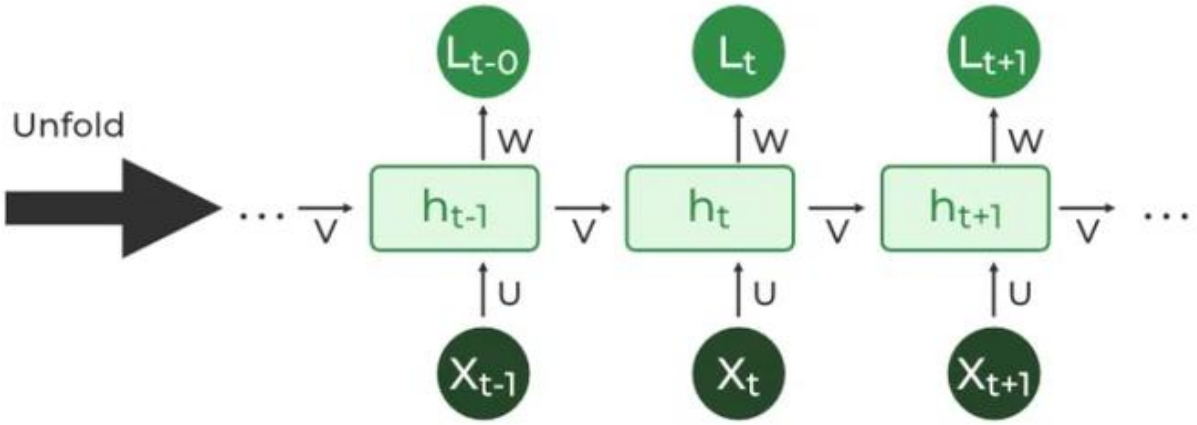


Figure 1-2 Illustration de ce qu'est un RNN [9]

L'image illustre le déroulement temporel d'un réseau de neurones récurrents (RNN). Chaque étape temporelle de la séquence est représentée par des éléments notés X_t (entrée),h_t (état caché),L_t (sortie). Le réseau traite la séquence en conservant une mémoire des états passés grâce à des connexions récurrentes.

Les poids U, V, W contrôlent les connexions dans le réseau :

- U représente les poids reliant l'entrée X_t à l'état caché h_t
- W représente les poids reliant l'état caché précédent h_{t-1} à l'état caché actuel h_t
- V représente les poids reliant l'état caché h_t à la sortie L_t

Le réseau est donc capable de modéliser les dépendances temporelles, où chaque état caché h_t est calculé en fonction de l'entrée courante X_t, de l'état caché précédent h_{t-1} et des poids U,W. La sortie L_t est générée à partir de l'état caché h_t et des poids de sortie V. Cette architecture permet au RNN de capturer l'information à travers la séquence et de la propager dans le temps.

Les Réseaux de Neurones Récurrents (RNN) ont révolutionné le traitement des données séquentielles, comme le langage naturel, la parole et les séries temporelles. Cependant, ils présentent des limites importantes qui entravent leur performance dans certaines tâches [10] :

- Le problème de la disparition du gradient:

  Un des défis majeurs des RNN est le problème de la disparition du gradient. Ce phénomène survient lors de l'apprentissage du réseau, où les gradients des poids des couches profondes du RNN s'affaiblissent au fil des itérations, rendant difficile l'apprentissage des dépendances à long terme dans les séquences. Imaginez un RNN qui doit apprendre à prédire le mot suivant dans une phrase. Pour ce faire, le réseau doit tenir compte du contexte des mots précédents. Cependant, si les gradients des poids des couches profondes s'affaiblissent, le réseau aura du mal à apprendre les relations entre les mots éloignés dans la phrase [11].

- Le problème de l'explosion du gradient:

  A l'inverse du problème de la disparition du gradient, le problème de l'explosion du gradient peut également survenir dans les RNN. Dans ce cas, les gradients des poids augmentent de manière exponentielle au fil des itérations d'apprentissage, ce qui peut conduire à une instabilité du processus d'apprentissage et à une mauvaise convergence du réseau [11].

Ces problèmes limitent la capacité des RNN à apprendre des séquences longues et complexes. Ils peuvent également entrainer un apprentissage lent et laborieux, ainsi qu'une sensibilité aux hyperparamètres du réseau. En réponse à ces deux différents problèmes citez, les cellules LSTM (*Long Short-Term Memory*) sont utilisées à la place des cellules RNN classiques.

1.3.2.1 **Les cellules LSTM**

Comme nous l'avons dit ci-haut pour surmonter les limitations des RNN, les réseaux de neurones *Long Short-Term Memory* (LSTM) ont été introduits. Les LSTM intègrent des mécanismes cellulaires qui permettent de mieux gérer les dépendances à long terme dans les séquences. Les architectures LSTM sont capables d'apprendre les dépendances à long terme

dans les données séquentielles, ce qui les rend particulièrement adaptées aux tâches telles que la traduction linguistique, la reconnaissance vocale, la prévision de séries chronologiques et autre [10]. Voici une image qui nous permettra de différencier une cellule RNN classique d'une cellule LSTM :

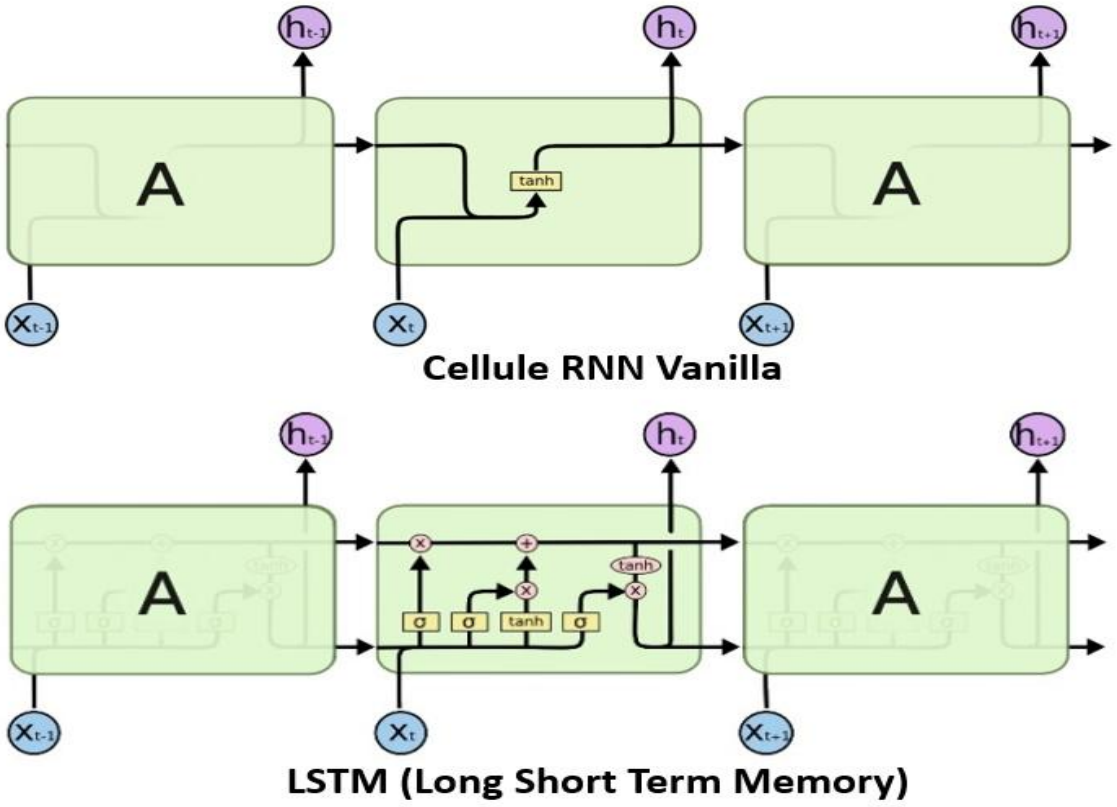


Figure 1-3 Comparaison entre cellules RNN classique et LSTM [12]

Comme nous pouvons le voir sur cette figure un RNN traditionnel possède un seul état caché qui est transmis au fil du temps, ce qui peut rendre difficile pour le réseau l'apprentissage des dépendances à long terme. Le modèle LSTM résout ce problème en introduisant une cellule mémoire, qui est un conteneur pouvant contenir des informations pendant une période prolongée. Voici une image qui illustre le fonctionnement d'une cellule LSTM :

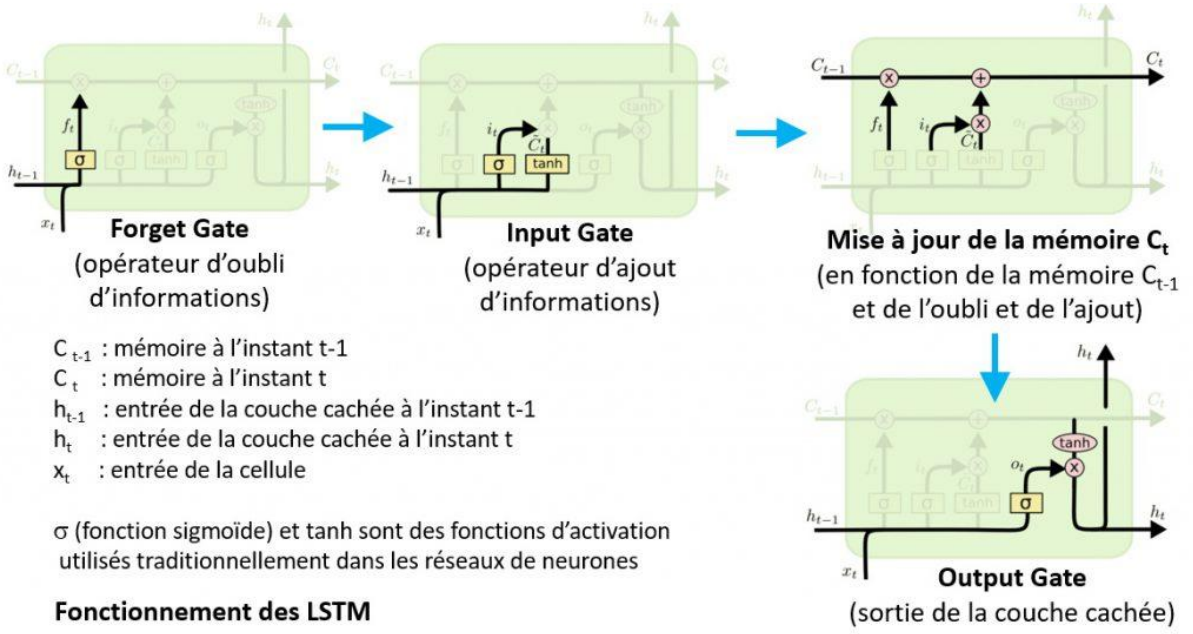


Figure 1-4 fonctionnement d'une cellule LSTM

De cette image, nous pouvons voir que les composants clés d'une cellule LSTM sont trois portes [9] : la porte d'oubli f_t qui détermine quelles informations de l'état de la cellule précédente C_{t-1} doivent être conservées, la porte d'entrée *It* qui contrôle l'ajout de nouvelles informations de l'entrée actuelle X_t à l'état de la cellule, et la porte de sortie Q_t qui détermine quelles informations de l'état de la cellule actuelle *Ct* doivent être sorties comme état caché h_t. En plus de ces portes, une cellule LSTM contient également une cellule mémoire C_t qui stocke les informations à long terme et un état caché h_t qui représente la sortie actuelle de la cellule [5]. Le flux d'informations dans une cellule LSTM se fait comme suit : la porte d'oubli détermine la proportion d'informations de l'état de la cellule précédente à conserver, la porte d'entrée contrôle l'ajout de nouvelles informations, la cellule mémoire est mise à jour en combinant ces deux éléments, et enfin la porte de sortie détermine quelles informations de l'état de la cellule actuelle doivent être sorties comme état caché. Ce mécanisme permet aux LSTM de capturer efficacement les dépendances à long terme dans les séquences de données [13]. Bien que les LSTM soient très puissants, ils peuvent être coûteux en calculs et en mémoire.

Les unités récurrentes gated (GRU) ont été proposées comme une alternative plus simple et plus efficace [13].

1.3.2.2 **Les cellules GRU**

Les GRU (Gated Recurrent Unit) combinent la porte d'oubli et la porte d'entrée en une seule porte d'actualisation, et fusionnent l'état caché et l'état de cellule en un seul vecteur d'état. Cela réduit le nombre de paramètres et de calculs tout en conservant la capacité des LSTM à capturer des dépendances à long terme. Les GRU ont montré des performances comparables aux LSTM sur de nombreuses tâches, tout en étant plus rapides et plus faciles à entraîner.

Et pour l'illustration, on peut considérer l'image suivante :

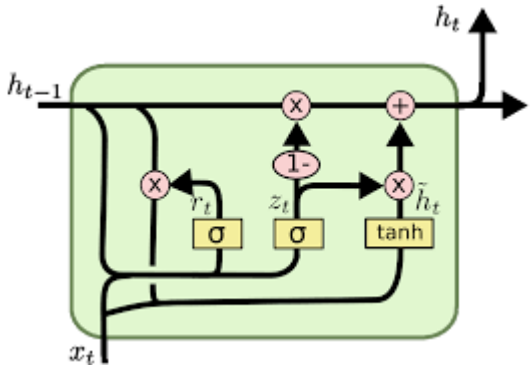


Figure 1-5 Cellule GRU [5]

Comme nous pouvons le voir l'image représente un diagramme d'un réseau neuronal récurrent à base de GRU (*Gated Recurrent Unit*). Ces différents blocs sont responsables de la mémorisation et du traitement des informations temporelles. Les flèches représentent les connexions entre les blocs. Les flèches pointées vers le haut indiquent les connexions descendantes, tandis que les flèches pointées vers le bas indiquent les connexions ascendantes. Les symboles h_t et h_{t-1} représentent les états cachés des blocs. L'état caché d'un bloc représente la mémoire de ce bloc. Le symbole x_t représente l'entrée du réseau. L'entrée peut être un mot, une phrase ou une séquence de nombres. La fonction *tanh* est utilisée pour

normaliser les sorties des blocs. Les symboles σ représentent les portes des blocs. Les portes contrôlent le flux d'informations dans les blocs [5].

Bien que les LSTM et les GRU ont été des avancées importantes dans le domaine du traitement des séquences, ces architectures présentent encore certaines limitations. En effet, les réseaux de neurones récurrents (RNN) comme le LSTM et le GRU traitent les séquences de manière séquentielle, en parcourant les éléments un par un. Cette approche peut s'avérer lente, ce qui est pourtant crucial pour de nombreuses tâches de traitement du langage naturel [14]. Une autre limitation des LSTM et GRU est leur manque de parallélisme. Comme ils traitent les éléments de la séquence de manière séquentielle, il n'est pas possible de les exécuter en parallèle sur différents éléments de la séquence. Cela contraint leur efficacité, surtout sur des séquences longues ou lors de l'entraînement sur de grands jeux de données [15].

Ces limitations ont motivé le développement d'une nouvelle architecture, les *transformers*, qui abordent ces problèmes de manière innovante. Comme nous allons le voir dans la partie qui va suivre, les *transformers* utilisent un **mécanisme d'attention** qui leur permet de traiter les éléments d'une séquence de manière parallèle et de modéliser plus efficacement les dépendances à longue distance. Cette approche a révolutionné de nombreuses tâches de traitement du langage naturel et a ouvert la voie à des progrès significatifs dans le domaine de l'intelligence artificielle appliquée au traitement du langage naturel [14].

1.3.3 Les transformers

Par définition le *transformer neural network (TNN)* ou tout simplement *Transformer* est une architecture de réseau de neurones qui utilise les mécanismes d'attention afin de résoudre les tâches de séquence à séquence tout en gérant facilement les dépendances à long terme. Autrement dit, ce modèle d'architecture ne suit pas une structure séquentielle. Ce qui facilite le traitement parallèle des données et accélère l'entraînement des modèles. C'est particulièrement intéressant pour les séquences longues en *deep learning* [16].

L'architecture de réseau de neurones *Transformer* se présente comme suit :

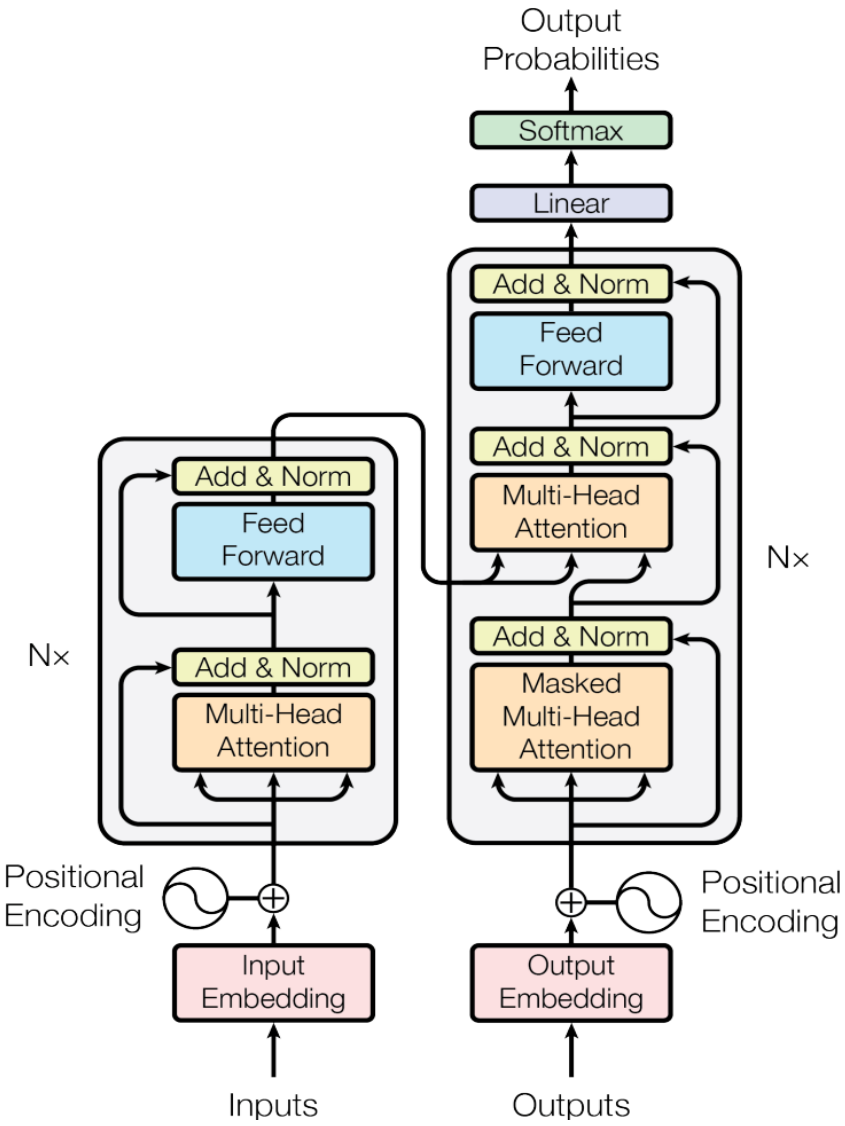


Figure 1-6 Architecture générique des transformer [16]

Comme nous pouvons le voir sur l'image, La Figure 1-6 illustre l'architecture complète du modèle *Transformer,* qui est constituée de deux composants principaux : **l'encodeur (à gauche) et le décodeur (à droite).**

- **Encodeur**

L'encodeur est formé par l'empilement de plusieurs blocs identiques, chacun comprenant deux sous-couches clés : [17] :

- *Multi-Head Attention* : Cette sous-couche permet au modèle de se concentrer simultanément sur différentes parties de la séquence d'entrée. Chaque tête d'attention attribue des pondérations distinctes, permettant au modèle d'identifier diverses relations dans la séquence.

- *Feed Forward* : Après la phase d'attention, la séquence est traitée par un réseau neuronal *feed-forward* qui applique des transformations non linéaires.

Chaque sous-couche est suivie par une opération "*Add & Norm*", qui combine une connexion résiduelle et une normalisation des couches. [17].

- **Décodeur**

Le décodeur, similaire à l'encodeur, est également composé de plusieurs blocs identiques, mais avec quelques ajustements [18] :

- *Masked Multi-Head Attention* : Cette sous-couche fonctionne de manière similaire à celle de l'encodeur, mais elle masque les positions futures dans la séquence pour empêcher le modèle de voir *les tokens* non encore générés pendant l'entraînement.

- *Multi-Head Attention* : Ici, le modèle prête attention non seulement à la séquence de sortie, mais aussi à la séquence d'entrée, utilisant ainsi les représentations fournies par l'encodeur.

- *Feed Forward* : La séquence passe ensuite à travers un réseau *feed-forward*, comme dans l'encodeur.

Ces sous-couches sont également suivies d'une opération "*Add & Norm*". [18].

- **Embeddings et Positional Encoding** [17]

À la base de l'encodeur et du décodeur se trouvent des vecteurs d'*embeddings* pour les séquences d'entrée et de sortie. Ces *embeddings* sont complétés par un encodage positionnel, qui introduit des informations sur l'ordre des *tokens*, car le modèle *Transformer* ne tient pas compte de l'ordre des éléments dans une séquence.

- **Output Layer** [18]

Au sommet du décodeur, les sorties sont transformées à l'aide d'une couche linéaire, puis d'une fonction *softmax*, afin de générer les probabilités des *tokens* de sortie.

1.4 Prétraitement des données audios

Le prétraitement pour la traduction directe de la parole à la parole avec des unités discrètes se concentre sur le traitement des données audios, contrairement au prétraitement classique basé sur le texte.

Le prétraitement dans ce contexte concerne les étapes nécessaires pour préparer les données audios avant l'entraînement d'un modèle de traduction *speech-to-speech*. Voici les principales étapes :

1.4.1 Conversion en format audio approprié

Les fichiers audios doivent être standardisés pour garantir une cohérence. Cela inclut :

- Format des fichiers : Le format *.wav* est largement utilisé pour le stockage de données audios. Il est considéré comme un format non compressé, ce qui signifie qu'il conserve la qualité audio d'origine sans perte. C'est le choix privilégié pour la plupart des systèmes de traitement de la parole, car il permet une meilleure analyse et un traitement plus efficace des données audio par les algorithmes de reconnaissance vocale. [19].

- Fréquence d'échantillonnage de 16 kHz : Pour les modèles S2UT (speech-to-unit), il est recommandé d'échantillonner à une fréquence de 16 kHz. Cela signifie que le signal audio est échantillonné 16 000 fois par seconde. Cette fréquence est suffisante pour capturer la plupart des informations pertinentes de la parole tout en réduisant la taille des fichiers audio [4].

1.4.2 Extraction d'unités discrètes

Pour les systèmes S2UT, le modèle utilise des unités discrètes extraites à partir des données audios. Voici le processus :

- Utilisation du modèle HuBERT pour extraire les unités discrètes. Ce modèle convertit les segments audios en représentations compactes discrètes en appliquant un clustering (par exemple, un modèle k-means sur les couches d'HuBERT) [19].

- Quantification : Les segments audio sont convertis en une séquence d'unités discrètes qui peuvent être traitées par des modèles de traduction. Ces unités sont généralement stockées sous forme de fichiers texte contenant les codes discrétisés.

1.4.3 Normalisation des fichiers audio [20]

Avant l'extraction des unités, il peut être nécessaire de normaliser les données audios :

- Normalisation de l'amplitude : Assurer que tous les fichiers audios aient des niveaux de volume cohérents.

- Suppression du bruit : Nettoyer les fichiers pour réduire les interférences ou les bruits de fond, ce qui peut améliorer la qualité de l'extraction des unités.

1.4.4 Préparation des données

- Alignement des fichiers source et cible : Les paires de fichiers audio source et cible doivent être correctement alignées et nommées de façon cohérente (ex :

`${SPLIT}/${SAMPLE_ID}.wav` pour chaque échantillon dans les répertoires source et cible) [19].

1.5 Quelques domaines courants du NLP

Conformément à l'objectif principal de notre projet de recherche, cette section vise à exposer les différents champs d'application du traitement automatique du langage naturel.

1.5.1 Traduction automatique

La traduction automatique est un domaine du traitement du langage naturel qui vise à traduire automatiquement un texte d'une langue source vers une langue cible. Cette technologie s'appuie sur des modèles d'apprentissage profond entraînés sur de vastes corpus de textes bilingues. Les approches modernes de traduction automatique, telles que les modèles de traduction neuronale, sont capables de produire des traductions de haute qualité, en tenant compte du contexte et de la sémantique du texte [27]. La traduction automatique trouve de nombreuses applications, notamment dans la communication multilingue, l'assistance aux voyageurs, la localisation de contenus web et la collaboration internationale. Bien que la traduction automatique ne puisse pas encore égaler la qualité d'une traduction humaine pour des textes complexes, elle offre une solution rapide et économique pour de nombreux cas d'usage [28].

1.5.2 Reconnaissance vocale

La reconnaissance vocale, également connue sous le nom de transcription automatique de la parole, est un domaine du traitement du langage naturel qui permet de convertir un signal de parole en texte. Cette technologie s'appuie sur des modèles d'apprentissage profond entraînés sur de vastes corpus d'enregistrements audio et de transcriptions textuelles. Les systèmes de reconnaissance vocale modernes sont capables de transcrire la parole avec une grande précision, en tenant compte des accents, des dialectes et du contexte. La reconnaissance vocale trouve de nombreuses applications, notamment dans les assistants vocaux, la transcription de réunions de conférences, et l'accessibilité pour les personnes malentendantes. Bien que la

reconnaissance vocale ne soit pas encore parfaite, elle offre une solution pratique et efficace pour de nombreuses tâches impliquant la conversion de la parole en texte [29].

1.5.3 Traduction directe parole-à-parole

La traduction directe parole-à-parole (S2ST, pour *Speech-to-Speech Translation*) est une avancée récente dans le domaine du traitement automatique du langage naturel, cherchant à transformer un message vocal dans une langue source en un message vocal équivalent dans une langue cible, sans recours à une étape intermédiaire de transcription textuelle. Traditionnellement, les systèmes de traduction vocale passaient par deux étapes principales : **la reconnaissance automatique de la parole** (ASR, *Automatic Speech Recognition*), suivie **d'une traduction automatique** (MT, *Machine Translation*). Cette approche a cependant des limites, notamment une perte de fluidité dans la communication et une tendance à mal interpréter certains éléments vocaux comme le ton [19].

En revanche, la traduction directe parole-à-parole, telle qu'elle est explorée dans des modèles modernes comme S2UT (*Speech-to-unit Translation*), s'affranchit de la nécessité d'une transcription. Ces modèles fonctionnent en prenant en entrée un signal audio brut et en générant directement un signal audio traduit dans la langue cible. Pour ce faire, ils exploitent des représentations audio discrètes. Par exemple, le modèle HuBERT (*Hidden-Unit BERT*), est capable d'extraire des unités discrètes à partir du flux audio. Ces unités sont ensuite utilisées pour la génération du signal dans la langue cible [20].

1.5.3.1 Pourquoi utiliser des unités discrètes ?

L'utilisation des unités discrètes, qui sont des segments audios encodés dans un format simplifié, est un élément clé de cette approche. Contrairement à la traduction basée sur du texte, les unités discrètes capturent à la fois les aspects linguistiques et acoustiques du signal vocal. Cela permet au modèle de préserver des nuances telles que le ton, le rythme, ou l'intonation, essentielles dans des langues où ces éléments modifient le sens du discours (comme en mandarin ou en thaï) [20].

Les modèles de traduction directe parole-à-parole utilisent une série de couches d'apprentissage profond, comme *les transformers*, pour analyser ces unités. Ils réalisent une sorte de correspondance entre les unités audio de la langue source et les unités correspondantes dans la langue cible. Ces modèles sont end-to-end, c'est-à-dire qu'ils prennent un signal audio en entrée et produisent directement un signal audio en sortie, sans passer par une étape intermédiaire de texte ou de transcription phonétique [19].

1.5.3.2 **Applications pratiques [19]**

La traduction directe parole-à-parole offre plusieurs avantages concrets dans le cadre d'applications en temps réel. Par exemple :

- Communication multilingue : Ce système peut faciliter les échanges en temps réel entre des interlocuteurs parlant des langues différentes, sans l'intervention d'un traducteur humain.

- Interprétation à distance : Dans des environnements où les réponses rapides sont cruciales, comme lors de conférences internationales ou de réunions diplomatiques, la traduction directe permet de fluidifier les conversations.

Accessibilité : Pour les personnes ayant des difficultés à lire ou à écrire, ce type de traduction permet un accès direct à des informations vocales dans leur langue préférée.

1.6 Conclusion partielle

Dans cette partie, nous avons présenté une vue d'ensemble du traitement du langage naturel (NLP), en expliquant ses principales applications, comme la reconnaissance vocale et la traduction automatique. Le langage naturel se distingue par sa flexibilité et sa complexité, ce qui le rend plus difficile à traiter que les langages formels, en raison des ambiguïtés et des nuances culturelles. Nous avons aussi abordé l'évolution des techniques du NLP, des approches symboliques et statistiques aux réseaux de neurones profonds, capables de capturer la complexité du langage sans nécessiter d'extraction manuelle de caractéristiques.

Nous avons mis en lumière les architectures de réseaux de neurones récurrents (RNN) et leurs variantes, ainsi que les limitations qui ont conduit à l'avènement des *transformers*, qui utilisent des mécanismes d'attention pour traiter les séquences en parallèle. Ces avancées ont permis d'améliorer considérablement les performances dans divers domaines. Nous sommes maintenant prêts à entamer la conception de notre système GILAN et de notre plateforme web de chat, en nous appuyant sur les architectures de *transformers*.

Chapitre 2

Conception Intégrée du Système

2.1 Introduction

Dans cette partie, nous entamons la conception détaillée de notre système GILAN et de la plateforme web de chat associé. En nous appuyant sur les bases théoriques et les différentes approches du traitement du langage naturel (NLP), nous explorons les choix technologiques et les architectures qui soutiendront le développement de ce système. Tout d'abord, nous analysons les besoins et les spécifications techniques essentiels pour la conception du Gilet d'Interprète du Langage Naturel (GILAN), en sélectionnant les technologies les plus appropriées pour garantir un système robuste et performant. Ensuite, nous nous concentrons sur la conception de la plateforme web de chat, en mettant l'accent sur l'intégration d'un module de traduction automatique pour faciliter une communication multilingue efficace et fluide. Ce chapitre marque ainsi le début de la réalisation concrète de notre projet, en traduisant les concepts théoriques en solutions techniques tangibles.

2.2 Conception du Gilet d'Interprète du Langage Naturel (GILAN)

2.2.1 Choix technologiques et définition

Dans cette section, nous entreprenons une exploration des choix technologiques effectués au cours de notre projet. Nous y exposons, avec rigueur, les différentes alternatives technologiques évaluées, en justifiant les raisons qui guident nos décisions. Chacune des technologies retenues fait l'objet d'une description minutieuse, mettant en lumière son architecture interne, ses mécanismes sous-jacents, ainsi que les avantages spécifiques qu'elle apporte à la réalisation de nos objectifs. Cette analyse détaillée permet de comprendre non seulement la pertinence de

chaque technologie dans notre contexte, mais aussi son intégration harmonieuse au sein de l'architecture globale du système.

2.2.1.1 Le cerveau du système

Dans le cadre du choix technologique pour le contrôle embarqué, nous avons décidé d'utiliser le Raspberry Pi, en l'intégrant dans notre architecture comme le cerveau du système. Ce choix repose sur une logique d'efficacité et de flexibilité, permettant une intégration harmonieuse au sein de notre système global. Bien que le Raspberry Pi soit un nano ordinateur, nous l'utilisons dans ce contexte pour ses capacités de traitement qui surpassent celles des microcontrôleurs traditionnels. Sa puissance de calcul, sa polyvalence, et son vaste écosystème en font une plateforme robuste pour le traitement des données en temps réel, tout en offrant l'évolutivité nécessaire pour répondre à nos exigences techniques [32].

2.2.1.2 Le Raspberry Pi

Le Raspberry Pi est un nano ordinateur monocarte développé par la Raspberry Pi Foundation. Conçu principalement pour la vulgarisation de l'informatique, il intègre un processeur ARM, de la mémoire vive, et divers périphériques (comme des ports USB, HDMI, GPIO, etc.) sur une seule carte compacte. Fonctionnant avec des systèmes d'exploitation basés sur Linux, comme Raspberry Pi OS, il offre une plateforme puissante et polyvalente pour une variété d'applications, allant de l'apprentissage de la programmation à des projets d'automatisation, en passant par des tâches de traitement de données et des applications multimédia. Sa capacité à exécuter des systèmes d'exploitation complets, son coût abordable et sa large communauté de support, font du Raspberry Pi un outil précieux pour les développeurs [33].

Figure 2-1 Raspberry pi

2.2.1.3 **Architecture du Raspberry Pi [25]**

L'architecture d'un Raspberry Pi est conçue pour être simple et flexible, tout en offrant des performances suffisantes pour une variété d'applications. Voici une vue d'ensemble de l'architecture typique d'un Raspberry Pi :

- **Processeur (CPU) :**

- **Type :** ARM Cortex-A (le modèle spécifique dépend du Raspberry Pi ; par exemple, ARM Cortex-A72 pour le Raspberry Pi 4).

- **Fonction :** Le CPU gère toutes les opérations de calcul et le traitement des données. Il est le cœur du système, exécutant les instructions des programmes.

- **Mémoire (RAM)**

- **Type :** SDRAM (le modèle et la capacité dépendent du modèle du Raspberry Pi ; par exemple, LPDDR4 pour le Raspberry Pi 4 avec jusqu'à 8 Go de RAM).

- **Fonction :** La RAM est utilisée pour stocker les données et les programmes en cours d'exécution, permettant un accès rapide par le CPU.

➢ **Stockage**

- **Type** : Carte microSD (pour le stockage principal), avec des options pour le stockage externe via USB ou disque dur.

- **Fonction :** La carte microSD est utilisée pour stocker le système d'exploitation, les applications et les fichiers de données. Sur les modèles plus récents, il est possible d'utiliser des périphériques de stockage externes.

➢ **Connectivité**

- **Ports USB :** Ports USB 2.0 ou 3.0 pour connecter des périphériques externes tels que claviers, souris, et dispositifs de stockage.

- **Ethernet/Wi-Fi** : Connexions pour réseau local et Internet (le modèle exact de la connectivité dépend du modèle du Raspberry Pi ; par exemple, Ethernet 10/100/1000 et Wi-Fi 802.11ac pour le Raspberry Pi 4).

- **Bluetooth** : Connexion sans fil pour les périphériques Bluetooth (présent dans les modèles récents comme le Raspberry Pi 4).

➢ **Affichage**

- **HDMI** : Port HDMI pour la connexion à des écrans externes, permettant l'affichage vidéo et audio.

- **Port GPIO** : Port pour les interfaces de connexion avec divers composants électroniques et capteurs.

➢ **Ports d'Entrée/Sortie (I/O)**

- **GPIO (General Purpose Input/Output)** : Broches pour connecter et contrôler divers dispositifs électroniques et capteurs.
- **Ports Audio** : Sortie audio analogique (sur certains modèles) pour connecter des haut-parleurs ou des casques.

➢ **Alimentation**

- **Source d'Alimentation** : Connexion via un port micro-USB ou USB-C (le modèle dépend du Raspberry Pi ; par exemple, USB-C pour le Raspberry Pi 4) pour fournir l'énergie nécessaire au fonctionnement.
- **Tension :** 5V
- **Intensité :** 2.5A (pour Raspberry Pi 4)

➢ **Système de Refroidissement**

- **Ventilateurs/Heatsinks** : Certains modèles peuvent nécessiter des accessoires de refroidissement pour maintenir une température de fonctionnement optimale.

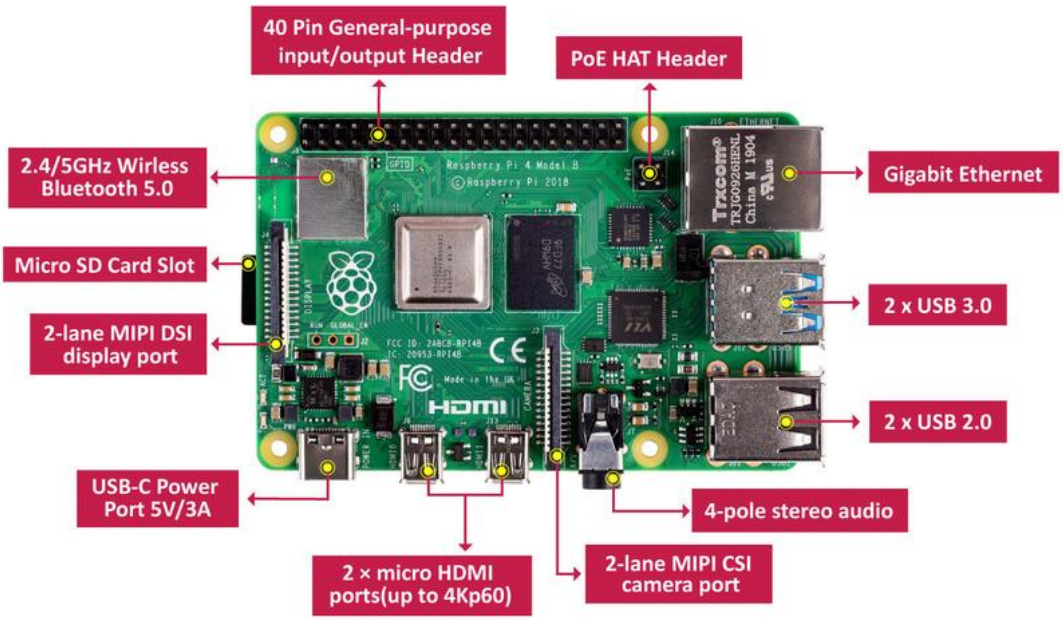


Figure 2-2 Photo detaillée du respberry pi 4

2.2.1.4 **Tableau Simplifié de l'Architecture :**

Composant	**Description**
Processeur (CPU)	ARM Cortex-A (varie selon le modèle, par exemple Cortex-A72 pour le Raspberry Pi 4)
Mémoire (RAM)	SDRAM (par exemple LPDDR4 jusqu'à 8 Go pour le Raspberry Pi 4)

Stockage	Carte microSD pour le stockage principal, avec options de stockage externe via USB ou disque dur
Connectivité	Ports USB (2.0 ou 3.0), Ethernet (10/100/1000), Wi-Fi (802.11ac), Bluetooth (selon le modèle)
Affichage	Port HDMI pour connexion à des écrans externes
Ports d'Entrée/Sortie (I/O)	GPIO (General Purpose Input/Output), ports audio (certains modèles)
Alimentation	Connexion via micro-USB ou USB-C (varie selon le modèle). Tension : 5V, Intensité : 2.5A (pour Raspberry Pi 4)
Système de Refroidissement	Ventilateurs et heatsinks (accessoires optionnels pour maintenir la température)

Tableau 2-1 Tableau Simplifié de l'Architecture

2.2.1.5 **Actionneurs pour Raspberry Pi [27]**

Écran

L'écran est un dispositif de sortie qui permet de visualiser les informations générées par le Raspberry Pi. Il peut s'agir d'un écran LCD, d'un écran OLED, ou d'un écran tactile. L'écran sera utilisé par le système pour afficher les différentes fonctionnalités de l'application, telles que la traduction des audios provenant du Raspberry Pi, permettant ainsi une visualisation claire et interactive des informations générées par le dispositif.

➢ Types d'Écrans Compatibles [32] :

- **Écran LCD (Liquid Crystal Display) :** Les écrans LCD sont couramment utilisés pour afficher des informations textuelles et graphiques. Ils peuvent se connecter via les ports GPIO ou le bus I2C.

- **Écran OLED (Organic Light Emitting Diode)** : Les écrans OLED offrent des couleurs vives et des contrastes élevés. Ils se connectent généralement via l'I2C ou le SPI.

- **Écran Tactile** : Certains écrans sont également tactiles, permettant une interaction directe avec l'interface utilisateur. Ils se connectent souvent via le port HDMI ou GPIO pour les écrans spécifiques.

Figure 2-3 Types d'écrans compatibles pour le raspberry pi

Haut-Parleurs [26]

Les haut-parleurs permettent au Raspberry Pi de produire des sons, des alertes, ou de la musique. Ils sont essentiels pour les projets nécessitant une sortie audio, comme les systèmes de reconnaissance vocale, les assistants personnels, ou les applications multimédia. Dans notre système, le haut-parleur sera utilisé pour diffuser l'audio traduit, offrant ainsi une expérience utilisateur fluide et immersive.

➢ Types de Haut-Parleurs Compatibles :

Haut-Parleurs USB :

- **Description** : Ces haut-parleurs se connectent directement au port USB du Raspberry Pi. Ils sont généralement simples à installer et ne nécessitent pas de configuration supplémentaire.

- **Caractéristiques** : Plug-and-play, souvent avec un contrôle du volume intégré.

Haut-Parleurs Jack Audio :

- **Description** : Se connectent via la prise jack 3,5 mm du Raspberry Pi. Ce type de haut-parleur est courant et facile à utiliser avec les modèles de Raspberry Pi qui ont une sortie audio analogique.

- **Caractéristiques** : Généralement compatibles avec la plupart des systèmes audio, nécessitent un câble pour la connexion.

Haut-Parleurs Bluetooth :

- **Description** : Se connectent sans fil au Raspberry Pi via Bluetooth. Ils offrent une flexibilité et évitent les câbles encombrants.

- **Caractéristiques** : Nécessitent que le Raspberry Pi soit équipé d'un module Bluetooth. La configuration peut nécessiter des étapes supplémentaires pour appairer le haut-parleur.

Haut-Parleurs USB avec Carte Son Intégrée :

- **Description :** Ces haut-parleurs incluent une carte son intégrée et se connectent au Raspberry Pi via USB. Ils offrent souvent une meilleure qualité audio que les haut-parleurs analogiques.

- **Caractéristiques** : Excellente qualité audio, compatible avec le Raspberry Pi via USB.

Haut-Parleurs Amplifiés :

- **Description** : Ces haut-parleurs ont un amplificateur intégré et se connectent via des sorties audio analogiques ou numériques. Ils sont utiles pour des projets nécessitant des volumes élevés.

- **Caractéristiques** : Haute puissance sonore, généralement connectés via des prises jack ou des sorties audio numériques.

Figure 2-4 Haut parleurs pour raspberry pi

2.2.1.6 **Capteurs pour Raspberry Pi [26]**

Microphone

Le microphone est un dispositif d'entrée qui permet au Raspberry Pi de capturer des sons et des voix. Il peut s'agir d'un microphone USB ou d'un microphone analogique avec un convertisseur analogique-numérique (ADC). Dans notre système, le microphone joue un rôle central en captant le message vocal, qui sera ensuite soumis à la traduction.

➢ Types de Microphones Compatibles :

- **Microphone USB :** Les microphones USB se connectent directement au port USB du Raspberry Pi et sont généralement plug-and-play.
- **Microphone Analogique :** Les microphones analogiques nécessitent un ADC pour convertir les signaux analogiques en données numériques que le Raspberry Pi peut traiter. Cela peut nécessiter un circuit supplémentaire ou un module ADC.

Figure 2 4 Microphone pour raspberry pi

➢ Caractéristiques Clés :

- **Sensibilité :** Détermine à quel point le microphone peut capturer les sons faibles.
- **Plage de Fréquence :** La gamme de fréquences que le microphone peut capter.
- **Connectivité :** USB ou GPIO (ADC pour les microphones analogiques).

➢ Utilisation :

- Capture de la voix pour les applications de reconnaissance vocale.
- Enregistrement de sons pour des projets de traitement audio.
- Interaction avec des systèmes de commande vocale.

2.2.2 Conception du modèle d'interprétation du langage naturel

Dans cette partie, nous allons nous concentrer sur la conception du modèle basé sur les *Transformers* pour l'interprétation du langage naturel. Nous aborderons les principes fondamentaux qui guideront le développement du modèle, notamment la structuration des données, le choix des unités discrètes, et l'optimisation des hyperparamètres pour maximiser la performance du modèle dans la traduction de séquences linguistiques.

2.2.2.1 **Approche de traduction choisie**

Nous avons opté pour une approche de traduction *direct speech-to-speech translation with discrete units*. Ce choix s'inscrit dans une démarche visant à améliorer la qualité de la traduction et à réduire la complexité computationnelle par rapport à l'utilisation traditionnelle des spectrogrammes.

Cette méthode repose sur les recherches présentées dans "*Direct speech-to-speech translation with discrete units*" par Lee et son équipe en 2021 [34]. Le système se base sur la transformation de la parole en unités discrètes, qui sont ensuite converties en parole dans la langue cible.

2.2.2.2 **Langues Choisies**

Les langues que nous avons choisies pour l'entraînement et la mise en œuvre de ce système sont :

- Français
- Anglais
- Swahili
- Chinois
- Coréen
- Lingala

Ces langues ont été sélectionnées en raison de leur diversité linguistique et de leur pertinence dans les contextes locaux et internationaux. Elles nous permettent de tester et de valider l'efficacité du modèle sur des langues appartenant à différentes familles linguistiques.

Il est important de noter que, bien que nous ayons constitué les jeux de données pour ces différentes langues, cela ne signifie pas nécessairement que nous avons collecté nous-mêmes les enregistrements audio d'origine. Cela a cependant été le cas pour certaines langues, comme le Lingala. Néanmoins, l'un des inconvénients de l'utilisation des unités discrètes dans la traduction est que le titre de l'audio source doit correspondre exactement au titre de l'audio cible correspondant. Par conséquent, nous avons dû renommer chaque fichier audio manuellement, un par un, comme si nous les avions reconstitués nous-mêmes. Cette étape laborieuse était essentielle pour assurer la précision et la cohérence des données utilisées dans notre système.

2.2.2.3 **Choix des Unités Discrètes [22]**

Les raisons qui nous ont poussés à l'utilisation des Unités Discrètes sont les suivantes :

- **Réduction de la Complexité du Modèle**:

Les unités discrètes simplifient la représentation de la parole, réduisant ainsi la complexité de la modélisation de la traduction. Cette simplification est particulièrement avantageuse lorsqu'il s'agit de manipuler de grandes quantités de données vocales provenant de plusieurs langues.

- **Amélioration de la Qualité de la Traduction:**

En utilisant des unités discrètes, il est possible de capter des aspects essentiels de la parole tout en éliminant les informations non pertinentes, ce qui peut améliorer la qualité globale de la traduction. Cela permet également de standardiser la représentation de la parole, facilitant ainsi la formation de modèles multilingues.

- **Efficacité des Ressources de Calcul:**

La transformation de la parole en unités discrètes est moins coûteuse en termes de ressources de calcul par rapport à la génération et à la manipulation de spectrogrammes, ce qui rend le modèle plus efficace.

- **Compression et Stockage**:

Les unités discrètes permettent une compression plus efficace des données vocales, réduisant ainsi les besoins en stockage par rapport aux spectrogrammes, qui sont plus volumineux.

- **Robustesse aux Variations**:

Les unités discrètes sont plus robustes aux variations de la parole telles que l'accent, le débit ou le bruit de fond, ce qui n'est pas toujours le cas pour les spectrogrammes.

- **Simplicité du Décodage:**

Le décodage des unités discrètes pour générer la parole cible est généralement plus simple et plus direct comparé au décodage des spectrogrammes, qui nécessite des étapes supplémentaires comme la conversion spectrogramme-vers-parole.

2.2.2.4 Architecture Basée sur les Transformers

- **Codage de la Parole**:

Le modèle utilise un réseau de *Transformers* pour encoder les données de la parole en séquences d'unités discrètes. Notons que le modèle basé sur les *Transformers* est adapté pour traiter des séquences longues et capturer les dépendances à longue portée dans la parole.

Pour notre système, l'encodeur du *Transformer* prend les caractéristiques de la parole en entrée (extraits par des modèles comme *HuBERT*) et les transforme en une séquence d'unités discrètes.

- **Modèle de Traduction:**

Une fois que la parole est convertie en unités discrètes, ces unités sont passées dans un autre modèle toujours basé sur les *transformers* (*s2ut_transformer_fisher* [21]) qui sert de modèle

de traduction. Ce modèle apprend à mapper les séquences d'unités discrètes de la langue source à celles de la langue cible.

Le modèle *s2ut_transformer_fisher* est entraîné de manière similaire à un modèle de traduction textuelle, où les unités discrètes jouent le rôle des "mots" dans la traduction.

- **Décodage et Synthèse de la Parole:**

Dans notre système, nous utilisons le vocodeur *HiFi-GAN* [21]. Ce vocodeur est principalement utilisé pour convertir des séquences d'unités discrètes (qui représentent la parole) en une forme de signal audio.

Ce décodage permet de capturer efficacement les informations contextuelles entre les unités discrètes et de générer une sortie vocale plus naturelle.

2.2.3 Schémas général du système

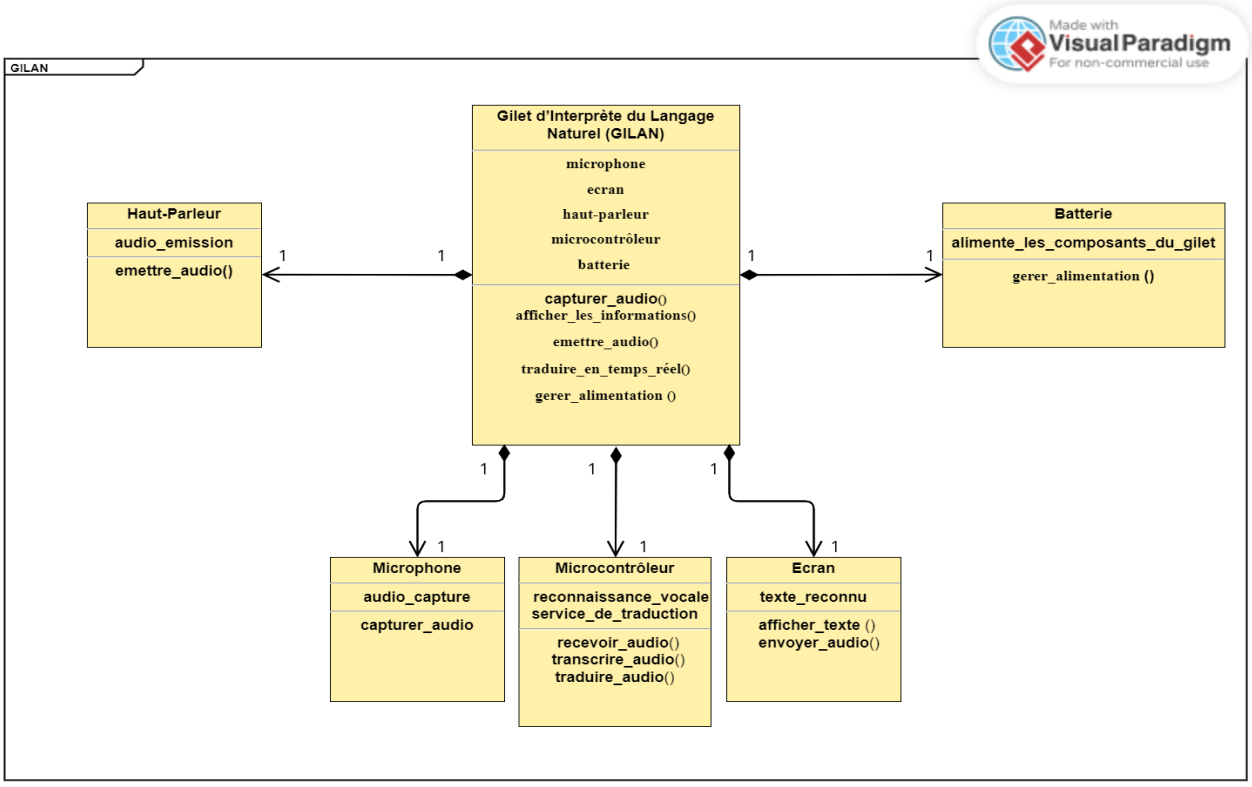


Figure 2-5 Diagramme de Bloc de Définition du système

Ce schéma identifie les composants principaux et définit leurs responsabilités.

Le microphone est responsable de la capture de l'audio, avec un attribut pour l'audio capturé et une méthode pour réaliser cette capture. L'audio capturé sera ensuite transmis au modèle de reconnaissance vocale.

Le modèle de reconnaissance vocale recevra l'audio capturé par le microphone et le convertira en texte. Ce modèle possède un attribut pour le texte transcrit ainsi que des méthodes pour recevoir l'audio, le transcrire en texte, et envoyer ce texte à l'écran pour l'affichage.

L'écran est chargé d'afficher le texte transcrit grâce au modèle de reconnaissance vocale. Il possède un attribut pour le texte reconnu ainsi qu'une méthode pour l'affichage de ce texte à l'utilisateur.

Le service de traduction, quant à lui, recevra l'audio et le traduira dans une autre langue. Il contient un attribut pour l'audio traduit et des méthodes pour recevoir l'audio, le traduire et envoyer l'audio traduit.

Enfin, **le haut-parleur** émettra l'audio traduit. Il dispose d'un attribut pour l'audio à émettre et d'une méthode pour diffuser cet audio.

2.3 Conception de la Plateforme Web de Chat

2.3.1 Choix du Modèle de Traduction

Dans le cadre de la conception de notre modèle de traduction, nous avons opté pour l'utilisation de Google Translate en raison de sa robustesse et de sa facilité d'intégration. En effet, Google Translate permet une prise en charge instantanée de plusieurs langues, garantissant une expérience utilisateur fluide et accessible à un public international. De plus, son API [29] bien documentée nous a permis de développer rapidement une solution fiable, capable de traduire des messages en temps réel. Ce choix nous offre également l'avantage de bénéficier des améliorations continues de l'algorithme de traduction de Google, ce qui assure une meilleure précision au fil du temps [29]. À la différence du modèle qui repose sur des unités discrètes évoqué précédemment, celui-ci est spécifiquement conçu pour s'intégrer dans notre application

de chat, permettant ainsi une communication multilingue fluide et harmonieuse entre les utilisateurs.

2.3.2 Conception de l'Interface

L'interface utilisateur est développée en utilisant des technologies web modernes telles que HTML, CSS, et JavaScript. Elle est conçue pour être intuitive et facile à utiliser, avec un focus sur l'expérience utilisateur.

2.3.3 Choix des technologies de développement

Nous avons opté pour Django, un framework web Python, en raison de son efficacité dans le développement rapide d'applications web et sa gestion intégrée de base des données qui est SQLite par défaut. Django est particulièrement adapté pour gérer le backend, c'est-à-dire la partie serveur de l'application.

Dans ce modèle d'architecture, l'intégration du processus de traduction est divisée en deux parties : le frontend et le backend.

Backend (Django) : Côté serveur, Django gère la logique de l'application et les requêtes API. Lorsqu'un utilisateur envoie un message, il est d'abord transmis dans sa langue d'origine au destinataire. Si le destinataire souhaite le traduire, une requête est envoyée au serveur Django, qui se charge d'appeler l'API de Google Translate pour effectuer la traduction. Le backend renvoie ensuite la version traduite du message.

Frontend : Le frontend, développé en HTML, CSS et JavaScript, est responsable de l'interface utilisateur et de l'interaction en temps réel. Lorsque le destinataire clique sur un bouton pour demander la traduction, JavaScript déclenche une requête AJAX vers le backend Django. Cela permet de récupérer le message traduit, offrant ainsi une expérience utilisateur fluide et interactive.

Cette architecture claire, avec Django gérant le backend et JavaScript gérant le frontend, assure une communication fluide entre le serveur et l'interface utilisateur, tout en optimisant les

performances globales du système. Grâce à cette séparation des responsabilités, la gestion des sessions de chat et des traductions en temps réel est à la fois efficace et évolutive.

2.3.4 Conception de la base des données

2.3.4.1 **Besoins fonctionnels de l'application**

Les besoins fonctionnels de l'application incluent :

-L'authentification des utilisateurs avant l'accès à l'application.

-L'envoi et la réception de messages privés via une interface intuitive.

-La possibilité d'envoyer et de recevoir des messages dans des groupes de discussion.

-La notification des utilisateurs lors de la réception de nouveaux messages.

-La consultation de l'historique des messages privés et de groupe.

-L'envoi de messages en temps réel, sans recharger la page.

-La possibilité de mentionner des utilisateurs dans des messages de groupe pour des notifications directes.

-La traduction des messages dans plusieurs langues, permettant à chaque utilisateur de lire dans sa langue préférée.

-Le choix de la langue souhaitée pour la visualisation des messages traduits.

2.3.4.2 **Besoins non fonctionnels de l'application**

Les besoins non fonctionnels comprennent :

-Un code clair et bien structuré pour faciliter les évolutions futures.

-Une interface conviviale et facile à utiliser, adaptée à tous les utilisateurs.

-Un système sécurisé pour protéger la confidentialité des messages.

-La capacité à supporter un grand nombre d'utilisateurs actifs simultanément sans dégradation des performances.

-Une conception visant à assurer une haute disponibilité et à minimiser les temps d'arrêt.

-Un module de traduction rapide et précis pour garantir une expérience utilisateur fluide.

2.3.4.3 **Diagramme de Classes du système**

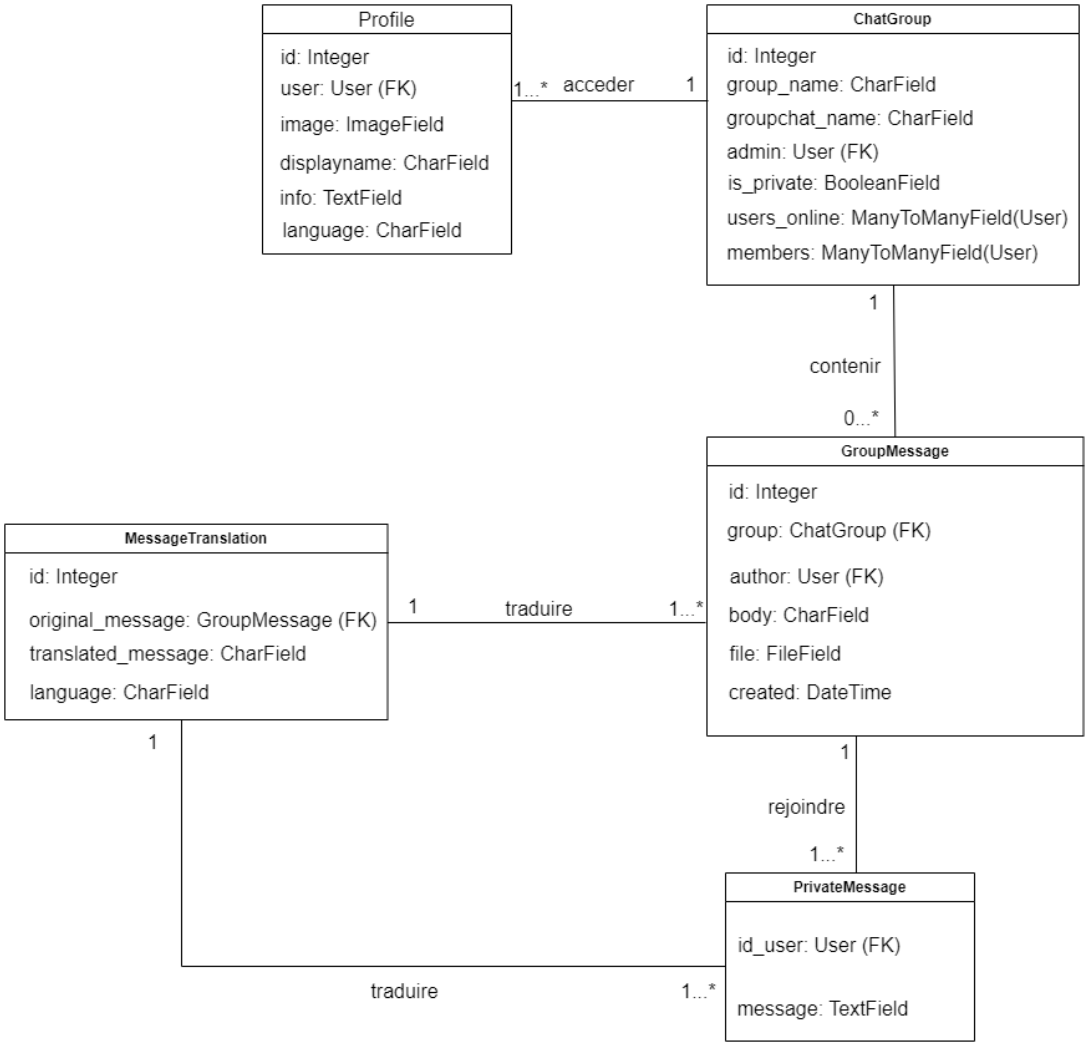


Figure 2-6 Diagramme de Classes du chat web

La conception de cette plateforme de chat repose sur plusieurs entités clés qui interagissent entre elles pour permettre une communication fluide entre les utilisateurs, ainsi que la gestion des messages et des groupes de discussion.

Profile : Dans cette classe, chaque utilisateur est associé à un profil unique. Ce profil contient des informations importantes sur l'utilisateur, telles que son identifiant unique, un champ d'image pour stocker sa photo de profil, un nom d'affichage pour l'identifier dans l'application, et des informations textuelles supplémentaires. Le profil inclut également un champ de langue, qui spécifie la langue préférée de l'utilisateur. Le profil est lié à un utilisateur à travers une clé étrangère, permettant ainsi à chaque utilisateur d'avoir un profil spécifique.

ChatGroup : La classe ChatGroup représente un groupe de discussion dans l'application. Chaque groupe a un identifiant unique et un nom de groupe pour le distinguer des autres. Il existe également un champ nom de chat pour un nom de chat spécifique si nécessaire. Un administrateur (relié à la classe utilisateur) gère le groupe. Un groupe peut être privé ou non, selon le paramètre défini dans le champ booléen correspondant. La classe enregistre également les utilisateurs en ligne et les membres du groupe, via des relations de type plusieurs-à-plusieurs avec les utilisateurs. Ces membres peuvent envoyer et recevoir des messages au sein du groupe.

GroupMessage : Cette classe est utilisée pour représenter un message envoyé dans un groupe de discussion. Chaque message a un identifiant unique et est associé à un groupe spécifique ainsi qu'à un auteur (un utilisateur). Le contenu du message est stocké dans un champ body, tandis qu'il existe aussi un champ pour attacher un fichier si nécessaire. Le message est également horodaté pour enregistrer la date et l'heure de son envoi. Un message de groupe peut également être traduit dans différentes langues via la classe de traduction de message.

PrivateMessage : Cette classe représente les messages privés échangés entre les utilisateurs. Chaque message privé est lié à un utilisateur via une clé étrangère (FK User). Cela permet aux utilisateurs d'envoyer des messages directs en dehors des groupes de discussion.

MessageTranslation : La classe MessageTranslation gère les traductions des messages de groupe. Chaque traduction est liée à un message d'origine, permettant d'identifier quel message est traduit. Le message traduit est stocké dans un champ texte, et la langue dans laquelle le message est traduit est également précisée. Cela permet à un même message de groupe d'exister

dans plusieurs langues, facilitant la communication entre utilisateurs parlant différentes langues.

Les relations entre ces entités montrent un système cohérent pour gérer la messagerie privée, les discussions de groupe, ainsi que la traduction automatique des messages. Les utilisateurs peuvent envoyer des messages, et voir les messages traduits en fonction de leur langue préférée.

2.4 Conclusion Partielle

Dans cette section, nous avons présenté les éléments essentiels de la conception du Gilet d'Interprète du Langage Naturel (GILAN) et de sa plateforme de chat. Nous avons choisi le Raspberry Pi comme cerveau du système en raison de sa puissance de calcul et de sa flexibilité, adaptés au traitement des données en temps réel. Les caractéristiques techniques du Raspberry Pi, telles que son architecture et ses options de connectivité, ont été mises en avant, tout comme les dispositifs d'entrée et de sortie nécessaires, notamment les écrans, haut-parleurs et microphones. Par ailleurs, nous avons opté pour un modèle de traitement du langage naturel basé sur les *Transformers*, qui améliore la qualité des traductions. Ce modèle est conçu pour gérer la complexité du langage, avec une diversité linguistique choisie pour tester son efficacité. Ces choix technologiques et conceptuels posent des bases solides pour la réalisation de notre projet, dont la prochaine étape sera la mise en œuvre technique des solutions pour le GILAN et la plateforme de chat.

Chapitre 3

Réalisation et Tests

3.1 Introduction

Ce chapitre couvre la réalisation et les tests du système GILAN ainsi que de la plateforme web de chat qui lui est associée. Après avoir défini les choix technologiques, nous passons à l'implémentation des solutions conçues. Nous développons d'abord le GILAN, intégrant les composants matériels et logiciels avec soin. Simultanément, nous mettons en place la plateforme de chat, en assurant le bon fonctionnement des fonctionnalités telles que la traduction automatique et la communication multilingue. Une fois la réalisation terminée, nous procédons aux tests. Ceux-ci incluent des tests unitaires et d'intégration pour vérifier chaque composant, ainsi que des tests de performance et de validation pour garantir que le système répond aux exigences et fonctionne correctement sous différentes conditions.

3.2 Présentation de l'architecture détaillée du GILAN

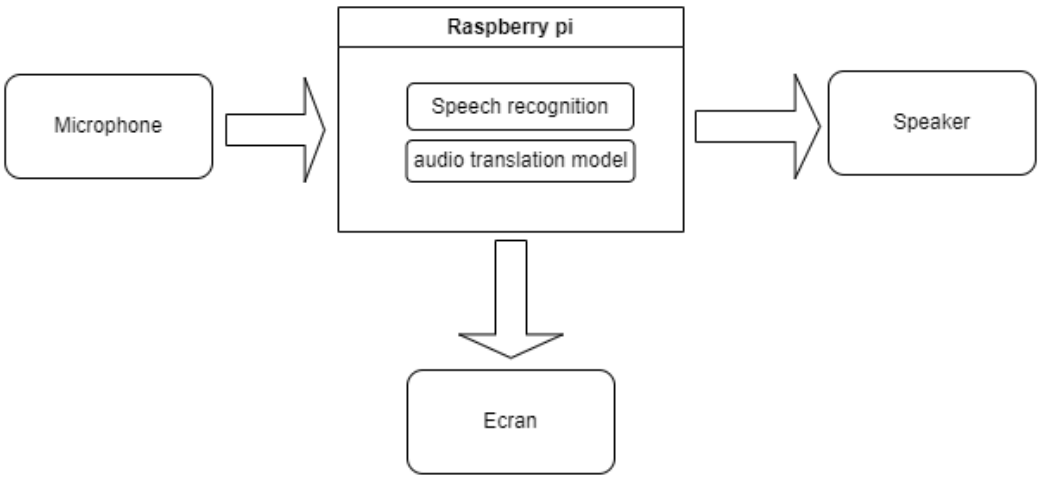


Figure 3-1 Architecture detaillée du GILAN

L'architecture détaillée de notre système repose sur l'intégration ingénieuse du Raspberry Pi, du microphone, du haut-parleur, de l'écran et de l'intelligence artificielle comme décrit dans la section précédente Figure 3-1, ou chaque composant jouent un rôle spécifique dans le fonctionnement global du système. Une fois correctement assemblés et configurés, ces éléments permettent de créer un dispositif d'interprétation intelligent, capable de traduire en temps réel et de manière fluide les conversations dans différentes langues. Cette conception structurelle nous permet de proposer une solution autonome, portable et extrêmement polyvalente. Le choix du Raspberry Pi, grâce à ses capacités d'extension et sa faible consommation énergétique, s'est naturellement imposé pour centraliser le traitement et coordonner les interactions entre les différentes parties. En effet, la puissance de calcul du Raspberry Pi, associée à ses interfaces d'entrée/sortie polyvalentes, lui confère la flexibilité

nécessaire pour traiter les flux de données audio et visuels tout en permettant une communication fluide avec les modules externes.

De plus, notre gilet se distingue par l'intégration de l'intelligence artificielle, qui joue un rôle fondamental dans la gestion des données de traduction. L'intelligence artificielle nous offre non seulement la possibilité de traduire la parole d'une langue à une autre, mais elle permet également de reconnaître les nuances de ton, de capturer les émotions dans la voix, et d'adapter la traduction en fonction du contexte. Ainsi, le système est capable de simuler un interprète humain, avec un haut degré de précision. Par conséquent, chaque composant, qu'il s'agisse du microphone pour la capture de la voix, du Raspberry Pi pour le traitement des données, ou encore du haut-parleur pour la diffusion du message traduit, est soigneusement sélectionné et configuré pour obtenir une performance optimale. Ce niveau de détail et de sophistication technique fait du GILAN (Gilet d'Interprétation de Langue Assisté par le Numérique) une solution unique sur le marché.

3.2.1 Le Raspberry Pi

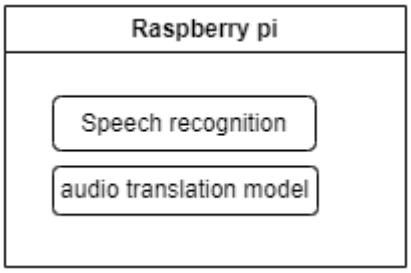


Figure 3-2 Schéma bloc du raspberry pi

Le Raspberry Pi est le pilier central de notre architecture, jouant le rôle de « cerveau ». Ce micro-ordinateur, bien que compact, est extrêmement puissant et polyvalent. Grâce à son architecture ARM et sa capacité à exécuter des systèmes d'exploitation complets tels que Raspbian, il nous permet d'installer des bibliothèques complexes de traitement du signal, des frameworks d'intelligence artificielle, et d'effectuer des calculs intensifs en temps réel. Sa capacité à se connecter facilement à des périphériques via ses multiples interfaces (GPIO, USB,

HDMI, I2C, etc.) le rend idéal pour ce projet, car il nous permet de contrôler simultanément le microphone, le haut-parleur, et l'écran, tout en gérant l'affichage et les interactions avec l'utilisateur.

La décision d'utiliser le Raspberry Pi n'a pas été prise à la légère. Nous avons choisi ce dispositif pour ses capacités à intégrer des modèles d'intelligence artificielle modernes tels que les réseaux de neurones profonds et les transformateurs, qui nécessitent une infrastructure de calcul flexible mais peu gourmande en énergie. De plus, sa compatibilité avec les langages de programmation comme Python facilite le développement et le déploiement d'algorithmes sophistiqués. En centralisant toutes les opérations, le Raspberry Pi assure une orchestration fluide des différentes tâches, que ce soit la capture de la voix, le traitement du signal, la traduction, ou la restitution audio. Il gère également la synchronisation entre les modules, garantissant que le processus de traduction est réalisé sans latence perceptible, ce qui est essentiel pour maintenir la fluidité des conversations.

3.2.2 Utilité du microphone dans le GILAN

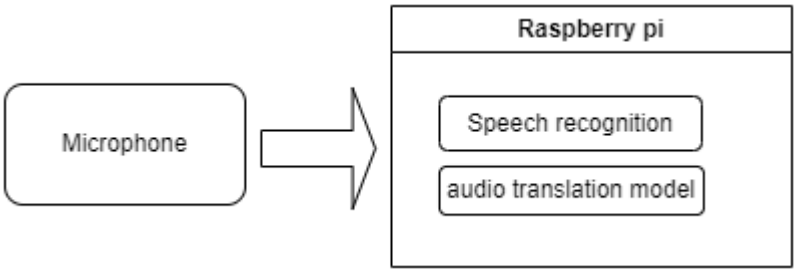


Figure 3-3 Schéma bloc du microphone avec un raspberry pi

Le microphone est l'élément de captation initial de notre système. Il constitue l'interface d'entrée permettant de convertir la parole de l'utilisateur en un signal audio numérique exploitable par le Raspberry Pi. Le microphone est connecté via une interface USB, selon les spécifications choisies, et est configuré pour filtrer les bruits ambiants et se concentrer uniquement sur la voix de l'utilisateur. Le choix d'un microphone à haute sensibilité est crucial

pour assurer la clarté du signal audio, car tout bruit parasite pourrait fausser les résultats de la reconnaissance vocale.

Une fois la voix captée par le microphone, celle-ci est numérisée et transmise au Raspberry Pi pour un prétraitement. Ce prétraitement inclut la réduction du bruit, l'amplification du signal, et l'extraction des caractéristiques vocales telles que la fréquence fondamentale et l'intensité, qui sont ensuite utilisées par le module de reconnaissance vocale pour convertir l'audio en texte. L'efficacité du microphone à capter les subtilités de la voix humaine est primordiale, car une mauvaise qualité de capture se répercuterait directement sur la qualité de la traduction finale. Ainsi, l'utilisation d'un microphone adapté, doté de capacités de suppression de bruit et d'un diagramme de directivité approprié, est un facteur clé dans la réussite de notre projet.

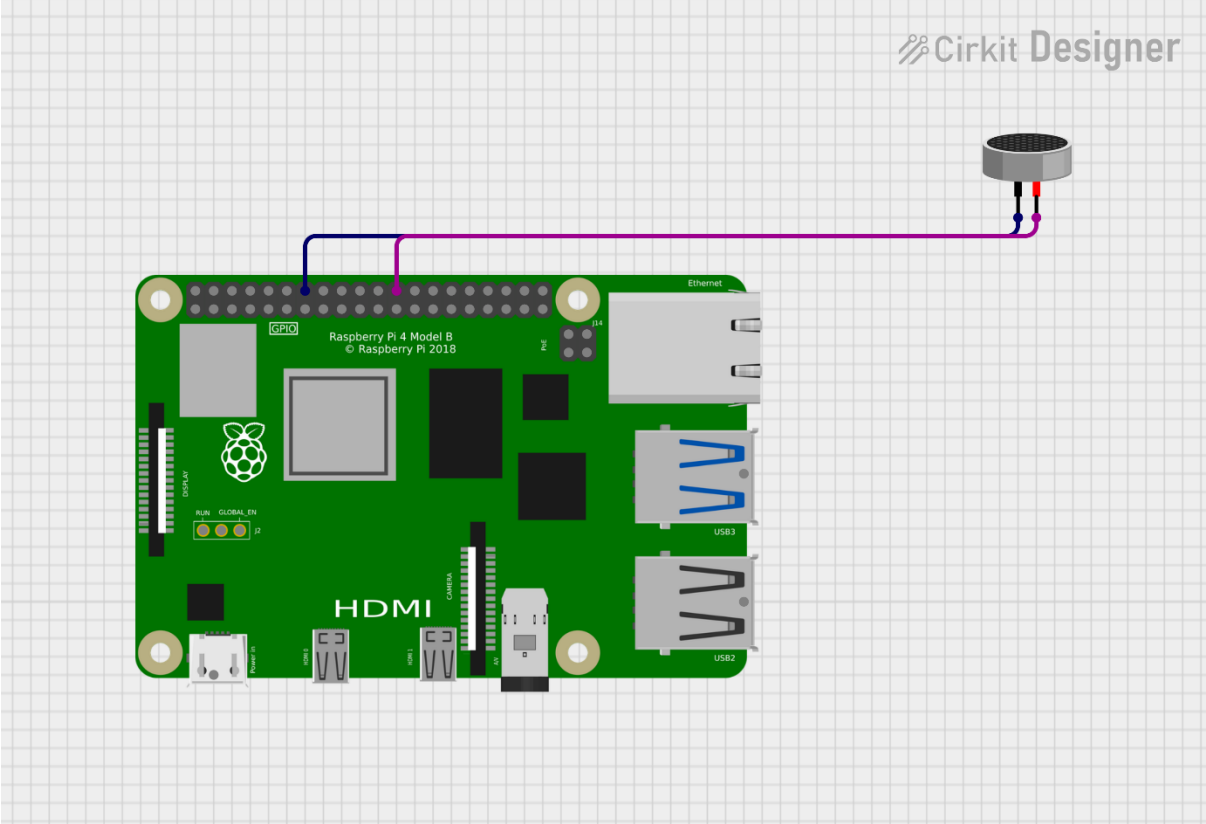


Figure 3-4 Microphone avec raspberry pi

3.2.3 Utilité du haut-parleur dans le GILAN

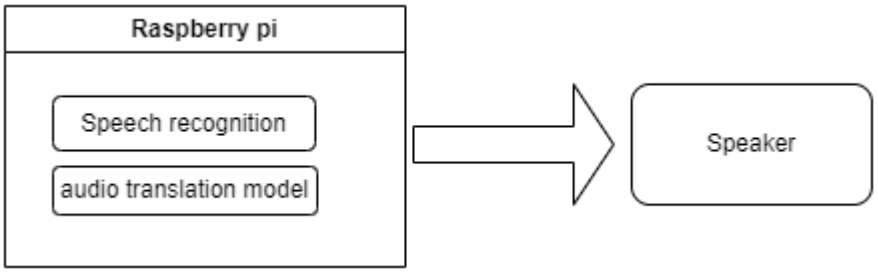


Figure 3-5 Schéma bloc du haut-parleur

Le haut-parleur, quant à lui, est le dispositif de sortie qui permet de diffuser la traduction vocale générée par le système. Il est connecté au Raspberry Pi via une interface audio dédiée, garantissant une restitution sonore de haute qualité. Le choix d'un haut-parleur puissant, capable de reproduire fidèlement les variations de tonalité et de volume, est essentiel pour assurer que la traduction soit clairement audible, même dans un environnement bruyant. Le rôle du haut-parleur dans notre système est de faire la diffusion de la traduction.

Nous avons configuré le haut-parleur pour qu'il puisse émettre un son de qualité studio, tout en minimisant la distorsion et l'écho. Cela est rendu possible grâce à l'utilisation de bibliothèques de traitement audio avancées, qui ajustent dynamiquement les niveaux sonores en fonction du contexte. De plus, le haut-parleur est intégré de manière ergonomique dans le gilet, permettant une diffusion sonore directionnelle. Ce souci du détail dans le choix et l'intégration du haut-parleur contribue à améliorer l'expérience utilisateur globale, rendant notre système à la fois performant et parfait.

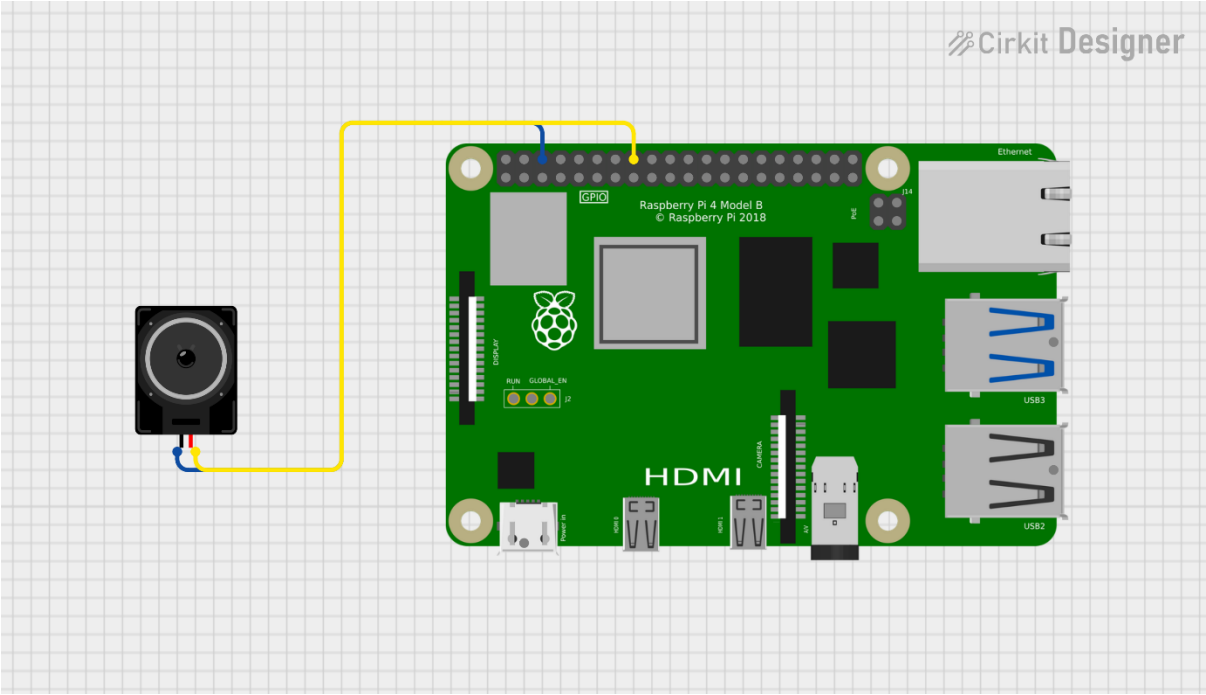


Figure 3-6 Haut-parleur avec raspberry pi

3.2.4 Interface de l'écran du Raspberry pi

Voici une image qui montre l'interface affichée sur l'écran du Raspberry Pi :

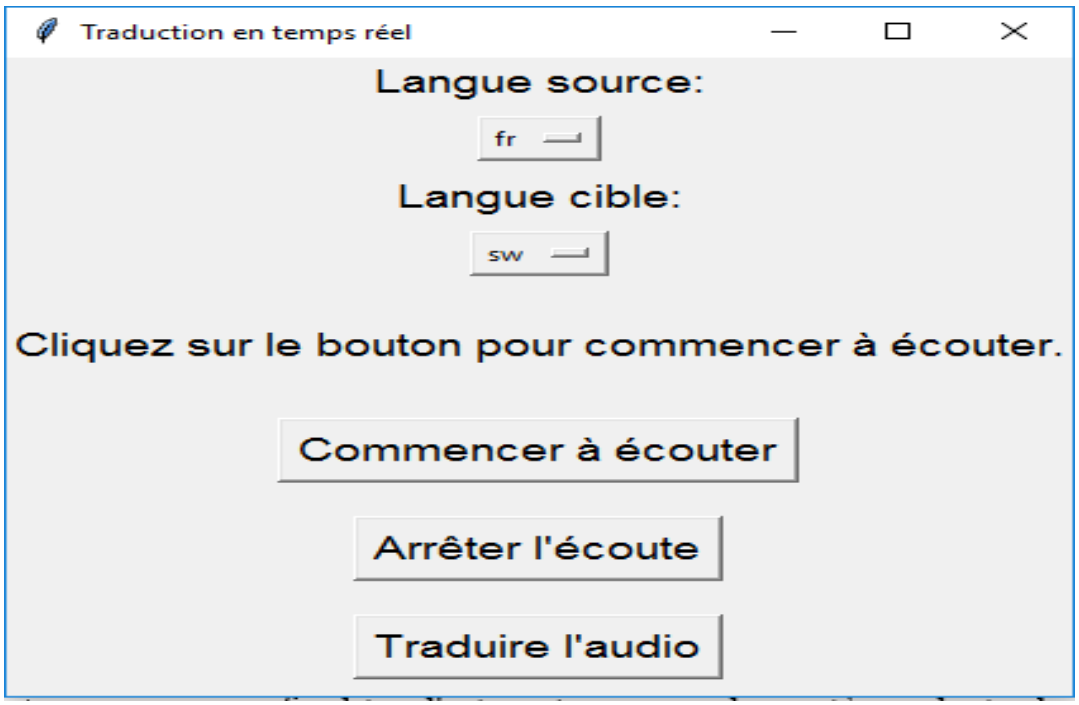


Figure 3-7 Interface de traduction

L'interface est conçue pour faciliter l'interaction avec le système de traduction vocale et comporte les éléments suivants :

- Champs de Sélection des Langues :
  - Champ de Sélection de la Langue Source : Ce champ permet de choisir la langue de l'audio d'entrée. Il est crucial pour s'assurer que le modèle de reconnaissance vocale puisse interpréter correctement le discours initial.
  - Champ de Sélection de la Langue Cible : Ce champ permet de sélectionner la langue dans laquelle l'audio sera traduit. Ce choix détermine la langue finale de sortie après la traduction.
- Affichage du Texte Capté : Avant que l'audio d'entrée ne soit traité par le modèle de traduction, il est important de vérifier que ce que vous avez dit a été correctement capté. C'est pourquoi l'interface comporte une zone d'affichage pour le texte résultant de la reconnaissance vocale. Nous utilisons un modèle existant pour traduire la voix en texte.

Ce modèle est choisi pour sa capacité à fournir des transcriptions précises et efficaces du discours en texte.

- Boutons d'Action :
  - **Bouton "Commencer à Capter"** : Lorsque ce bouton est cliqué, la reconnaissance vocale commence à capter votre discours. Il initialise le processus de capture audio.
  - **Bouton "Arrêter l'Écoute" :** Ce bouton arrête la capture audio une fois que vous avez terminé de parler. Il affiche le texte transcrit à partir de votre discours.
  - **Bouton "Traduire l'Audio"** : Une fois que vous êtes satisfait de la transcription affichée, cliquez sur ce bouton pour envoyer l'audio au modèle de traduction. Ce modèle génère l'audio souhaité dans la langue cible sélectionnée.

3.2.5 Utilité de l'intelligence artificielle dans le GILAN

L'intelligence artificielle joue un rôle central dans notre système en traduisant une langue source vers une langue cible. Nous avons intégré deux modèles distincts dans notre gilet : l'un pour la reconnaissance vocale, qui se charge de convertir la voix en texte, et l'autre pour la traduction.

Bien que nous aurions pu opter pour un seul modèle dédié à la traduction, cette approche aurait présenté des limites. En effet, lors de la capture audio, le système peut mal interpréter le son d'entrée, ce qui pourrait conduire à une traduction erronée, même avec un modèle de traduction bien conçu. Si l'audio est mal capturé, le processus de traduction en sera gravement affecté.

C'est pourquoi nous avons choisi d'utiliser deux modèles distincts : l'un pour la reconnaissance vocale et l'autre pour la traduction. Le modèle de reconnaissance vocale intervient dès que l'utilisateur parle, convertissant l'audio en texte. Ce texte est ensuite affiché à l'écran, permettant de vérifier si ce qui a été dit a été correctement capté par le système. Après cette vérification, nous pouvons choisir de procéder à la traduction de l'audio.

Nous avons opté pour l'approche de "*speech-to-speech translation with discrete units*", qui utilise un modèle basé sur les *transformeurs* (*s2ut_transformer_fisher*) pour réaliser la traduction. Cette approche s'est révélée prometteuse pour la traduction de la parole à la parole.

3.3 Présentation de l'architecture globale du GILAN

Le développement de notre GILAN intègre divers composants matériels soigneusement sélectionnés pour assurer une performance optimale. Les composants principaux incluent un microphone, un Raspberry Pi 4 Model B/4GB, un haut-parleur, une alimentation régulée, et un écran tactile de 5 pouces. Voici un schéma qui montre le système avec ses parties.

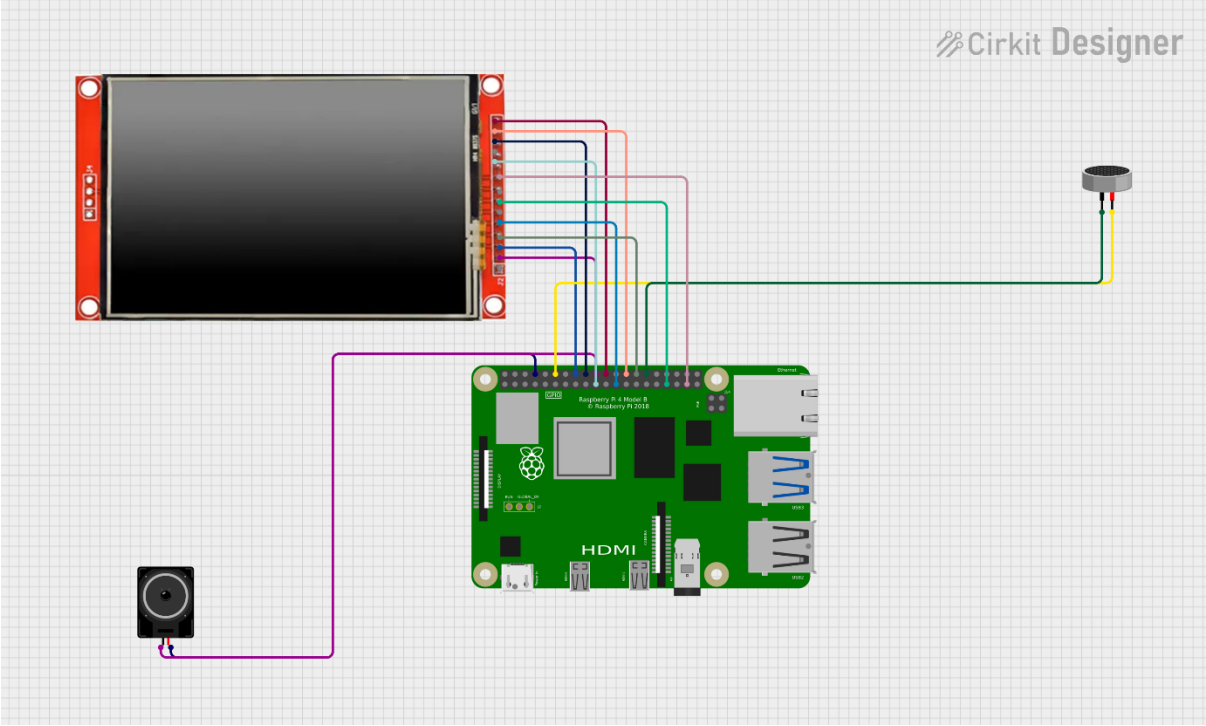


Figure 3-8 Connexion entre les composants

3.4 Évaluation du Modèle

L'évaluation de notre modèle de traduction vocale a été réalisée en deux étapes principales :

- **Reconnaissance Automatique de la Parole :** Nous avons appliqué un modèle de reconnaissance automatique de la parole sur les sorties vocales traduites. Cela a permis de convertir les voix en texte, fournissant ainsi des transcriptions automatiques. Pour les langues que le modèle ne reconnaissait pas, nous avons manuellement fourni les traductions, puis observé la sortie du modèle pour ces cas. Cette méthode nous a permis de vérifier si le modèle pouvait tout de même produire des résultats cohérents même sans support automatique pour ces langues.
- **Calcul du Score BLEU (***Bilingual Evaluation Understudy*) : Pour mesurer la qualité de la traduction de notre modèle de *Direct speech-to-speech translation with discrete units*, nous avons comparé les transcriptions automatiques avec les transcriptions de référence en utilisant le score BLEU. Ce score, couramment utilisé dans le domaine de la traduction automatique, permet d'évaluer la fidélité de la traduction par rapport à un texte de référence.

3.4.1 Résultats de l'Évaluation

3.4.1.1 Définition de la métrique BLEU

Le score BLEU (Bilingual Evaluation Understudy) est une métrique couramment utilisée pour évaluer la qualité des traductions automatiques. Il compare les traductions générées par un modèle avec des traductions de référence, en mesurant la correspondance entre les n-grammes (séquences de mots) des deux ensembles. Un score BLEU plus élevé indique une meilleure qualité de traduction [30].

3.4.1.2 Application du score BLEU dans notre modèle

Pour mesurer le score BLEU dans le cadre de notre approche de *Direct speech-to-speech translation with discrete units*, nous avons suivi les étapes suivantes :

1.Collecte des données : Nous avons constitué un corpus initial pour six langues : le français, l'anglais, le swahili, le chinois le coréen et le lingala. Pour chaque langue, nous avons collecté 50 enregistrements audios, chaque enregistrement étant associé à une transcription écrite de référence.

2.Calcul du score : Nous avons utilisé la formule standard du score BLEU, qui compare chaque traduction générée par notre modèle à ses transcriptions de référence. Le score BLEU est calculé sur la base de la correspondance n-gramme (séquences de n mots consécutifs), où nous avons utilisé des n-grammes allant de 1 à 4 pour capturer à la fois des correspondances lexicales simples et des séquences plus longues. Les résultats sont exprimés en pourcentage, où 0 indique aucune correspondance et 1 (ou 100 %) indique une correspondance parfaite.

3.Résultats : Les scores BLEU ont été analysés pour déterminer la performance de notre modèle sur différentes langues.

3.4.1.3 **Scores par langue**

Les langues avec des scores élevés : Notre modèle a été évalué sur un ensemble de 50 phrases pour chaque langue, basées sur 50 échantillons audio distincts. Les langues telles que le français, le swahili, l'anglais, le coréen et le chinois ont montré de très bons résultats. Grâce à un large corpus de données de formation, ces langues ont permis au modèle de bien généraliser, atteignant des scores BLEU satisfaisants compris entre 0,70 et 0,85. Les traductions étaient précises et les sorties vocales se rapprochaient fortement des transcriptions de référence. Voici quelques exemples de phrases traduites avec leur score BLEU :

- Tableau pour le Français

Transcription de Référence	**Traduction Générée**	**Score BLEU**
"Le chat saute vite sur la table."	"Le chat bondit rapidement sur la table."	0.85

"Les élèves étudient à la bibliothèque."	"Les élèves travaillent à la bibliothèque chaque jour."	0.78
"Il fait beau aujourd'hui, le soleil brille."	"Il fait beau aujourd'hui, et le soleil brille."	0.82
"Les enfants jouent tout l'après-midi dans le jardin."	"Les enfants jouent dans le jardin toute l'après-midi."	0.80

Tableau 3-1 Score BLUE pour le francais

- Tableau pour l'Anglais

Transcription de Référence	**Traduction Générée**	**Score BLEU**
"The cat jumps swiftly onto the table."	"The cat jumps quickly onto the table."	0.80
"Students work diligently in the library every day."	"Students study hard in the library every day."	0.75
"It is a sunny day, and the sky is bright today."	"It is a sunny day, and the sky is clear today."	0.78
"Kids play in the park every afternoon after school."	"Children play in the park each afternoon after school."	0.79

Tableau 3-2 Score BLUE pour l'anglais

- Tableau pour le Swahili

Transcription de Référence	Traduction Générée	Score BLEU
"Paka anaruka kwa haraka juu ya meza."	"Paka anaruka kwa haraka juu ya meza."	0.78
"Wanafunzi husoma sana kwenye maktaba kila siku."	"Wanafunzi husoma kwa bidii kwenye maktaba kila siku."	0.76
"Leo kuna jua kali, na anga ni wazi na safi."	"Leo kuna jua kali, na anga ni safi."	0.80
"Watoto wanacheza bustanini kila jioni pamoja na marafiki."	"Watoto wanacheza kwenye bustani kila jioni na marafiki."	0.74

Tableau 3-3 Score BLUE pour le swahili

Les langues avec des Scores Moins Élevés : En revanche, pour des langues comme le lingala, nous avons rencontré des défis, nous avons effectué une évaluation basée sur 50 phrases distinctes. Le manque de données disponibles a entraîné une faible performance du modèle lors de la validation, avec des scores BLEU inférieurs à 0.50. Cela met en lumière les limitations du modèle lorsqu'il est appliqué à des langues sous-représentées dans le corpus de formation. Voici quelques résultats obtenus lors de l'évaluation du modèle :

Transcription de Référence	Traduction Générée	Score BLEU
"Mpese akuti mesa mbangu, akoti te."	"Mpese akuti mesa mbangu, akoti ."	0.45

"Bana baleki sima na ndako mokolo mobimba na maboko."	"Bana baleki na ndako sima mokolo mobimba na mabo."	0.48
"Mokonzi azali makasi mingi, akopesa eloko."	"Mokonzi azali makasi, akopesa ."	0.47
"Moto asali likambo esengo mingi, balekisi seko."	"Moto asali likambo ya esengo mingi, bakisi ko."	0.46

Tableau 3-4 Score BLUE pour le lingala

3.5 Interface de la plateforme de chat

Comme nous l'avons précédemment évoqué, nous avons conçu une plateforme de chat où chaque utilisateur a la possibilité de recevoir les messages dans la langue de son choix. Pour mettre en place cette plateforme, nous avons utilisé le framework Django, qui offre une solution robuste pour le développement web.

Voici un aperçu des différentes sections de la plateforme de chat :

- **Page de Connexion (Login)** : Elle permet aux utilisateurs de se connecter à leur compte en entrant leurs identifiants. La page est conçue pour être sécurisée et intuitive, offrant un accès facile à l'application pour les utilisateurs enregistrés.

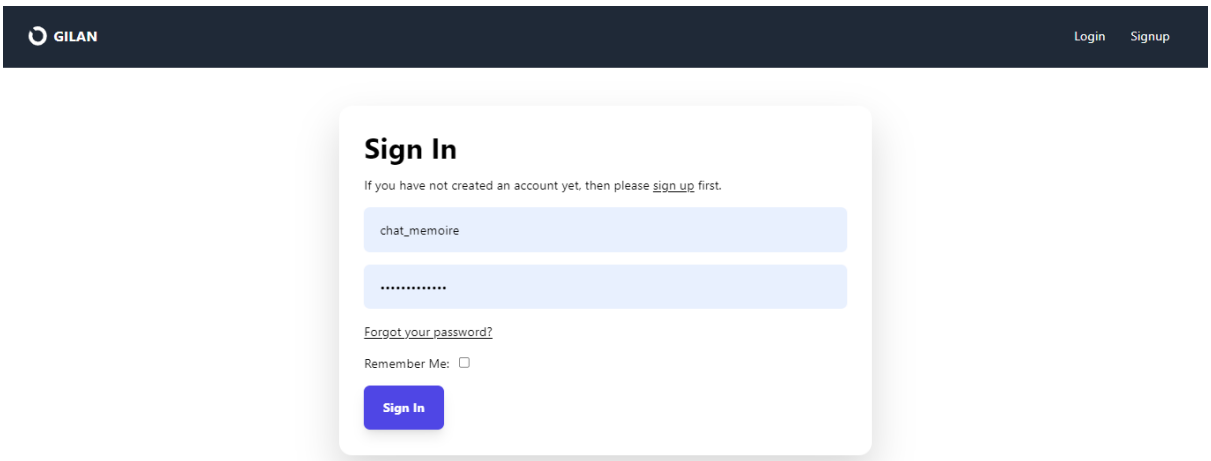


Figure 3-9 Page de connexion

- **Page d'Inscription (Signup)** : Cette page permet aux nouveaux utilisateurs de créer un compte en fournissant leurs informations personnelles. L'inscription est simple et rapide, avec des champs pour le nom, l'email, la langue préférée, et le mot de passe.

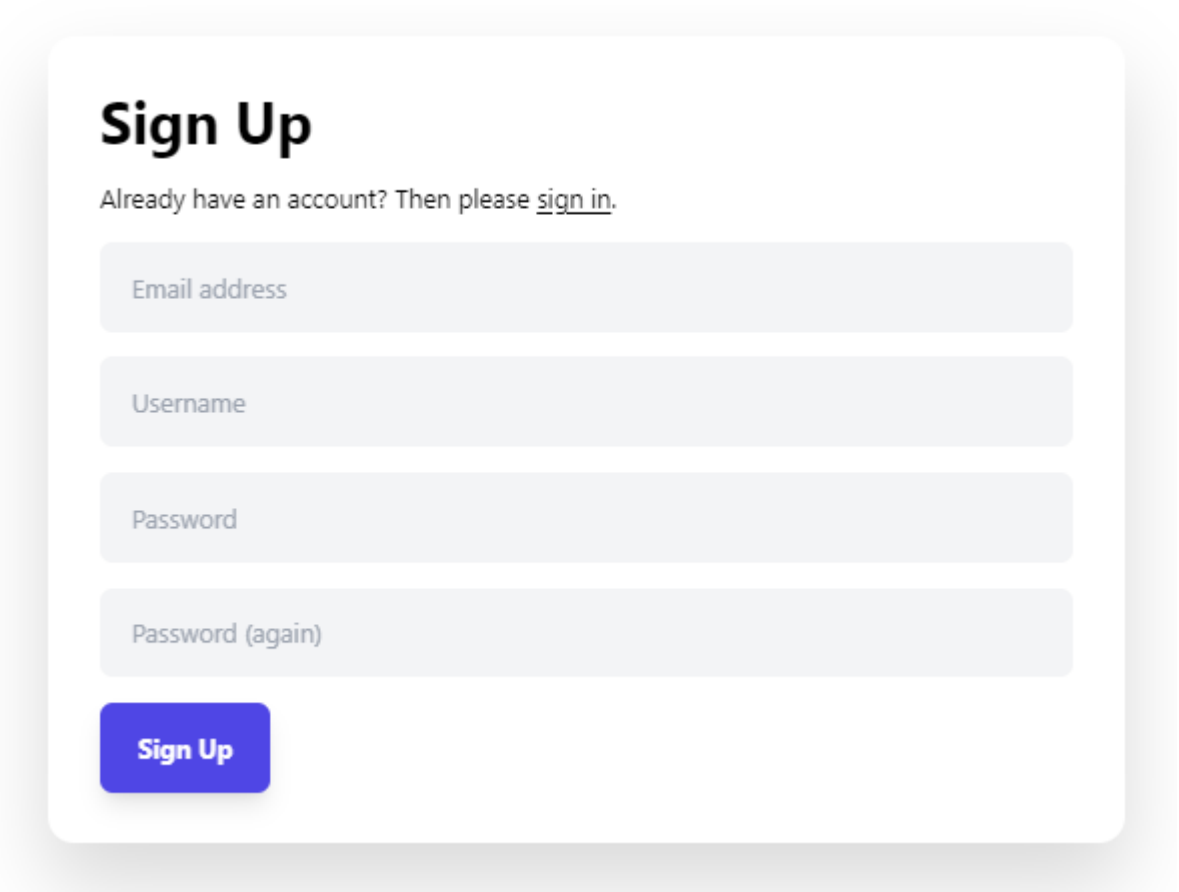


Figure 3-10 Page d'inscription

- **Page d'Accueil (Home)** : La page d'accueil présente une vue d'ensemble des fonctionnalités de la plateforme. Elle affiche les chats récents, les groupes de discussion disponibles, et un accès rapide aux différentes sections de l'application. Cette section permet aux utilisateurs de participer à des discussions de groupe. Ils peuvent rejoindre différents groupes, envoyer des messages à l'ensemble du groupe, et voir les conversations en temps réel.

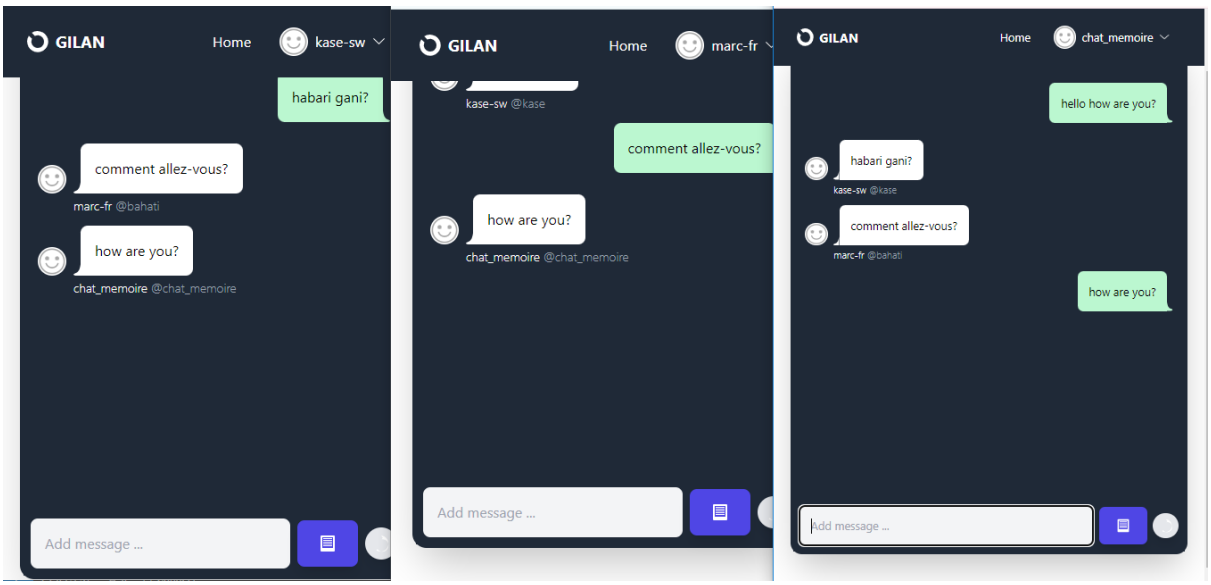


Figure 3-11 Chat en groupe avant la traduction

- **Partie Chat Privé** : Les utilisateurs peuvent avoir des conversations privées avec d'autres membres. Cette section offre un espace pour des discussions individuelles, avec la possibilité d'envoyer des messages directs.

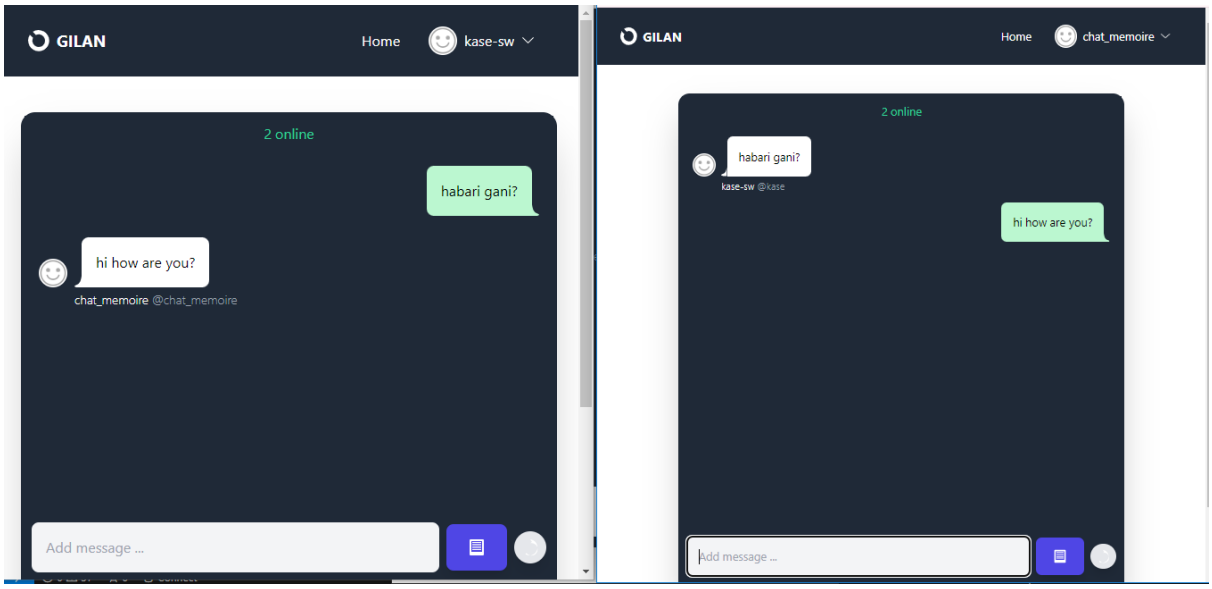


Figure 3-12 Chat en privé avant la traduction

- **Icone de Traduction** : Cette icône, située à côté de celle d'envoi des messages en rond blanc, permet de traduire les messages reçus dans la langue choisie lors de la création du compte. Par défaut, les messages sont affichés dans la langue du destinataire. Cependant, en cliquant sur cette icône, les utilisateurs peuvent voir le message traduit dans leur langue préférée.

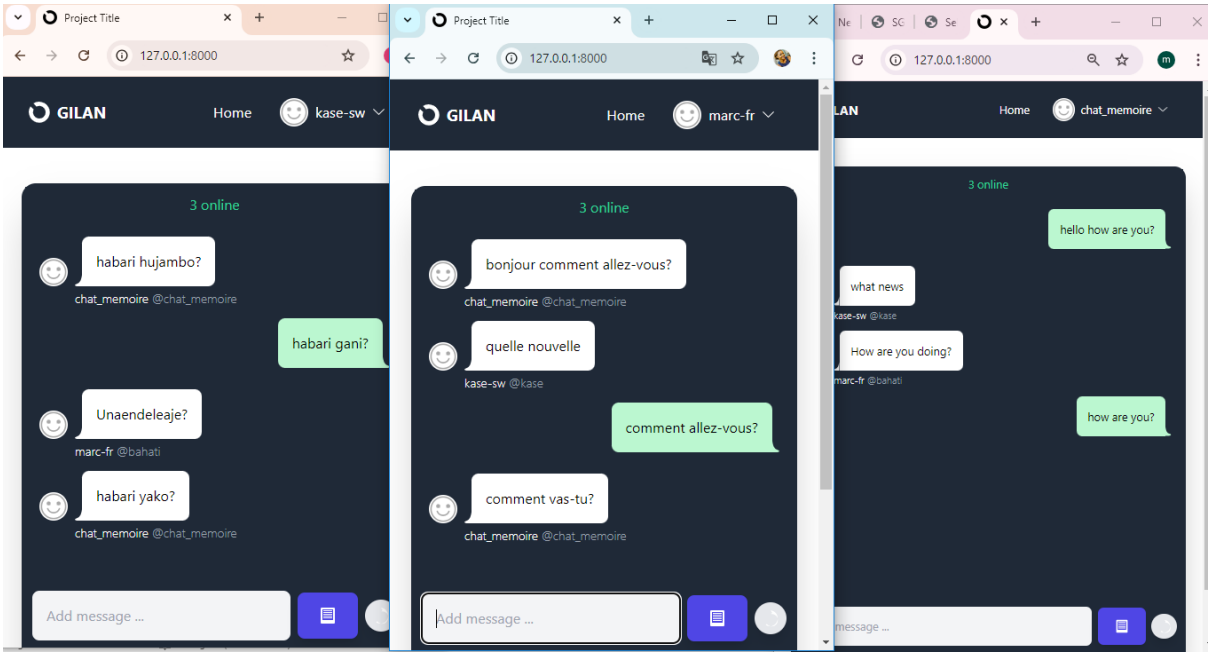


Figure 3-13 Traduction du message en groupe

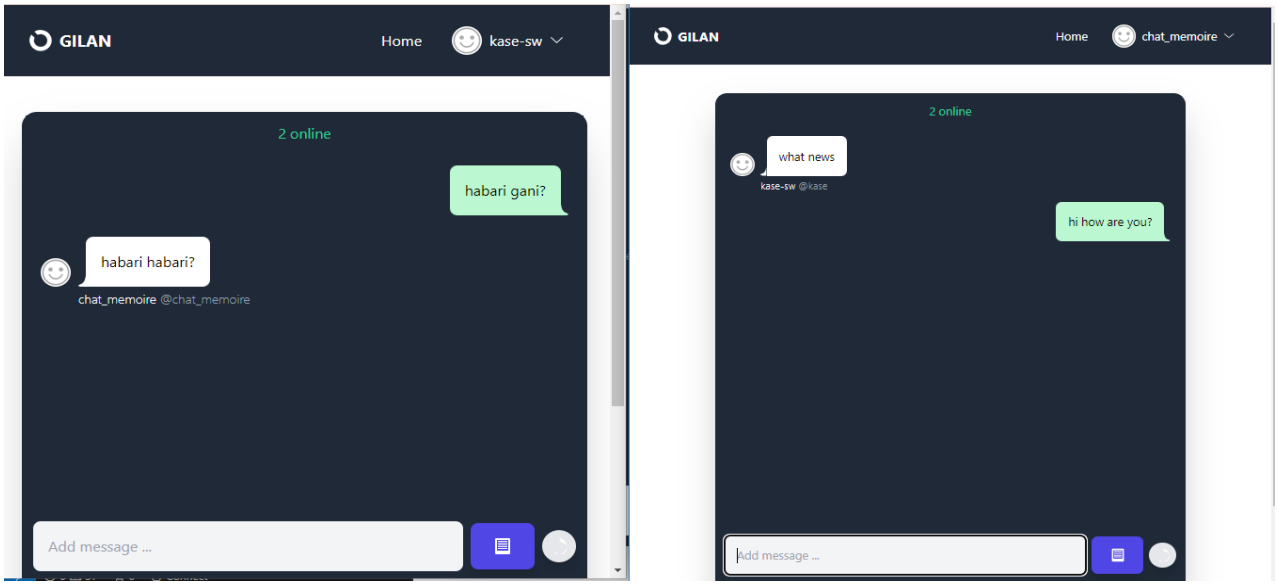


Figure 3-14 Traduction du message en privé

3.6 Architecture globale du système

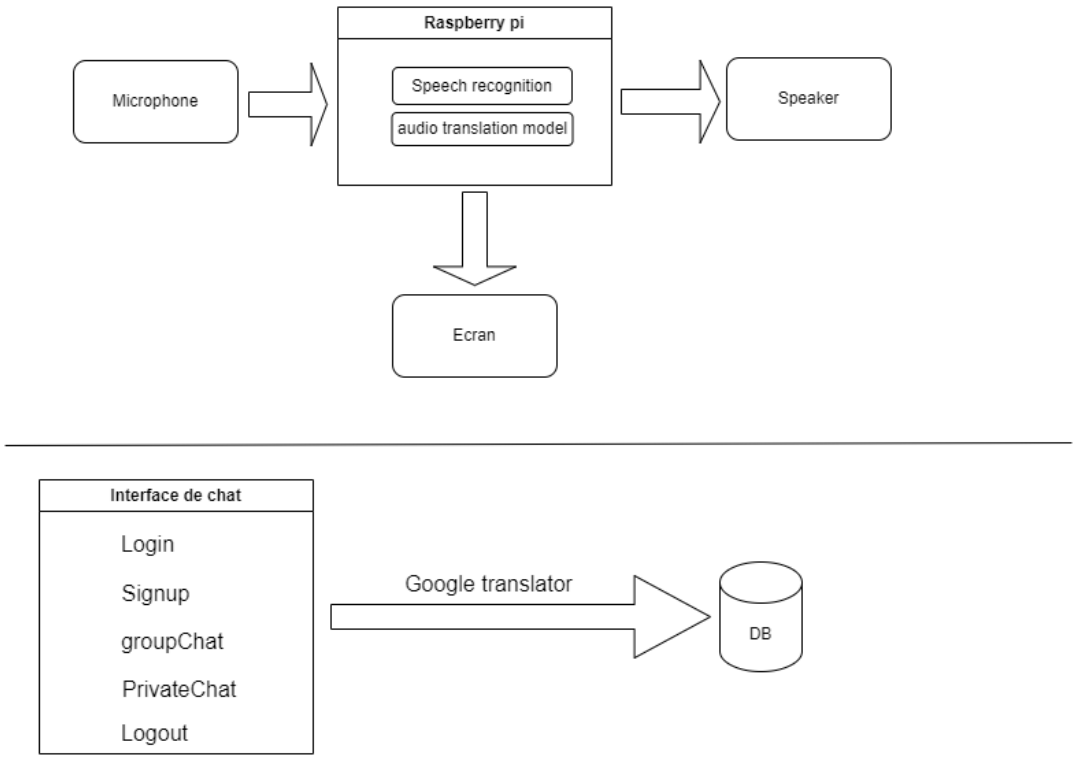


Figure 3-15 Architecture globale du système

Comme illustré dans le schéma ci-dessus, deux systèmes distincts ont été développés pour la traduction linguistique. Le premier, destiné à un gilet d'interprète, permet de traduire une langue vers une autre, avec une précision accrue. Lorsqu'une personne parle dans un microphone connecté à un Raspberry Pi, la voix est d'abord captée et convertie en texte par un modèle de reconnaissance vocale. Cependant, ce texte n'est utilisé que pour vérifier si la voix a bien été transcrite fidèlement. Si le texte correspond exactement à ce qui a été dit, l'utilisateur peut alors appuyer sur un bouton pour activer la traduction audio dans une autre langue. Ce processus repose sur un modèle de traduction vocale de type "speech-to-speech" utilisant des unités discrètes, où ce n'est pas le texte qui est traduit, mais bien l'audio. Le texte sert

uniquement à vérifier la qualité de la reconnaissance vocale initiale, garantissant ainsi une interprétation correcte avant la conversion audio.

Le second système, intégré à une plateforme de chat en ligne, offre une fonctionnalité différente mais complémentaire. Ici, un utilisateur peut envoyer un message dans sa langue, et le destinataire a la possibilité de traduire ce message dans sa langue native. Le processus repose sur un traducteur automatique, tel que Google Translator, qui s'interface avec une base de données pour stocker et gérer les échanges. L'interface de chat propose les fonctionnalités standards, telles que l'inscription, la connexion, les discussions de groupe et privées, ainsi que la déconnexion.

En somme, ces deux systèmes exploitent des technologies de traitement du langage naturel avancées, avec une approche innovante pour la traduction vocale et textuelle, garantissant ainsi une fluidité et une précision dans les interactions multilingues, que ce soit en direct ou en ligne.

3.7 Conclusion partielle

Nous avons abordé en détail la réalisation et l'évaluation du système GILAN ainsi que de la plateforme de chat associée. Nous avons commencé par le développement et l'intégration des composants matériels du GILAN, en mettant l'accent sur les choix technologiques qui assurent une performance optimale. Le Raspberry Pi 4 Model B, le microphone, le haut-parleur, l'alimentation, et l'écran tactile ont été soigneusement sélectionnés pour leur adéquation avec les exigences du projet. Nous avons ensuite détaillé l'interface utilisateur du système de traduction vocale, décrivant ses éléments essentiels tels que les champs de sélection des langues, les boutons d'action, et la zone d'affichage du texte capté. Cette interface permet aux utilisateurs de vérifier la précision de la reconnaissance vocale avant de traduire l'audio dans la langue cible.

La mise en place de la plateforme de chat a également été réalisée avec succès, offrant des fonctionnalités telles que la traduction automatique des messages. L'utilisation de Django a

permis de développer une solution web robuste permettant une communication multilingue efficace.

Conclusion générale

Contributions

Dans ce mémoire, nous avons approfondi la conception et l'implémentation d'un système de traduction basé sur l'intelligence artificielle, en nous concentrant particulièrement sur le Gilet d'Interprète du Langage Naturel (GILAN) et une plateforme de chat associée. Notre modèle repose sur un réseau de Transformers (*s2ut_transformer_fisher*) pour encoder les données de la parole en séquences d'unités discrètes, permettant ainsi un traitement efficace des séquences longues et des dépendances à longue portée dans la parole.

- **Utilisation des systèmes de traduction Speech-to-Speech (S2ST) pour garantir une traduction précise et fluide en temps réel :** Le modèle S2UT, utilisant un encodeur Transformer pour traiter les caractéristiques de la parole extraites par des modèles comme HuBERT, offre une traduction précise et fluide. Cette approche nous a permis de répondre à notre objectif en convertissant la parole en séquences discrètes avant de procéder à la traduction.

- **Conception d'un dispositif portable pour la traduction instantanée et son intégration dans une plateforme de chat en ligne :** La conception d'un dispositif portable, tel que le Gilet d'Interprète du Langage Naturel (GILAN), a permis d'améliorer la performance et l'efficacité du système en offrant une solution de traduction instantanée. L'intégration de cette fonctionnalité dans une plateforme de chat en ligne a optimisé l'expérience utilisateur en facilitant la communication multilingue.
- **Prise en charge des langues locales rares pour leur préservation et leur utilisation dans un contexte global** : Notre système se distingue par sa capacité à prendre en charge des langues locales, souvent ignorées dans les solutions de traduction classiques. En intégrant ces langues dans le processus de traduction en temps réel, nous contribuons

non seulement à leur préservation, mais nous facilitons également les échanges interculturels au sein de communautés linguistiques diversifiées.

- **Architecture globale la plus adaptée pour concevoir un système de traduction efficace :** Nous avons déterminé que l'architecture basée sur les Transformers est la plus adaptée pour concevoir un système de traduction performant. Cette approche permet de gérer efficacement les séquences longues et les dépendances linguistiques complexes, offrant ainsi une qualité de traduction supérieure par rapport aux autres méthodes.

Critique du travail

Nous n'avons pas pu intégrer toutes les langues locales souhaitées en raison d'un manque de données linguistiques disponibles. Les langues telles que le tshiluba, le kikongo et d'autres dialectes régionaux n'ont pas été correctement prises en charge. Cette limitation est due à un déficit de corpus annotés et de ressources linguistiques adaptées pour l'entraînement du modèle. En conséquence, le système fonctionne de manière optimale uniquement avec un nombre restreint de langues locales. De plus, cela a eu un impact sur la qualité de la traduction et la fluidité du dialogue pour ces langues. L'obtention de données supplémentaires est donc nécessaire pour assurer une couverture complète et une traduction précise de toutes les langues ciblées.

Travaux futurs de recherche/Perspectives

Pour développer davantage ce projet, plusieurs pistes de recherche et améliorations peuvent être envisagées :

- **Optimisation des modèles pour les langues moins représentées** : Il serait bénéfique d'explorer des techniques comme le transfert learning pour améliorer les traductions dans les langues avec des ressources limitées.

- **Amélioration des performances du dispositif portable** : La recherche de microcontrôleurs plus puissants ou l'optimisation des composants matériels du GILAN pourrait permettre de surmonter les limitations actuelles et d'améliorer la performance globale du système.

- **Expansion des fonctionnalités de la plateforme de chat :** Envisager l'ajout de fonctionnalités supplémentaires, telles que la traduction contextuelle avancée et des améliorations de l'interface utilisateur, pourrait enrichir l'expérience utilisateur. L'intégration de la traduction en temps réel pour les vidéos et les appels reste une direction prometteuse pour rendre le système encore plus polyvalent et efficace.

Bibliographie

[1] Tomorrow.bio, "Communication multilingue : Briser les barrières linguistiques grâce à la traduction assistée par ordinateur," 28 mai 2024.

[2] Africanews, "Pour une intelligence artificielle au service des langues africaines," 23 décembre 2021.

[3] A. Medad, «Les Techniques de base du NLP».

[4] R. Bonfillon, "La reconnaissance vocale et le NLP : sur la voix du succès ?," 2022.

[5] Publithings, "guide complet du traitement du langage naturel," 2024.

[6] Golem.ai, "Le langage naturel et sa complexité," 2024.

[7] K. K. David, CONCEPTION ET REALISATION D'UN SYSTEME BASE SUR L'INTELLIGENCE POUR RESUMER AUTOMATIQUEMENT LES TEXTES, 2021 − 2022.

[8] K. Vancappel, «TUTORIEL | Machine Learning : comprendre ce qu'est un réseau de neurones et en créer un !,» *La révolution Data & IA par ceux qui la font,* 2023.

[9] É. Blent, «Réseaux récurrents (RNN) : on vous explique tout,» 2022.

[10] P. Bio, «Fabrique & comprend ton premier réseau de neurones en partant de zéro !,» 2018.

[11] M. Abdallah, «Comparaison et architecture de LSTM, GRU et RNN. Quels sont les problèmes rencontrés avec RNN pour traiter de longues séquences,» 2023.

[12] O. Education, "Recurrent Neural Network (RNN) : de quoi s'agit-il ?," 2024.

[13] IBM, «Qu'est-ce que la descente de gradient ?,» 2024.

[14] Y. AMIRAT, EXTRACTION D'ENTITÉS NOMMÉES PAR APPRENTISSAGE PROFOND, 2020.

[15] L. R., "Comprendre le fonctionnement d'un LSTM et d'un GRU en schémas".

[16] O. F. Rokon, "RNN vs. LSTM vs. Transformers: Unraveling the Secrets of Sequential Data Processing," 2023.

[17] R. d’Assignies, «Les Transformers,» 2023.

[18] DataScientest, «Transformer Neural Network : Qu’est-ce que c’est ? Comment ça fonctionne ?,» 2024.

[19] E. Series, "Encoders in NLP," 2022.

[20] T. KELDENICH, «Qu’est-ce qu’un Encodeur Décodeur et pourquoi l’utiliser ? – Explication Facile,» 2021.

[21] P.-J. C. C. W. J. G. S. P. X. M. A. P. Y. A. Q. H. Y. T. J. P. W.-N. H. Ann Lee, "Direct speech-to-speech translation with discrete units," Submitted on 12 Jul 2021 (v1), last revised 21 Mar 2022 (this version, v2.

[22] B. B. Y.-H. H. T. K. L. R. S. A. M. Wei-Ning Hsu, "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units," 2021.

[23] D. erran, "Machine Translation With Natural Language Processing: A Strategy for Artificial Intelligence," 2023.

[24] S. Robertson, "PNL à partir de zéro : traduction avec un réseau de séquences à séquences et attention," 2024.

[25] salhifatimazahra57, Les Avantages Et Les Inconvénients de Raspberry, transfertle Dec 04, 2023.

[26] L. é. IONOS, "30 idées de projets avec Raspberry Pi," 04/07/2022.

[27] Damien.SO, "Tout savoir sur le Raspberry Pi," 30.11.2023 .

[28] P.-J. C. C. W. J. G. S. P. X. M. A. P. Y. A. Q. H. Y. T. J. P. W.-N. H. Ann Lee, "Direct speech-to-speech translation with discrete units," 2021.

[29] A. Marotel, "Google Translate : Guide Complet Du Traducteur Multilingue," 11/04/2022 .

[30] R. Demetrio, "Qu'est-ce que le score BLEU ?," 2023.

[31] JUnit.org. [Online]. Available: http://www.junit.org. [Accessed 2 Janvier 2019].

[32] L. A3P, "L’art de comprendre le langage : l’évolution du traitement automatique du langage," 2024.

[33] B. L., «Réseau de neurones artificiels : qu'est-ce que c'est et à quoi ça sert ?,» 2019.

[34] P. STORAGE, «Qu'est-ce que le prétraitement des données pour l'apprentissage machine ?,» 2024.

[35] K. SHI, «La tokenisation avec des sous-mots : une révolution dans le traitement du langage naturel,» 2023.

[36] S. C. Enterprise, "Subword Tokenization," 2024.

[37] S. Nord, «Dédoublement et lemmatisation».

[38] M. Macary, «Analyse de données massives en temps réel pour,» 2022.

[39] M. A. Yimer, «APPROCHE D'APPRENTISSAGE AUTOMATIQUE POUR LA DÉTECTION DE SEGMENTS DE PAROLE VOIX/SANS VOIX/SILENCE,» 2013.

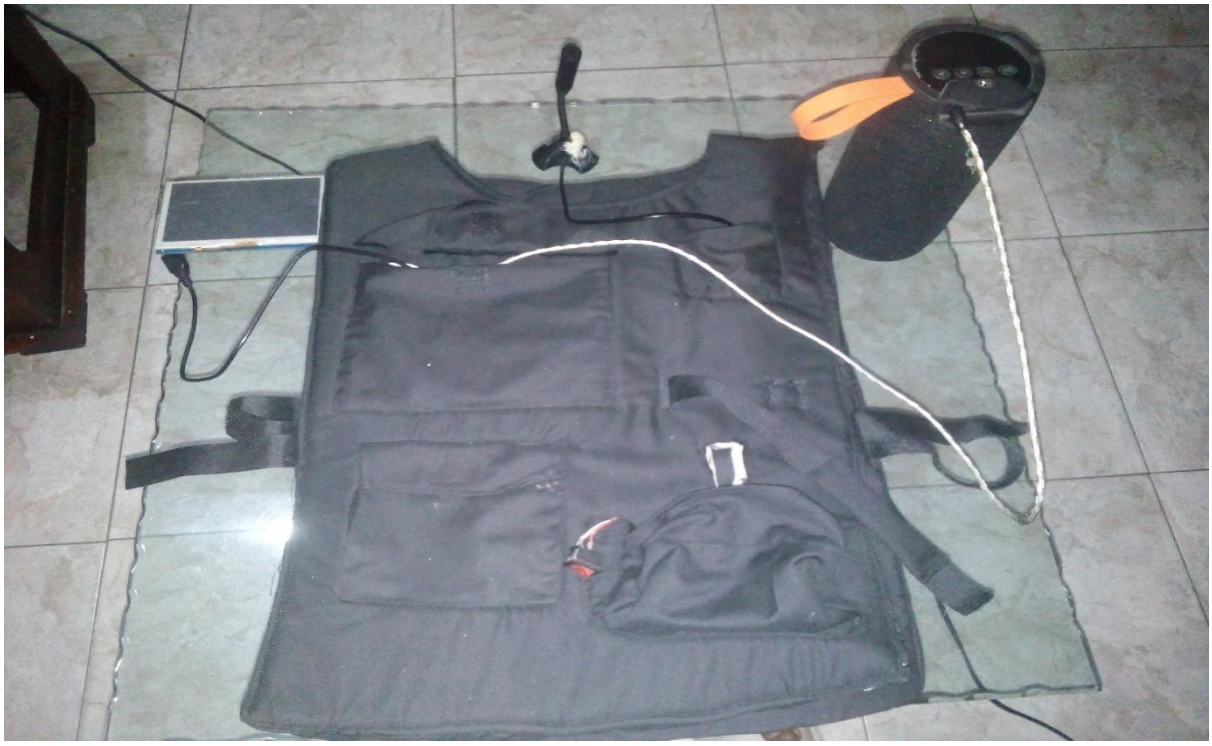
[40] B. P. ,. C. P. Ludovic Lebart, Analyse des données textuelles, 2024.

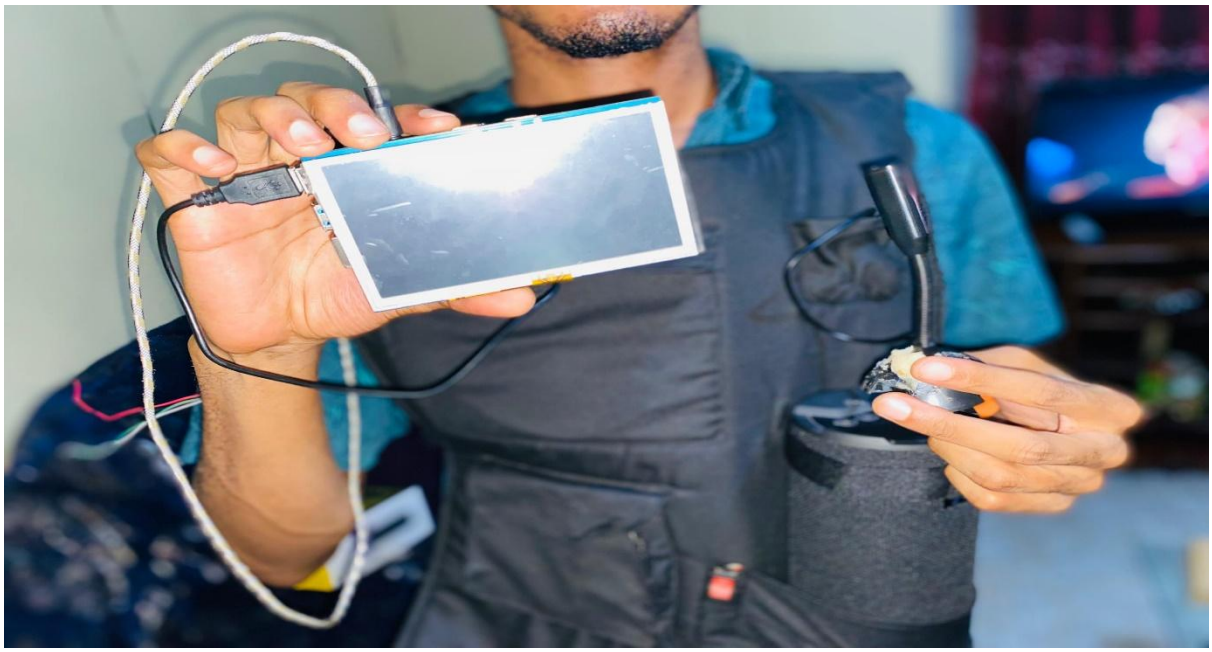
[41] E. Nachmani, "Unsupervised speech-to-speech translation from monolingual data," 2023.

[42] J. Niehues, "End-to-End Speech Translation," 2021.

ANNEXE

1. GILLET D’INTERPRETE DU LANGAGE NATUREL





2. QUELQUES CODES SOURCES

```python
import nltk
from nltk.translate.bleu_score import corpus_bleu

# Télécharger le tokenizer punkt si ce n'est pas déjà fait
nltk.download('punkt')

# Exemple de phrases de référence et de prédiction
references = [["Les étudiants étudient à l'école"]]  # Références possibles pour la traduction
candidate = "Les étudiants travaillent à l'école."# Phrase générée par le modèle

# Tokeniser les phrases de référence et de prédiction
reference_tokens = [[nltk.word_tokenize(ref) for ref in references[0]]]  # Liste de listes de tokens
candidate_tokens = nltk.word_tokenize(candidate)  # Liste de tokens pour le candidat

# Calculer le score BLEU
# Format correct : `corpus_bleu` prend une liste de listes de listes pour les références
# et une liste de listes pour les candidats
bleu_score_value = corpus_bleu(reference_tokens, [candidate_tokens])

print(f"Score BLEU : {bleu_score_value:.4f}")
```

```
{% block content %}

<wrapper class="block max-w-2xl mx-auto my-10 px-6">
    {% if chat_group.groupchat_name %}
    <div class="flex justify-between">
        <h2>{{ chat_group.groupchat_name }}</h2>
        {% if user == chat_group.admin %}
        <a href="{% url 'edit-chatroom' chat_group.group_name %}">
            <div class="p-2 bg-gray-200 hover:bg-blue-600 rounded-lg group">
                <svg class="fill-gray-500 group-hover:fill-white" width="16" height="16">
                    <path d="M11.013 1.427a1.75 1.75 0 0 1 2.474 0l1.086 1.086a1.75 1.75 0 0 1 0 2.474l-8.61 8.61c-.21.21-
                </svg>
            </div>
        </a>
        {% endif %}
    </div>
    {% endif %}
    <div id="chat_window" class="h-[45rem] flex flex-col bg-gray-800 rounded-2xl shadow-2xl relative p-1">
        <div class="flex justify-center text-emerald-400 bg-gray-800 p-2 sticky top-0 z-10">
            {% if other_user %}
            <div id="online-icon" class="gray-dot absolute top-2 left-2"></div>
            <a href="{% url 'profile' other_user.username %}">
                <div class="flex items-center gap-2 p-4 sticky top-0 z-10">
                    <img class="w-10 h-10 rounded-full object-cover" src="{{ other_user.profile.avatar }}" />
                    <div>
```

```
{% block javascript %}
<script>
document.querySelector('#chat_message_form').onsubmit = async (e) => {
    e.preventDefault();
    const formData = new FormData(e.target);
    const response = await fetch(e.target.action, {
        method: 'POST',
        body: formData
    });

    if (response.ok) {
        const data = await response.json();
        // Ajouter le message à l'interface utilisateur
        const messagesDiv = document.getElementById('chat_messages'); // Correction ici
        const newMessage = document.createElement('div');
        newMessage.classList.add('message');
        newMessage.innerHTML = `<strong>${data.author}</strong>: ${data.message} <span class="timestamp">${new Date().toLoc
        messagesDiv.appendChild(newMessage);
        e.target.reset(); // Réinitialiser le formulaire
    }
};
</script>
```

```
@login_required
def chatroom_delete_view(request, chatroom_name):
    chat_group = get_object_or_404(ChatGroup, group_name=chatroom_name)
    if request.user != chat_group.admin:
        raise Http404()

    if request.method == "POST":
        chat_group.delete()
        messages.success(request, 'Chatroom deleted')
        return redirect('home')

    return render(request, 'a_rtchat/chatroom_delete.html', {'chat_group':chat_group})


@login_required
def chatroom_leave_view(request, chatroom_name):
    chat_group = get_object_or_404(ChatGroup, group_name=chatroom_name)
    if request.user not in chat_group.members.all():
        raise Http404()
```

```python
class ChatGroup(models.Model):
    group_name = models.CharField(max_length=128, unique=True, blank=True)
    groupchat_name = models.CharField(max_length=128, null=True, blank=True)
    admin = models.ForeignKey(User, related_name='groupchats', blank=True, null=True, on_delete=models.SET_NULL)
    users_online = models.ManyToManyField(User, related_name='online_in_groups', blank=True)
    members = models.ManyToManyField(User, related_name='chat_groups', blank=True)
    is_private = models.BooleanField(default=False)

    def __str__(self):
        return self.group_name

    def save(self, *args, **kwargs):
        if not self.group_name:
            self.group_name = shortuuid.uuid()
        super().save(*args, **kwargs)


class GroupMessage(models.Model):
    group = models.ForeignKey(ChatGroup, related_name='chat_messages', on_delete=models.CASCADE)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    body = models.CharField(max_length=300, blank=True, null=True)
    file = models.FileField(upload_to='files/', blank=True, null=True)
    created = models.DateTimeField(auto_now_add=True)
```

```python
class TranslatorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("GILAN")

        self.recognizer = sr.Recognizer()
        self.mic = sr.Microphone()

        # Optimisations pour la reconnaissance vocale
        self.recognizer.dynamic_energy_threshold = False
        self.recognizer.energy_threshold = 300
        self.recognizer.pause_threshold = 0.1
        self.recognizer.non_speaking_duration = 0.1
        self.recognizer.operation_timeout = 5

        self.recording = False
        self.text = ""

        # Options pour la langue source et cible
        self.source_lang = tk.StringVar(value="fr")  # Langue source par défaut
        self.target_lang = tk.StringVar(value="sw")  # Langue cible par défaut

        # Ajout de menus pour sélectionner les langues
        tk.Label(root, text="Langue source:", font=("Helvetica", 14)).pack()
        source_menu = tk.OptionMenu(root, self.source_lang, "fr", "en")
        source menu.pack(pady=5)
```

```
class TranslatorApp:
    def __init__(self, root):
        # Ajout d'un label pour afficher les résultats de la transcription et de la traduction
        self.result_label = tk.Label(root, text="Cliquez sur le bouton pour commencer à écouter.", font=("Helvetica", 14))
        self.result_label.pack(pady=20)

        # Bouton pour démarrer l'écoute
        self.start_button = tk.Button(root, text="Commencer à écouter", command=self.start_listening, font=("Helvetica", 14
        self.start_button.pack(pady=10)

        # Bouton pour arrêter l'écoute
        self.stop_button = tk.Button(root, text="Arrêter l'écoute", command=self.stop_listening, font=("Helvetica", 14))
        self.stop_button.pack(pady=10)

        # Bouton pour traduire le texte
        self.translate_button = tk.Button(root, text="Traduire l'audio", command=self.translate_text, font=("Helvetica", 14
        self.translate_button.pack(pady=10)

    def start_listening(self):
        self.recording = True
        self.text = ""
        self.result_label.config(text="Écoute en cours...")
        threading.Thread(target=self.listen_loop, daemon=True).start()

    def stop_listening(self):
        # Planifie l'arrêt avec un délai de 3 millisecondes
        self.root.after(3, self.stop_listening_with_delay)

    def stop_listening_with_delay(self):
        self.recording = False
        self.result_label.config(text="Écoute arrêtée. Cliquez sur 'Traduire l'audio' pour traduire.")
```