



RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE



UNIVERSITÉ DES SCIENCES ET DE LA TECHNOLOGIE

HOUARI BOUMEDIENNE

FACULTÉ D'ÉLECTRONIQUE ET D'INFORMATIQUE

DÉPARTEMENT INFORMATIQUE

Mémoire de Master

Filière: Informatique

Spécialité : Systèmes Informatiques Intelligents

---

## **Construction d'un assistant vocal virtuel personnalisé supportant la langue arabe**

---

**Sujet Proposé et encadré par :**

- **M. ZELLAL Nassim**

**Soutenu le : 03/11/2020**

**Devant le jury composé de :**

- **Mme DOUKHA Zouina Présidente**
- **Mme BOUZIANE Nabila Membre**

**Présenté par :**

- **OUHOCINE Sarah**
- **SMAILI Ali**

PFE N° : SII / 044 / 2020



# REMERCIEMENTS



**A**u moment où s'achève le travail, il nous est gracieux d'adresser nos meilleures expressions de remerciements, de gratitude et de reconnaissances aux dignes.

Tout d'abord, gloire soit rendu au Dieu tout puissant créateur de toutes choses, le très miséricordieux pour toutes les bénédictions dont il nous a comblées et de nous avoir doté d'intelligence, de santé, de courage et de volonté pour aller jusqu'au bout de ce projet.

Nous tenons ensuite à exprimer notre gratitude à notre promoteur, Monsieur ZELLAL Nassim, pour son encadrement, sa patience et ses constantes orientations tout au long de ce projet en y accordant une méticuleuse attention, pour ses judicieux conseils et ses précieuses explications qui nous ont été indispensables à la conduite de ce projet, nous le remercions ainsi pour son extrême amabilité et sa totale disponibilité ; toujours à disposition pour nous guider dans notre projet malgré les difficiles conditions de travail et de communication suite à la crise sanitaire que traverse le monde en ce moment.

Qu'il trouve dans ce modeste travail un hommage vivant à sa haute personnalité.

Nous tenons ainsi, à remercier infiniment les membres de jury d'avoir consacré de leur précieux temps pour examiner notre travail malgré leur charge et quantité de travail, nous les remercions ainsi pour leur lecture attentive de notre mémoire, pour leur présence malgré les difficiles conditions causées par l'épidémie, ainsi que pour toutes les remarques qu'ils nous adresseront lors de notre soutenance afin d'améliorer notre travail.

Pourvu que ce travail soit à la hauteur de leurs attentes.

Nous sommes également très reconnaissants à tous nos enseignants du département informatique, en particulier ceux de la formation « Systèmes Informatiques Intelligents (SII) » ; nous vous remercions d'avoir partagé vos connaissances avec nous ; l'enseignement de qualité dispensé par ce master a su nourrir nos réflexions et a représenté une profonde satisfaction intellectuelle.

Enfin, nos remerciements s'adressent à toute personne, qui a contribué, de près ou de loin, à la réalisation de ce travail.



## DÉDICACES

Le présent travail est non seulement le résultat de notre patience, courage, sacrifice et force mais aussi une participation de plusieurs personnes qui nous sont chères, son succès dépend d'une série d'interventions tant morales qu'intellectuelles. De cette tribune-là, et avec le cœur plein de joie, je dédie chaleureusement ce travail à ...

*Ma très chère mère*, aimable, honorable, affable : Tu représentes pour moi le symbole de la charité par excellence, la source de tendresse et l'exemple du dévouement qui n'a jamais cessé de m'encourager et de prier pour moi, tes prières et tes bénédictions m'ont été d'un grand secours pour mener à bien mes études. Aucune dédicace ne saurait être assez éloquente pour exprimer ce que tu mèrites et ce que tu représentes pour moi. Je te dédie ce travail en témoignage de mon profond amour. Puisse Dieu, te préserver et t'accorder santé, longue vie et bonheur.

*À la mémoire de mon très cher père*, rien au monde ne vaut les efforts fournis jour et nuit pour mon éducation. Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation. Ça fait 5 ans que tu nous as quitté mais les conseils que tu m'avais donnés m'ont toujours accompagnés et m'ont été d'un grand secours pour mener à bien mes études. Aucune dédicace ne saurait exprimer l'amour, l'estime, le dévouement et le respect que j'ai toujours eu pour toi. Que dieu t'accueille dans son vaste paradis.

*Mon très cher frère Rabah*, présent dans tous les moments par son soutien moral, tu m'es le père, les mots ne suffisent guère pour exprimer l'attachement, l'amour et l'affection que je porte pour toi, Je te dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.

*Ma très chère soeur Dalila et son mari Ali*, en témoignage de l'attachement, de l'amour et de l'affection que je porte pour vous. Vous avez toujours été présents pour les bons conseils et le soutien moral. Je vous dédie ce travail avec tous mes vœux de bonheur, de santé et de réussite.

*Tous les membres de ma famille maternelle et paternelle, grands et petits,* particulièrement mes oncles et tantes et leur petite famille : Kamel, Youcef, Hakim, Mourad, Zina, Kahina et Malika, veuillez trouver dans ce modeste travail l'expression de mon affection.

*Mon cher binôme Ali*, avec lequel j'ai pris beaucoup de plaisir à travailler et sans lui rien n'aurait été pareil. Nous avons formé une belle équipe, je te dédie donc ce travail avec tous mes vœux de bonheur, de santé et de réussite.

*Ma chère cousine Karima*, une dédicace particulière à toi, une cousine qui m'est la sœur, j'aimerais te remercier pour ton soutien, ton amour, ton aide et ton encouragement à toute épreuve. Je te dédie donc ce modeste travail avec tous mes vœux de bonheur et de réussite.

*Ma chère et meilleure amie Yasmine*, encore une dédicace particulière, cette fois ci pour celle qui m'a ouvert son cœur et m'a offert la plus belle amitié qu'on puisse attendre d'un étranger, je voudrais te remercier pour tout ce que tu m'as apporté au cours de ces années partagées. Je te dédie donc ce modeste travail avec tous mes vœux de bonheur et de réussite.

*Mes cher(e)s ami(e)s et camarades*, j'espère que ce travail satisfera la confiance que vous avez en moi. Ma plus grande source d'inspiration aura été votre amour et votre grande attention psychologique qui n'ont cessé de me donner le sourire et m'ont poussé à continuer, en gardant la tête haute lorsque j'ai traversé des moments difficiles. Mes mots sont trop petits pour exprimer toute la gratitude que mon cœur contient pour vous et particulièrement Lilia, Asma, Manel, Mezhora et Nassim, sans oublier Oussama, Ali et Obada, vous êtes pour moi des frères, sœurs et des amis sur qui je peux compter. En témoignage de l'amitié qui nous unis et des souvenirs de tous les moments que nous avons passés ensemble, je vous dédie ce travail avec tous mes vœux de bonheur et de réussite.

*Mes cher(e)s enseignant(e)s et encadreurs de l'USTHB*, les plus grandes leçons ne sont pas tirées des livres mais des enseignants tels que vous. Sans vous, ce travail n'aurait jamais pu exister. Je vous dédie donc ce modeste travail en espérant d'avoir été à la hauteur.

*Je dédie ainsi ce travail, à toutes les personnes qui m'ont aidée ou encouragée de près ou de loin.*



Sarah



# DÉDICACES

Je tiens à remercier toute personne de mon entourage qui m'a encouragé et donné de l'énergie positive. Sans vous, je n'aurais jamais pu aller jusqu'au bout de ce travail. De cette tribune-là, et avec le cœur plein de joie, je dédie ce travail à ...

*À la mémoire de ma très chère mère*, celle qui a semé mes valeurs, qui m'a appris les premières lettres, qui m'a offert le premier livre, qui m'a toujours poussé à apprendre et à être assoiffé au savoir. J'en suis sûr que tu me regarde du paradis chère mère et que tu es fière de moi. C'est ton sang qui circule toujours dans mes veines et tous mes succès dans cette vie sont bien évidemment grâce à toi.

*Mon très cher père*, tes conseils étaient toujours ma source d'inspiration, tes sacrifices nous ont protégés de tout malheur. Tu es mon guide et mon exemple dans cette vie ; toutes les encres du monde ne seront pas suffisantes pour te remercier. J'espère que ce travail te rend fier de moi.

*Mon cher petit frère Ishak Salah El Din*, je ne trouve pas les mots pour exprimer l'énergie que tu me donnes, ton enthousiasme me pousse toujours en avant. Tes mots innocents et tes expressions d'encouragements ont énormément contribué à réaliser ce travail. J'espérerai d'être le bon exemple du grand frère.

*Toute ma famille*, oncles et tantes, cousins et cousines, grand et petit, plus particulièrement Billel, Redouane et Ahmed, merci d'être toujours là pour moi.

*Ma chère binôme Sarah*, sans toi ce travail n'aurait pas été possible, merci pour tous tes efforts, les nuits blanches et surtout pour les remarques que tu me faisais. Tu es la meilleure binôme et c'est toujours un grand plaisir de travailler avec toi.

*Mes chers amis et camarades,* Adel, Ahcen, Amine, Habib et Chaima, merci pour votre sincère amitié et votre soutien dans les moments les plus difficiles, vous étiez toujours là pour moi dans le meilleur et dans le pire. Je cite ainsi, Abdenour, Kenz, Taha et Oussama, sans oublier Ali, Obada et Oussama ; au nom de l'amitié et de tous moments que nous avons passés ensemble, je vous dédie donc ce travail. Vos conseils m'ont été utiles pour la réalisation de ce projet.

*Mes cher(e)s enseignant(s),* depuis mon premier jour à l'école, vous m'avez construit, non seulement avec des connaissances, mais surtout avec des valeurs, je cite, Mme Rezki, Mme Halladj, Mme Taflis et Mme Derradj. Ainsi, mes enseignants universitaires, M. Zellal, M. Bessa, Mme Nacer, Mme Rihani, Mme Bensaou, Mme Amrous, M. Gussoum et Mme Derias. Je vous dédie donc ce modeste travail en espérant d'avoir été à la hauteur.

*Je dédie ainsi ce travail, à toutes les personnes qui m'ont aidé ou encouragé de près ou de loin.*



Ali

# RÉSUMÉ

Depuis leur création, les ordinateurs ont toujours épargné aux humains de nombreuses tâches difficiles en fournissant des services plus rapides et plus efficaces. Mais avec l'apparition de l'Intelligence Artificielle (IA) et en particulier l'apprentissage automatique, nous voulons aller encore plus loin, et ce, en tentant de créer une entité pouvant imiter et simuler notre façon de raisonner, produire et interagir avec l'information.

Il est clair qu'aujourd'hui nous avons atteint ces objectifs. En effet, dans certains domaines spécifiques, les récents progrès de l'IA ont même réussi à rendre l'ordinateur capable de dépasser le raisonnement humain et ont ouvert ainsi, un nouvel angle de voir les choses. Néanmoins, l'un des domaines où l'ordinateur est encore loin d'atteindre le comportement humain est la communication ; étant donné que cette dernière dépend fortement des langages naturels, qui sont souvent assez complexes et délicats.

La langue arabe est l'une des langues les plus utilisées. Cette langue a beaucoup de spécificités et de particularités, ce qui rend la construction d'un système intelligent traitant cette langue-là, très difficile et pas évident.

À travers ce projet, nous nous familiarisons avec les systèmes d'assistance vocale. En tirant partie des dernières recherches et approches avancées en intelligence artificielle, allant des modèles de reconnaissance vocale aux modèles de synthèse vocale, en passant notamment par la compréhension et le traitement automatique du langage naturel ; notre assistant est capable d'aider les utilisateurs et les locuteurs de cette langue, à communiquer avec leur appareil et à accomplir aisément leurs tâches ordinaires par le biais de directives vocales.

**Mots clés :** Intelligence Artificielle, Traitement Automatique du Langage Naturel, Apprentissage Profond, Reconnaissance Vocale, Assistant Vocal Intelligent, Langue Arabe.

# ABSTRACT

Since their inception, computers have always saved humans many difficult tasks by providing faster and more efficient services. But with the advent of Artificial Intelligence (AI), and in particular machine learning, we want to go even further, by trying to create an entity that can imitate and simulate the way we reason, produce and interact with the information.

It is clear that today we have achieved these goals, such as in some specific areas, recent advances in AI have even succeeded in making the computer capable of going beyond human reasoning and thus opening up a new angle to see things. However, one of the areas where the computer is still far from reaching human behavior is communication. Indeed, the latter strongly depends on natural languages, which are often quite complex and delicate.

The Arabic language is one of the most widely used languages. This language has many specificities and peculiarities, which makes the construction of an intelligent system dealing with this language, very difficult and not easy.

Through this project, we become familiar with voice assistance systems. By taking advantage of the latest research and advanced approaches in artificial intelligence, ranging from speech recognition models to speech synthesis models, including the understanding and automatic processing of natural language; our assistant is able to help users and speakers of this language to communicate with their device and to easily perform their ordinary tasks through voice instructions.

**Keywords :** Artificial Intelligence, Natural Language Processing, Deep Learning, Speech Recognition, Intelligent Voice Assistant, Arabic Language.

## ملخص

منذ نشأتها، لطالما استطاعت أجهزة الكمبيوتر توفير العديد من المهام الصعبة على البشر، وهذا من خلال تقديمها لخدمات أسرع وأكثر كفاءة. لكن مع ظهور الذكاء الاصطناعي، وخاصة التعلم الآلي، نريد الذهاب إلى أبعد من ذلك، وهذا من خلال محاولة إنشاء كيان يمكنه محاكاة وتقليد طريقة تفكيرنا وإنتاجنا وتفاعلنا مع المعلومات.

من الواضح أننا اليوم حققنا هذه الأهداف، حيث أنه في بعض المجالات، تمكّنت التطورات الحديثة في الذكاء الاصطناعي من جعل الكمبيوتر قادرًا حتى على تجاوز التفكير البشري، وبالتالي فتح زاوية جديدة لرؤية الأشياء. رغم ذلك، فإنه لا تزال بعض المجالات التي تبقى فيها أجهزة الكمبيوتر بعيدة عن الوصول إلى السلوك البشري، خاصة مجال الاتصال، كون هذه الأخيرة تعتمد بشدة على اللغات الطبيعية، والتي غالباً ما تكون معقدة وحسّاسة للغاية.

تعتبر اللغة العربية من إحدى اللغات الأكثر استخداماً، فهي تملك العديد من الميزات والخصائص، مما يجعل برمجة نظام ذكي للتعامل مع هذه اللغة صعباً جدًا وليس بالأمر السهل.

من خلال هذا المشروع، تعرّفنا على أنظمة المساعدة الصوتية. من خلال الاستفادة من أحدث الأبحاث والأساليب المتقدمة في الذكاء الاصطناعي، بدءاً من نماذج التعرف على الكلام، إلى نماذج تركيب الكلام، بما في ذلك الفهم والمعالجة التلقائية للغة الطبيعية ؟ فإن مساعدنا الصوتي قادر على مساعدة مستخدمي ومحبي اللغة العربية على التواصل مع أجهزتهم وأداء مهامهم الاعتيادية بسهولة بواسطة التعليمات الصوتية.

**الكلمات المفتاحية :** الذكاء الاصطناعي، المعالجة التلقائية للغة الطبيعية، التعلم العميق، التعرف التلقائي على الكلام، مساعد صوتي ذكي، اللغة العربية.

# TABLE DES MATIÈRES

TABLE DES FIGURES.....	IV
LISTE DES TABLEAUX .....	VI
LISTE DES ÉQUATIONS .....	VII
LISTE DES ALGORITHMES .....	VIII
<b>INTRODUCTION GÉNÉRALE.....</b>	<b>1</b>
Contexte .....	1
Problématique.....	2
Objectifs et contributions .....	2
Organisation du mémoire .....	3
<b>1 CHAPITRE I GÉNÉRALITÉS.....</b>	<b>5</b>
1.1 C'est quoi un assistant vocal virtuel ? .....	5
1.2 Les assistants virtuels et les smartphones.....	5
1.3 Comment fonctionne un assistant vocal virtuel ? .....	6
1.3.1 La reconnaissance vocale .....	7
1.3.1.1 Le déclenchement du système par la voix (Wake-up Word) .....	7
1.3.1.2 La capture et la retranscription de la voix en texte (Speech-to-Text).....	7
1.3.2 Le traitement automatique du langage naturel .....	8
1.3.2.1 Les niveaux de traitement .....	9
1.3.2.2 La langue arabe.....	10
1.3.2.3 Les caractéristiques et particularités de la langue arabe .....	11
1.3.2.4 La morphologie de la langue arabe.....	18
1.3.2.5 La syntaxe de la langue arabe .....	20
1.3.2.6 Les difficultés du TALN Arabe.....	22
1.3.3 La synthèse vocale .....	25
1.4 À quoi sert l'assistant vocal ? .....	25
1.5 Conclusion.....	26
<b>2 CHAPITRE II ÉTAT DE L'ART .....</b>	<b>27</b>
2.1 Analyse des articles connexes.....	27
2.1.1 Approches pour la reconnaissance vocale.....	27
2.1.1.1 Approches symboliques .....	28
2.1.1.2 Deep learning (apprentissage profond).....	29
2.1.2 Approches pour le traitement automatique du langage naturel .....	31
2.1.2.1 Reconnaissance de l'acte de dialogue.....	31
2.1.2.2 Extraction d'information.....	34
2.1.2.3 Génération de réponses .....	40

2.1.2.4	Création d'une base de connaissances .....	47
2.1.3	Approches pour la synthèse vocale.....	48
2.1.3.1	Analyse de texte .....	48
2.1.3.2	La synthèse de forme d'onde (Waveform synthesis) .....	49
2.2	Analyse de l'existant .....	50
2.2.1	Revue de quelques applications d'assistance vocale disponibles sur le marché.	50
2.2.1.1	Assistant vocal de Google : Google Assistant, le plus intelligent .....	50
2.2.1.2	Assistant vocal d'Amazon : Alexa, la meilleure compatibilité .....	51
2.2.1.3	Assistant vocal d'Apple : Siri, le meilleur pour l'écosystème Apple.....	52
2.2.1.4	Assistant vocal de Microsoft : Cortana, le meilleur pour les pros.....	53
2.2.1.5	Assistant vocal arabe : RafiQ .....	54
2.2.1.6	Autres assistants vocaux virtuels en développement.....	54
2.2.2	Critères pour choisir un assistant vocal.....	55
2.2.3	La solution de l'assistant vocal proposé « FASSIHA ».....	56
2.2.3.1	Comparatif des assistants vocaux existants et FASSIHA .....	56
2.2.3.2	Description de l'assistant vocal FASSIHA, le meilleur pour tous .....	57
2.3	Conclusion.....	57
<b>3</b>	<b>CHAPITRE III CONCEPTION .....</b>	<b>58</b>
3.1	Architecture globale de l'assistant vocal « FASSIHA » .....	58
3.2	L'interface utilisateur (Front-end).....	60
3.2.1	L'interface graphique .....	60
3.2.2	Interface de capture audio (M1) .....	60
3.2.3	Interface de réponse audio (M7).....	61
3.2.4	Architecture et conception de l'interface utilisateur (Front-end) .....	61
3.3	Le système intelligent d'arrière-plan (Back-end) .....	63
3.3.1	Module de reconnaissance vocale (M2).....	63
3.3.2	Module de TALN structuration des requêtes (M3) .....	65
3.3.3	Module de classification des requêtes (M4).....	69
3.3.3.1	Classifieur de niveau 0.....	69
3.3.3.2	Classifieur de niveau 1.....	70
3.3.3.3	Classifieur de niveau 2.....	73
3.3.4	Module de génération de réponses (M5).....	73
3.3.4.1	Générateur de réponses de niveau 0 .....	74
3.3.4.2	Générateur de réponses de niveau 1 .....	74
3.3.4.3	Générateur de réponses de niveau 2 .....	74
3.3.5	Module de synthèse vocale (M6).....	78
3.4	Conclusion.....	79

<b>4 CHAPITRE IV IMPLÉMENTATION ET ÉVALUATION .....</b>	<b>80</b>
4.1 Environnement de développement.....	80
4.1.1 Environnement matériel.....	80
4.1.2 Environnement logiciel .....	81
4.1.2.1 Plateformes.....	82
4.1.2.2 Langages de programmation .....	82
4.1.2.3 Bibliothèques et librairies .....	83
4.1.2.4 Outils de développement.....	86
4.2 Module de reconnaissance vocale.....	87
4.2.1 Expérimentations et évaluation .....	89
4.2.1.1 Données expérimentales (ensemble de test) .....	89
4.2.1.2 Métriques d'évaluation .....	89
4.2.1.3 Résultats et discussion .....	90
4.3 Module de TALN et structuration des requêtes.....	91
4.4 Module de classification des requêtes.....	92
4.4.1 Classifieur de niveau 0 .....	92
4.4.2 Classifieur de niveau 1 .....	93
4.4.3 Classifieur de niveau 2 .....	94
4.5 Module de génération de réponses.....	94
4.5.1 Générateur de réponses de niveau 0 .....	95
4.5.2 Générateur de réponses de niveau 1 .....	95
4.5.3 Générateur de réponses de niveau 2 .....	96
4.5.3.1 Expérimentations et évaluation du modèle d'apprentissage .....	104
4.6 Module de synthèse vocale.....	108
4.7 Application Fassiha.....	109
4.7.1 Le système intelligent d'arrière-plan (Back-end).....	110
4.7.2 L'interface utilisateur (Front-end) .....	111
4.7.3 Expérimentations et évaluation générale de Fassiha .....	116
4.7.3.1 Données expérimentales (ensemble de test) .....	116
4.7.3.2 Métriques d'évaluation .....	116
4.7.3.3 Résultats et discussion .....	118
4.8 Conclusion.....	122
<b>CONCLUSION GÉNÉRALE ET PERSPECTIVES .....</b>	<b>123</b>
<b>BIBLIOGRAPHIE .....</b>	

# TABLE DES FIGURES

1.1	Logo d'un AVV .....	5
1.2	Assistant vocal sur smartphone .....	5
1.3	Schéma de base du fonctionnement d'un assistant vocal virtuel .....	6
1.4	Les AVV et L'IA .....	6
1.5	La reconnaissance vocale .....	8
1.6	Les 19 formes des lettres arabes .....	12
1.7	Les 10 marques des lettres arabes .....	12
1.8	Les 28 lettres arabes de base .....	13
1.9	Les diacritiques .....	14
1.10	Les voyelles longues .....	15
1.11	Prononciation, appellation et positionnement des diacritiques et voyelles longues sur la lettre “ـ ” .....	15
1.12	Mots avec et sans Tatweel .....	16
1.13	Mots arabes écrits en différentes topographies .....	16
1.14	Les pronoms personnels de la langue arabe .....	16
1.15	Structure d'un mot dans la langue arabe (mot maximal) .....	17
1.16	Transducteur à états finis montrant la syntaxe de base d'une phrase verbale arabe ...	21
1.17	Transducteur à états finis montrant la syntaxe de base d'une phrase nominale arabe .....	21
1.18	Transducteur à états finis montrant la syntaxe de base d'une phrase locative arabe .....	22
1.19	Quelques services fournis par les assistants vocaux virtuels .....	26
2.1	Exemple d'un MLP .....	33
2.2	Étiquetage POS d'une phrase .....	36
2.3	Extraction d'entités nommées d'une phrase .....	36
2.4	Arbre d'analyse de HVS .....	38
2.5	Exemple de Seq2Seq RNN .....	44
2.6	Logo de Fassiha .....	57
3.1	Architecture globale du système « FASSIHA » .....	59
3.2	Architecture MVVM .....	62
3.3	Image du livre sacré (corpus coranique) .....	64
3.4	Architecture du module de reconnaissance vocale .....	65

3.5	Document XML associé à la requête افتح تطبيق الطقس.....	68
3.6	Extrait de la base de règles associée aux requêtes de niveau 0.....	70
3.7	Caractéristiques du dataset associé à l'application Météo.....	71
3.8	Termes non composés du dataset associé à l'application Météo .....	71
3.9	Termes composés du dataset associé à l'application Météo .....	71
3.10	Applications affectées à une requête de niveau 1 avant et après stemming .....	72
3.11	Transformateur de notre réseau deep learning .....	75
3.12	Corpus de questions-réponses arabes factoïdes TALAA-AFAQ .....	76
3.13	La nouvelle structure du corpus questions-réponses factoïdes arabes TALAA-AFAQ .....	76
3.14	Architecture du transformateur utilisé .....	77
3.15	Architecture du module de synthèse vocale .....	79
4.1	Caractéristiques des machines .....	80
4.2	Caractéristiques des appareils de test .....	81
4.3	Logos des différents logiciels utilisés.....	81
4.4	Boîte de dialogue demandant une entrée vocale .....	88
4.5	Histogramme montrant la variation du WER en fonction de 3 modèles de reconnaissance vocale .....	90
4.6	Analyse Bottom-up sur un exemple de requête de niveau 0 .....	93
4.7	Exemple d'encodage des tokens .....	100
4.8	Variation du taux d'apprentissage en fonction du pas d'apprentissage .....	103
4.9	Graphe de variation de la précision du modèle.....	108
4.10	Graphe de variation du taux d'erreur du modèle .....	108
4.11	Requête sous format JSON .....	111
4.12	Lanceur de l'application Fassiha.....	112
4.13	Activités de l'application Fassiha .....	112
4.14	Requête de niveau 0 et sa réponse.....	114
4.15	Lancement de l'application Météo (à Alger) .....	114
4.16	Requête de niveau 1 et sa réponse.....	115
4.17	Lancement de l'application Météo (météo de demain) .....	115
4.18	Requête de niveau 2 et sa réponse.....	116
4.19	VP, VN, FP, FN, Rappel et Précision .....	117
4.20	Histogramme montrant la classification des requêtes (VP, VN, FP, FN) en fonction des niveaux (0, 1, 2) .....	118
4.21	Histogramme montrant les valeurs de la précision, du rappel et de la F-Mesure en fonction des niveaux (0, 1, 2).....	119
4.22	Histogramme montrant le nombre de réponses pertinentes et non pertinentes en fonction des niveaux (0, 1, 2).....	121

# LISTE DES TABLEAUX

1.1	Structuration du mot ..... أستنذكرونها	18
1.2	Racines trilitères des mots de la phrase “ <i>إستغم الأعرابي ناقة من الحروب</i> ” ..... ” <i>إستغم الأعرابي ناقة من الحروب</i> “	18
1.3	Quelques modèles de schèmes appliqués à la racine (كتب/KTB/écrire) ..... كتب	19
1.4	Lemmes des mots de la phrase ‘ <i>إستغم الأعرابي ناقة من الحروب</i> ’ ..... ’ <i>إستغم الأعرابي ناقة من الحروب</i> ‘	19
1.5	Quelques exemples de phrases verbales arabes ayant différentes syntaxes .....	21
1.6	Ambiguïté du mot « <i>بين</i> » causée par l'absence de diacritiques .....	22
1.7	Ambiguïté morphologique du mot « <i>فصل</i> » ..... فصل	23
1.8	Ambiguïté morphosyntaxique de mot « <i>كتب</i> » ..... كتب	23
1.9	Ambiguïté 1 de lemmatisation .....	24
1.10	Ambiguïté 2 de lemmatisation .....	24
2.1	Comparatif des assistants vocaux existants et Fassiha .....	56
4.1	Exemple d'arguments tirés et d'applications lancées à partir des requêtes de niveau 1 .....	95
4.2	Résultats d'apprentissage avec une seule couche de transformateur .....	105
4.3	Résultats d'apprentissage avec une seule couche de transformateur et un taux de dropout de 0.1 .....	106
4.4	Résultats d'apprentissage avec deux couches de transformateur et un taux de dropout de 0.1 .....	107
4.5	Guide d'utilisation de l'application Fassiha .....	113
4.6	Tableau montrant la classification des requêtes (VP, VN, FP, FN) en fonction des niveaux (0, 1, 2) .....	118
4.7	Tableau montrant les valeurs de la précision, du rappel et de la F-Mesure en fonction des niveaux (0, 1, 2) .....	119
4.8	Tableau montrant le nombre de réponses pertinentes et non pertinentes en fonction des niveaux (0, 1, 2) .....	120

# LISTE DES ÉQUATIONS

2.1	Fonction de décodage d'un mot (règle de Bayes) .....	28
2.2	Fonction pour trouver la séquence d'actes de dialogue avec la plus forte probabilité de se produire.....	32
2.3	Fonction pour trouver la séquence d'actes de dialogue avec la plus forte probabilité de se produire si les mots d'un énoncé sont connus et les hypothèses d'indépendance des mots naïfs de Bayes subsistent .....	32
2.4	Fonction de coût d'entropie croisée .....	33
2.5	Fonction qui calcule la probabilité qu'un ensemble de mots W se produise avec un ensemble de concepts C et un ensemble d'opérations de transition N .....	39
2.6	Fonction d'entraînement du modèle de He et Young .....	39
2.7	Fonction de classement global des documents .....	43
2.8	Fonction de māj de l'état caché du RNN.....	45
2.9	Fonction cachée du décodeur.....	45
2.10	Fonction pour maximiser la probabilité de production d'une séquence de réponse étant donné une séquence d'entrée .....	45
2.11	Fonction pour maximiser la probabilité conditionnelle log .....	46
3.1	Formule de calcul de l'attention.....	75
3.2	Formules d'encodage de position .....	77
4.1	La métrique WER pour l'évaluation des systèmes de reconnaissance vocale .....	89
4.2	Formule de calcul de la précision .....	121

# LISTE DES ALGORITHMES

4.1	Démarrage du processus de reconnaissance vocale arabe .....	88
4.2	Extrait de la base de règles associée au classifieur de niveau 0.....	92
4.3	Lancement d'une application externe sous Android.....	95
4.4	Préparation de l'environnement pour l'apprentissage .....	96
4.5	Construction du Tokenizer .....	97
4.6	Calcul de l'attention du produit scalaire .....	98
4.7	Implémentation de la couche d'attention .....	98
4.8	Implémentation des couches d'encodeur et de décodeur .....	100
4.9	Implémentation du transformateur .....	101
4.10	Implémentation de la fonction Loss .....	103
4.11	Implémentation de la fonction du taux d'apprentissage personnalisé .....	103
4.12	Lancement de l'apprentissage.....	104
4.13	Lancement du service Text To Speech de Google Cloud .....	109

# INTRODUCTION GÉNÉRALE

## Contexte

Nous sommes dans une ère où l'usage des différents dispositifs électroniques ne cesse de croître, particulièrement les appareils mobiles tels que les smartphones, smartwatches, etc.

En 2018, les statistiques estiment qu'il y a environ 5 milliards d'utilisateurs de smartphones [1], la majorité d'entre eux sont connectés à Internet [2] et génèrent une énorme quantité de données. Avec cette puissante dépendance aux appareils, l'utilisateur cherche de plus en plus des applications personnalisées et ergonomiques, qui l'aident et facilitent au maximum ses tâches. Dès l'apparition du premier appareil mobile déjà, les utilisateurs ont voulu l'utiliser et exploiter ses différentes fonctionnalités sans avoir besoin de faire le moindre geste, comme cliquer sur un bouton, taper un texte à partir du clavier et cela juste en effectuant une discussion avec l'appareil.

Avant le développement de l'informatique ceci paraissait magique et invraisemblable, mais après le développement de l'informatique et surtout avec l'apparition de l'**intelligence artificielle** (IA) et l'apprentissage automatique, l'idée de tenir une conversation significative avec un appareil mobile semblait futuriste et novatrice.

Il y a quelques décennies, la technologie a permis aux appareils de reconnaître la voix de l'utilisateur. Dans ce contexte, plusieurs produits de consommation développés ont introduit des applications dédiées à la reconnaissance vocale et à la compréhension linguistique [3], comme les assistants vocaux virtuels (les fameux Siri d'Apple, Google Assistant de Google et Alexa d'Amazon, etc. [4]). Ces applications assistent l'utilisateur à organiser les tâches de sa vie quotidienne, rechercher des informations de façon rapide et fiable, lancer des applications et des tâches sur ces appareils, et ce, uniquement en utilisant la voix et la parole et allant même jusqu'à avoir de réelles conversations avec l'assistant virtuel.

De telles applications sont connues pour leur haute personnalisation et grande adaptation à l'utilisateur, mais la **langue** reste un élément clé pour pouvoir décider ou juger ces assistants vocaux, autrement dit la qualité de l'assistant vocal dépend inéluctablement de la langue qu'il supporte.

## Problématique

Les solutions existantes pour certaines langues, qui utilisent l'alphabet latin, comme le français ou l'anglais, donnent de bons résultats malgré les ambiguïtés sémantiques. Cependant, certaines langues, comme l'arabe, sont extrêmement complexes à traiter [5] à cause de la quantité d'ambiguïtés, la formation linguistique de ses signes diacritiques, etc. D'ailleurs le chercheur et linguiste informatique américain **Kenneth R. Beesley** a dit dans ce contexte : « *La règle générale pour l'arabe est que tout est au moins cinq fois plus compliqué que pour n'importe quelle langue européenne* ». De ce fait, les assistants vocaux ne supportent pas l'arabe ou bien ont un faible support de cette langue. Ainsi, comment un smartphone peut-il supporter la langue arabe et résoudre ses ambiguïtés ?

Lorsque nous parlons d'intelligence et d'efficacité, le domaine de l'intelligence artificielle (**IA**) et l'apprentissage automatique donnent bien souvent de bons résultats pour résoudre ce type de problème.

Notre smartphone serait donc capable de reconnaître la voix humaine, de comprendre la langue arabe, de répondre, d'interagir avec l'utilisateur et de réduire les erreurs dans la mesure du possible, alors qu'il ne possède ni organes, ni cerveau, ni autre source d'intelligence, de sentiments ou de réflexion.

Ainsi avec l'augmentation de la population arabe et l'utilisation excessive des smartphones par les nouvelles générations arabes, la langue arabe a pu se répandre et se propager d'une façon très rapide, de ce fait, des corpus vocaux et textuels et des outils sont construits spécialement pour les traitements délicats de la langue arabe. Devant ce constat, la mise en place d'un assistant vocal virtuel personnalisé supportant la langue arabe intelligent et efficace s'impose. Le terrain est donc prêt pour relever le défi du développement d'un assistant vocal virtuel, supportant la langue arabe.

## Objectifs et contributions

Pour ce faire, nous proposons **un assistant vocal virtuel personnalisé supportant la langue arabe**. Un tel assistant permet à l'utilisateur de passer depuis son smartphone.

- Des commandes vocales de base (partie exécution directe) : où l'utilisateur peut d'une façon explicite ou implicite régler l'alarme, passer des appels, lancer des applications et

des jeux, écouter de la musique, ajuster la luminosité, etc. Cette partie explorera le fonctionnement de base et les caractéristiques communes des assistants vocaux.

- Des commandes vocales avancées (partie apprentissage) : où l'utilisateur peut effectuer une conversation réelle avec son appareil, lui poser des questions pour avoir des informations ou juste pour s'amuser (une sorte de divertissement). C'est cette partie, qui va donner une touche personnelle à notre propre assistant virtuel.

Nous allons dans le cadre de ce projet de fin d'études réaliser le système décrit plus haut, autrement dit nous allons réaliser un système intelligent du début jusqu'à la fin : le sous-système intelligent de base et le sous-système intelligent avancé.

Notre système est bien évidemment personnalisé et adaptatif quelle que soit la requête de l'utilisateur. Il est ainsi muni d'une application mobile facile d'utilisation, pouvant fournir la meilleure expérience utilisateur.

Ce projet se veut assez ambitieux et vise à faciliter l'interaction entre l'homme et son smartphone et consiste à comprendre l'intention humaine, puis prendre les décisions qui conviennent mieux à l'utilisateur d'une manière intelligente, en explorant la collecte et l'extraction de données à partir de l'interaction de l'historique de l'utilisateur avec l'application.

## Organisation du mémoire

Ce mémoire est organisé en 4 chapitres :

Dans le chapitre 1, nous commençons par introduire les assistants vocaux virtuels et leur fonctionnement, pour nous attarder par la suite sur les notions fondamentales de la langue arabe.

Dans le chapitre 2, nous analysons dans un premier temps les articles connexes, les recherches essentielles et les travaux intéressants qui ont été réalisés dans ce domaine, d'une part pour ne pas être en marge du progrès de l'assistance vocale, d'autre part, pour adopter les meilleures méthodes et approches et les implémenter par la suite dans notre propre assistant vocal. Dans un second temps, nous analysons l'existant ou nous passons en revue quelques applications d'assistance vocale similaires existantes sur le marché, en essayant de détecter les points forts

et les points faibles de chacune, que nous prendrons en considération pour concevoir notre assistant vocal. Nous dégagons enfin la description de notre solution qui reprend les forces et comble au mieux les lacunes des solutions existantes.

Dans le chapitre 3, nous nous penchons sur la conception. Nous présentons notre contribution personnelle en dévoilant l'architecture détaillée de notre assistant vocal et notre apport par rapport aux autres architectures.

Quant au chapitre 4 (dernier chapitre) ; il est consacré à l'implémentation et l'évaluation. Nous présentons tout d'abord les outils et environnement de travail nécessaire au développement et au déploiement. Nous détaillons par la suite les grandes lignes de la mise en œuvre du système avec ses deux aspects : le sous-système intelligent de base et le sous-système intelligent avancé, puis, nous terminons ce chapitre par la présentation de l'interface utilisateur de l'application mobile ainsi que les expérimentations et les résultats obtenus. Cela permet à l'utilisateur d'évaluer notre système et ce, en effectuant plusieurs tests, puis comparer les résultats retournés par notre assistant vocal virtuel avec les résultats attendus par l'utilisateur, en tenant compte du temps de réponse.

Chapitre

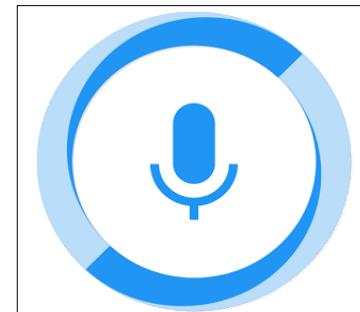
# 1 GÉNÉRALITÉS

Ce chapitre se veut introductif, présentant le cadre général de notre projet, nous définissons d'abord les assistants vocaux virtuels, leur fonctionnement et architecture de base, leur usage ainsi que leur importance dans la vie quotidienne, pour ensuite entrer dans le vif du sujet et mettre en évidence les notions importantes de la langue arabe qui devront être prises en compte dans la réalisation d'un bon assistant vocal supportant la langue arabe.

## 1.1 C'est quoi un assistant vocal virtuel ?

Un assistant vocal virtuel (AVV), aussi appelé « assistant personnel intelligent », est une application logicielle qui comprend les commandes vocales en langage naturel et accomplit les tâches pour l'utilisateur ou plus précisément c'est un agent logiciel basé sur la reconnaissance vocale du langage naturel, le traitement de ce dernier puis la restitution d'informations par synthèse vocale, ces agents logiciels sont ensuite intégrés dans des appareils de haut-parleurs, des ordinateurs ou des smartphones dédiés afin de faciliter l'interaction entre l'utilisateur et ses dispositifs.

Lorsque ces assistants vocaux virtuels sont connectés à Internet, ils sont capables d'interpréter la parole humaine et se chargent de répondre à toutes les questions des utilisateurs par des voix synthétisées. Ils peuvent également effectuer des tâches, services, dialogues, commandes, etc.

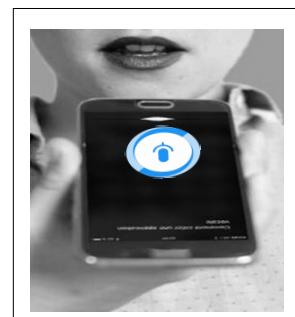


**Figure 1.1**  
Logo d'un  
AVV

## 1.2 Les assistants virtuels et les smartphones

L'assistant vocal virtuel est devenu une réalité surtout sous l'impulsion des dispositifs tactiles et des objets connectés particulièrement les smartphones.

Ces assistants qui se présentent comme une alternative ou un complément intéressant ont apporté une véritable révolution sur le marché de la haute technologie (high-tech).



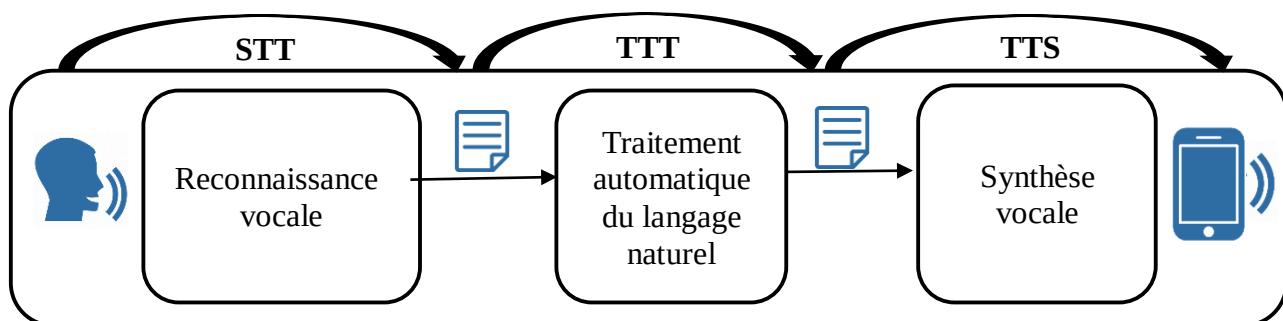
**Figure 1.2**  
Assistant vocal  
sur smartphone

### 1.3 Comment fonctionne un assistant vocal virtuel ?

Le fonctionnement des assistants vocaux virtuels comprend trois étapes primordiales [6].

- 1) La reconnaissance vocale / RV (Speech-To-Text / STT).
- 2) Le traitement automatique du langage naturel / TALN (Text-To-Text / TTT).
- 3) La synthèse vocale / SV (Text-To-Speech / TTS).

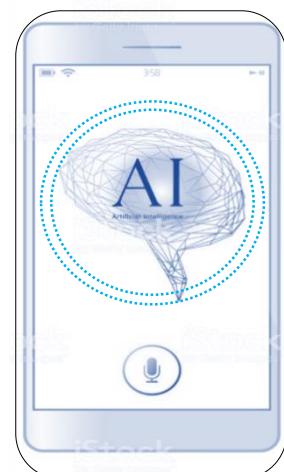
Ces étapes permettant d'apporter des réponses orales aux requêtes des utilisateurs, le logiciel attend constamment l'écoute d'un mot clé pour s'activer. Une fois le mot clé identifié, le logiciel enregistre la voix de l'utilisateur et l'envoie au serveur spécialisé de traitement de la langue, qui traite et interprète cette voix comme une commande et suivant cette commande, le serveur fournira à l'assistant vocal les informations appropriées et adéquates aux requêtes de l'utilisateur, à lire à ce dernier par synthèse vocale.



**Figure 1.3** Schéma de base du fonctionnement d'un assistant vocal virtuel

Ce processus, de son début jusqu'à sa fin, semble assez intelligent, mais cela ne relève pas de la magie, mais plutôt de **l'intelligence artificielle (IA)**, qui utilise des algorithmes sophistiqués de **l'apprentissage automatique (AA)**.

L'assistant vocal étant un programme entièrement intelligent, il est associé à une forme d'intelligence artificielle grâce à laquelle il apprend à connaître les habitudes et les préférences de l'utilisateur. En effet, au fur et à mesure que l'utilisateur interagit avec l'assistant virtuel, la programmation de l'IA utilise des algorithmes sophistiqués et des apprentissages automatiques pour apprendre de l'entrée et de la saisie des données et mieux prédire les besoins de l'utilisateur et essayer par la suite de lui proposer certaines recommandations, des rappels d'événements, etc. Pour ce faire, le logiciel a besoin d'accéder à une grande quantité de données concernant l'utilisateur comme l'historique Internet, la géolocalisation, etc.



**Figure 1.4**  
Les AVV et L'IA

Certains assistants sont construits avec des technologies informatiques cognitives<sup>1</sup> [7] plus avancées qui permettront à un assistant de comprendre et d'exécuter des demandes en plusieurs étapes avec de nombreuses interactions et d'effectuer des tâches plus complexes, telles que la réservation de sièges dans une salle de cinéma ou la réservation d'un billet d'avion.

### 1.3.1 La reconnaissance vocale

La reconnaissance vocale (RV), aussi appelée « reconnaissance automatique de la parole » est une technique permettant d'analyser et de capter la voix humaine par le biais d'un microphone, afin de la transcrire en texte exploitable par la machine.

La RV se divise en deux grandes parties, le déclenchement du système par la voix (Wake-up Word) et la capture et la retranscription de la voix en texte (Speech-to-Text) [8].

#### 1.3.1.1 Le déclenchement du système par la voix (Wake-up Word)

C'est la première étape de la reconnaissance vocale, il s'agit de réveiller l'assistant vocal par un mot clé prononcé par l'utilisateur. Le but principal de cette étape est en fait d'activer le système et de le prévenir qu'il va recevoir une commande vocale.

#### 1.3.1.2 La capture et la retranscription de la voix en texte (Speech-to-Text)

Une fois la reconnaissance vocale initiée grâce au Wake-up Word, il est nécessaire d'exploiter la parole. Cela en l'enregistrant puis la numérisant via le Speech-to-Text. Pour ce faire, la voix de l'utilisateur est captée en fréquences sonores, i.e. sous forme de fichiers audios comme la musique ou tout autre bruit, puis plusieurs traitements seront faits dans le but d'améliorer l'enregistrement de ces fréquences.

- **La normalisation** : éliminer les pics et les creux dans les fréquences pour équilibrer l'ensemble et le mettre en harmonie.
- **La réduction des bruits de fond** : diminuer les bruits pour une qualité audio meilleure.
- **L'Analyse phonologique** : ou le découpage en phonèmes, il s'agit de découper les segments en unités distinctives et spécifiques au sein des fréquences, exprimées en millième de seconde, permettant la distinction des mots les uns des autres, puis les

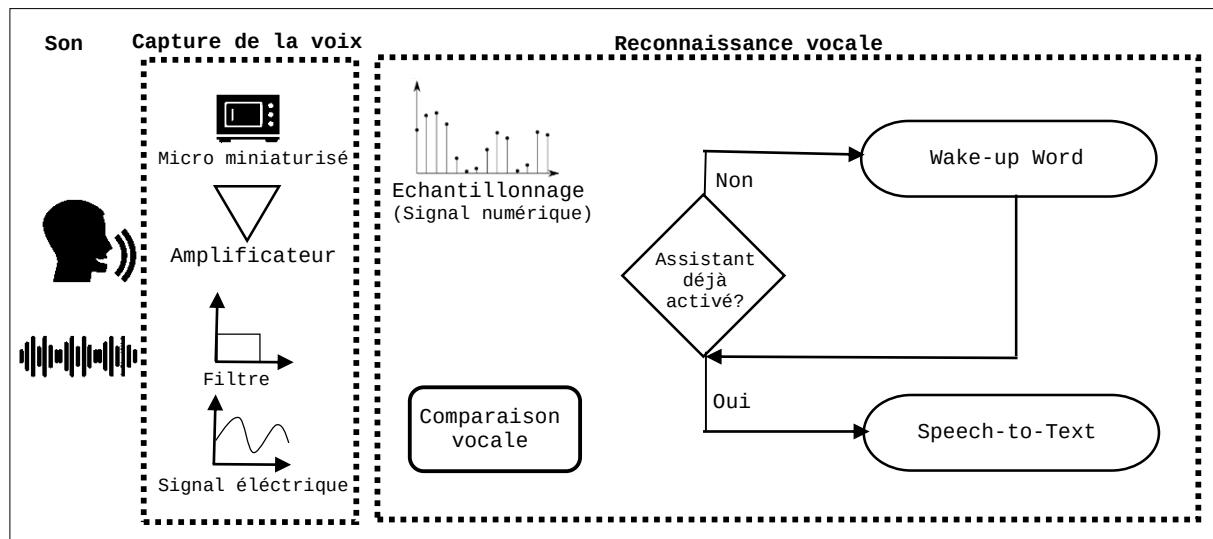
---

<sup>1</sup> La simulation du processus de pensée humaine dans un modèle informatique. Cette technologie repose sur des systèmes d'auto-apprentissage qui utilisent l'exploration de données (data mining), la reconnaissance des schémas et le TALN pour tenter de reproduire le mode de fonctionnement du cerveau humain, dans le but de résoudre des problèmes sans l'intervention humaine.

fréquences, une fois enregistrées, peuvent être analysées pour associer à chaque phonème un mot ou un groupe de mots afin de produire un texte.

En résumé, nous pouvons récapituler la reconnaissance vocale comme suit.

- L'utilisateur parle à son assistant vocal, le son de la voix est capté par les micros miniaturisés, puis est amplifié, filtré et transformé en signal électrique analogique propre.
- Le signal électrique analogique est transformé en signal numérique (échantillonnage), la comparaison du signal donc avec des phrases préenregistrées est alors possible.
- La partie électronique de système agit en fonction des cas suivants.
  - Si l'assistant n'est pas activé, la phase d'activation (Wake-up Word) est détectée.
  - Si l'assistant est déjà actif, le système retranscrit de la voix en texte (Speech-to-Text).



**Figure 1.5** La reconnaissance vocale [3]

### 1.3.2 Le traitement automatique du langage naturel

Une fois la reconnaissance vocale effectuée, le texte (données textuelles) est transmis au module du traitement automatique du langage naturel (TALN) pour analyser les phrases et en extraire par la suite le maximum d'informations linguistiques. Nous regroupons sous le vocable «TALN» l'ensemble de recherches et développements visant à modéliser et reproduire, à l'aide de machines, la capacité humaine à produire et à comprendre des énoncés linguistiques dans des buts de communication entre l'humain et la machine [9]. Il sera donc question ici de langage humain naturel et non pas formel.

Pour compléter cette introduction, nous commencerons par étudier les différents niveaux de représentation et de traitement des énoncés linguistiques, puis nous présenterons les notions

importantes de la langue arabe, pour analyser ensuite les difficultés les plus courantes dans le TALN en langue arabe.

### 1.3.2.1 Les niveaux de traitement

Nous allons présenter les différents niveaux de traitements essentiels pour arriver à comprendre un énoncé de langage naturel donné, ces niveaux correspondent à des modules qu'il faut d'abord développer, puis les faire interagir dans le cadre d'une application complète de traitement de la langue.

Pour parvenir automatiquement à la compréhension la plus complète d'un énoncé d'une langue donnée, voyons à présent les traitements successifs qu'il convient d'appliquer à cet énoncé.

**1) L'Analyse morphologique (Tokenisation)** : consiste à segmenter le texte en unités lexicales (tokens).

**2) L'analyse morphologique** : comprend, entre autres, le stemming (racinisation), i.e. détection du stem en supprimant ses affixes.

**3) L'Analyse morphosyntaxique** : c'est une étape nécessaire avant de faire l'analyse syntaxique. Il s'agit de l'étiquetage morphosyntaxique (en anglais, « POS tagging » pour Part-Of-Speech tagging), qui consiste à affecter à chaque token une étiquette morphosyntaxique dont la granularité peut aller d'une simple catégorie morphosyntaxique (partie du discours) à une analyse plus fine, qui comprend des traits morphologiques, comme le genre, le nombre, le temps, le mode, etc.

**4) L'Analyse syntaxique (Parsing)** : étape clé dans la gestion de dialogues, réponse aux questions, traduction et correction automatique, etc. L'analyse syntaxique s'intéresse à la structure syntaxique des phrases et sert donc à fournir les analyses plausibles des phrases et ceci en corrélation avec une grammaire qui spécifie les structures autorisées du langage. Elle consiste à prendre une chaîne de caractères et une grammaire et retourner un arbre d'analyse de cette chaîne de caractère, il est analogue à l'exécution d'un transducteur à états finis avec 1 ruban. Ce processus vise à modéliser la structure de la phrase en identifiant les relations entre ses différentes données textuelles et leur bon endroit dans la phrase (leur position relative par rapport aux autres données textuelles) afin de comprendre la relation entre ces dernières.

**5) L'Analyse sémantique** : dans cette analyse, il y a entre autres, l'utilisation d'ontologie (concepts, relations sémantiques, etc.) et l'extraction d'entités nommées et les relations entre entités nommées, afin de comprendre le sens du texte, i.e. une fois la nature et la position des mots trouvées, l'analyse sémantique a pour but de comprendre leur sens que

ce soit séparément ou lorsqu'ils sont assemblés dans la phrase, pour apprêhender le besoin de l'utilisateur puis lui retourner les résultats adéquats à sa demande vocale dans la mesure du possible.

**6) L'Analyse pragmatique :** aussi appelée « l'analyse du discours », c'est la dernière étape. Il s'agit de l'identification de la fonction de l'énoncé dans le contexte particulier de la situation dans lequel il a été produit. Cette analyse étudie donc le contexte et le contenu du discours en se basant sur l'historique des conversations et le profil collecté de l'utilisateur. Notons que dans le cas simple de l'utilisation de l'assistant vocal virtuel (cas des commandes de base), cette étape n'est pas nécessaire.

Ces niveaux de traitement, correspondant à des modules distincts de traitement, se retrouvent dans d'autres applications du TALN, l'idéal est que ces traitements soient séquentiels et il est préférable de concevoir ces niveaux de traitements en coopération, et ce, en échangeant de l'information dans les deux sens d'une façon, ainsi il est essentiel de détecter des informations sémantiques pour identifier la bonne structure syntaxique de la phrase.

Bien évidemment, comme dans tous les domaines, la perfection n'est jamais atteinte. En effet, au fur et à mesure que nous progressons dans cette hiérarchie des niveaux de traitement, les difficultés s'accumulent et les outils disponibles aujourd'hui sont moins performants, ou ne sont opérants que pour des sous-domaines particuliers. Les difficultés diffèrent également selon des critères spécifiques, mais le critère le plus important est la langue traitée.

La langue arabe est classée comme étant l'une des langues les plus complexes à analyser, d'où le manque d'outils et de corpus dédiés à son traitement automatique [10] et comme conséquence, le manque d'assistants vocaux virtuels supportant la langue arabe.

Dans ce qui suit, nous allons présenter les caractéristiques de la langue arabe les plus marquantes, pour passer ensuite aux obstacles majeurs et problèmes posés par le TALN en langue arabe ainsi que les difficultés que rencontrent les chercheurs ou toute autre personne qui souhaiterait travailler sur cette langue.

### 1.3.2.2 La langue arabe

L'arabe est la 5ème langue la plus utilisée dans le monde, avec un total de 422 millions des locuteurs devisés entre 290 millions locuteurs natifs et 132 millions utilisateurs comme une 2ème langue. Cette langue est l'une des plus anciennes dont l'origine vient de la péninsule arabique et étendue à une grande partie du monde (principalement l'Afrique du nord) avec la

diffusion de l'Islam, dont elle est considérée comme la langue liturgique de ce dernier. Nous trouvons l'arabe dans de nombreux écrits : des livres, des journaux, des TV-show, etc. Et avec l'ouverture culturelle du monde arabe, plusieurs travaux mondiaux ont été traduits vers l'arabe et vis-versa.

En ce qui concerne le monde de l'informatique, en Web, traduction et particulièrement en développement d'applications d'assistance vocale, la langue arabe est très loin d'atteindre le niveau de diffusion d'autres langues, telles que l'anglais ou le français que ce soit dans la qualité ou la quantité de résultats. Plusieurs problèmes ont conduit à cette faiblesse, comme le manque de corpus arabes dues aux difficultés du traitement automatique de l'arabe, vu sa richesse, sa diversité et sa variété. Tout cela a ralenti et empêché les chercheurs ou toute autre personne qui veut travailler sur l'arabe, de l'exploiter afin de faciliter la vie quotidienne des arabophones. De ce fait, nous allons détailler les caractéristiques et particularités de la langue arabe [11].

### 1.3.2.3 Les caractéristiques et particularités de la langue arabe

L'arabe est une langue surprenante, qui suscite l'intérêt et l'envie d'en apprendre davantage, et ce, grâce à sa richesse présentée ci-après.

#### 1) Les variétés de la langue arabe

L'arabe est la langue parlée et écrite par les peuples arabes de l'océan atlantique à l'ouest au golfe persien à l'est dont vingt-deux pays arabes. Elle est aussi la langue officielle dans deux autres pays : le Tchad et l'Erythrée. La langue arabe est marquée par ses différents dialectes et la différence qui existe entre ces derniers à cause des interactions culturelles entre les peuples arabes et d'autres peuples au cours de l'histoire. Nous pouvons citer comme exemple la différence entre le dialecte arabe des pays du golfe et le dialecte arabe des pays du Maghreb arabe, ces peuples n'ont pas la même histoire ce qui implique une différence dans les dialectes.

Selon le registre linguistique, nous distinguons trois variétés de la langue arabe [12].

- **L'arabe moderne standard,** ”العربية الفصحى”， MSA : c'est la langue officielle du monde arabe et la première pour les médias et l'éducation. L'MSA définit les règles d'écriture et de prononciation de l'arabe, qui est utilisée dans les journaux, les revues grand public, les médias, etc.
- **L'arabe classique,** ”فصحي التراث”， CA : L'ancienne arabe, de moins en moins utilisée à cause de sa complexité face au MSA, on la trouve généralement dans les textes coraniques et religieux, ou bien dans les anciens livres et aussi dans les universités et les institutions

académiques. Un exemple de l'arabe classique, est le mot ”الفَلَك“ qui veut dire bateau, remplacé par ”السَّفِينَة“ dans le MSA.

Notons que le MSA et le CA font partie de l'arabe littéral.

- **Les dialectes arabes,** ”اللهجات العربية“، DA : un dialecte est un langage parlé par les peuples arabes comme le dialecte algérien, marocain, etc. Ce qui différencie une langue d'un dialecte est le degré de reconnaissance officielle de leur statut, décrétée par l'état ou une autre forme de pouvoir dominant. D'une certaine manière, une langue est un dialecte qui a réussi à s'imposer aux autres.

## 2) Le script arabe

La langue arabe est écrite en utilisant le script arabe qui est aussi utilisé pour écrire d'autres langues indépendantes de l'arabe comme le persan, l'urdū et le kurde, mais avec plus de caractères. Dans ce qui suit nous présentons les principales caractéristiques du script arabe [13].

### – Le sens d'écriture et de lecture du script arabe

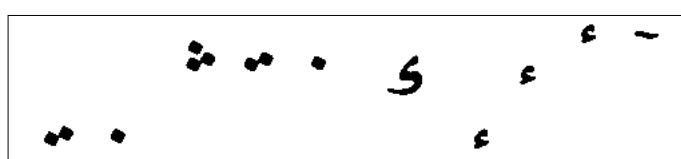
Le sens de lecture et d'écriture diffèrent d'une culture à une autre, de droite à gauche ou de gauche à droite. L'arabe, tout comme l'hébreu et le syriaque, s'écrivent et se lisent de droite à gauche, on dit qu'ils ont un sens sinistroverse, contrairement aux langues des pays occidentaux, le sens de lecture et d'écriture est dextroverse, ce qui signifie de gauche à droite.

### – Les lettres arabes

Comme toute autre langue, les lettres sont l'élément clé pour écrire l'arabe, les lettres arabes se composent de deux éléments constitutifs essentiels, les formes (dix-neuf formes) et les marques (dix marques). Ces marques sont divisées en 3 catégories : les points, le petit Kaf (ڭ) et la Hamza (ء).

Des combinaisons entre ces deux dernières, donnent les 28 lettres arabes de base.

**Figure 1.6** Les 19 formes des lettres arabes



**Figure 1.7** Les 10 marques des lettres arabes

### - La variation des formes des lettres arabes

L'écriture d'une lettre arabe dépend de sa position dans le mot i.e. la graphie d'une lettre peut changer selon qu'elle soit :

- non liée (isolée) ;
- liée à la lettre suivante mais pas la précédente (position initiale) ;
- liée à la lettre précédente et la lettre suivante (position médiane) ;
- liée à la lettre précédente mais pas la suivante (position finale).

Cependant il existe 6 lettres (ا, د, ذ, ر, ح, ق) qui ne s'attachent jamais à la lettre suivante.

Ecriture des 28 lettres arabes de base selon leur position							
Note : Lecture de droite à gauche							
Finale	Médiane	Initiale	Isolée	Finale	Médiane	Initiale	Isolée
ت	ت	ت	ت	ب	ب	ب	ب
t	tā'	b	bā'	'ā	'ā	'alif	
ح	ح	ح	ح	ج	ج	ج	ج
h	hā'	j (ğ, g)	jīm	th (ı)	th (ı)	thā'	
ذ	ذ	ذ	ذ	د	د	د	د
dh (ğ)	dhāl	d	dāl	kh (h,k)	kh (h,k)	khā'	
س	س	س	س	ز	ز	ز	ز
s	sīn	z	zāy	r	r	r	rā'
ض	ض	ض	ض	ص	ص	ش	ش
d	dād	ṣ	ṣād	sh (ş)	sh (ş)	shīn	
ع	ع	ع	ع	ظ	ظ	ط	ط
'	'ayn	z	zā'	ظ	ظ	ط	tā'
ق	ق	ق	ق	ف	ف	غ	غ
q	qāf	f	fā'	f	f	gh (ğ, ğ)	ghayn
م	م	م	م	ل	ل	ك	ك
m	mīm	l	lām	ل	ل	k	kāf
و	و	و	و	ه	ه	ن	ن
w (ū,aw)	wāw	h	hā'	ه	ه	n	nūn
				ي	ي	ي	yā'
				ي	ي	ي	y (ı,ay)

Figure 1.8 Les 28 lettres arabes de base [11]

Toutes les lettres sont des consonnes, sauf le « و » et le « ي » sont considérées comme des semi-consonnes, (ou semi-voyelles). Notons également la présence de 3 voyelles longues (le « ا », le « و » et le « ي ») et trois voyelles courtes (fatha, damma et kasra)

### – Le monocaméralisme d’écriture arabe

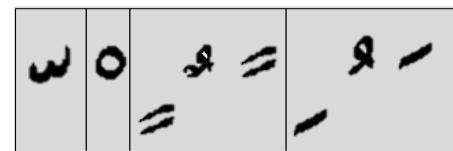
Le script arabe admet une graphie unique pour chacune des lettres, il n’existe pas donc de notion de lettres majuscules et minuscules, ainsi il n’y a pas de dissemblance entre le manuscrite et l’écriture d’imprimerie.

### 3) La prononciation en langue arabe

Pour la prononciation des lettres en langue arabe, nous utilisons ce qui suit.

- **Les diacritiques** : c’est une classe de symboles arabes, qui indiquent la façon de prononcer une lettre dans un mot. Elles sont optionnelles dans les textes arabes, un texte peut être complètement ou partiellement vocalisé, ou non vocalisé. Les diacritiques sont généralement utilisées dans les textes coraniques, ou encore dans les textes destinés à l’apprentissage de la langue arabe afin d’indiquer la prononciation correcte des lettres ambiguës et faciliter la lecture.

Il existe 4 types de diacritiques : les voyelles courtes, le tanwin, l’absence de voyelle (sukun) et la gémination (chedda ou tachdid).



**Figure 1.9** Les diacritiques [11]

#### - Les voyelles courtes

- la fatha/الفتحة : elle correspond à un petit trait sur la lettre et se prononce “a” ;
- la damma/الضمّة : elle correspond à un petit “و” sur la lettre et se prononce “u” ;
- la kasra/الكسرة : elle correspond à un petit trait sous la lettre et se prononce “i”.

#### - Le tanwin

- le tanwin avec fatha/الفتحتين : correspond à deux petits traits parallèles sur la lettre et se prononce “ane” ;
- le tanwin avec damma/الضمتين : correspond à deux petits “و” adjacents sur la lettre et se prononce “une” ;
- le tanwin avec kasra/الكسرتين : correspond à deux petits traits parallèles sous la lettre et se prononce “ine”.

#### - L’absence de voyelle : appelé aussi “sukun/السکون”, qui correspond à un petit cercle sur la lettre, ce signe indique qu’une consonne n’est pas suivie par une voyelle. Il suffit de ne pas prononcer de voyelle pour la lettre impactée par “sukun”.

- **La gémination** : l'arabe peut renforcer n'importe quelle lettre à l'aide d'un signe appelé “chadda/ الشدة” ou “tachdid/ التسديد” (intensité), qui se place au-dessus des lettres qui doivent être doublées, en vrai ce n'est que la répétition d'une même lettre dont la première porte le signe de l'absence de voyelle (sukun).
- **Les voyelles longues** : dans la langue arabe, 3 lettres sont employées pour indiquer la présence des voyelles longues :
  - o **le alif (ا)** : est précédé par une lettre portant une fatha et se prononce “aa” ;
  - o **le waw (و)** : est précédé par une lettre portant une damma et se prononce “uu” ;
  - o **le ya (ي)** : est précédé par une lettre portant une kasra et se prononce “ii”.



**Figure 1.10** Les voyelles longues [11]

Diacritiques			Voyelles longues
Voyelles Courtes	Tanwin	Absence de voyelle	Voyelles Longues
الفتحة بَ ba /ba/	الفتحتين بَّ bā /ban/	السكون بُ b. /b/	الالف بَا baa
الضمة بُّ bu /bu/	الضممتين بُّّ bū /bun/	الكسرة بِّ bi /bi/	الواو بُّّ bu
الكسرة بِّ bi /bi/	الكسرتين بِّّ bī /bin/	الشدة بـ b~ /bb/	الياء بِّي bi
Gémination			

**Figure 1.11** Prononciation, appellation et positionnement des diacritiques et voyelles longues sur la lettre “ب”

#### 4) La ponctuation en langue arabe

La langue arabe utilise la même ponctuation que les autres langues, avec quelques adaptations. Ceci est dû à son écriture de droite à gauche. En effet, une ponctuation utilisée seulement pour l'arabe est le **Tatwil**. C'est une forme d'écriture de l'arabe, elle consiste à prolonger une lettre en ajoutant le caractère du blanc souligné à cette lettre. Cette forme est utilisée pour faire apparaître une lettre clairement (équivalente à l'utilisation des lettres capitales ou la majuscule pour les langues comme le français), ou parfois pour des raisons artistiques.

Mot sans Tatweel	Mot avec Tatweel	Translation
الحمد	الحمد	Les louanges
رحيم	رحيم	miséricorde

**Figure 1.12** Mots avec et sans Tatweel**5) Styles d'écriture arabe (la topographie)**

La langue arabe a plusieurs styles d'écriture (font), ce qu'on appelle aussi "topographie".

Arabe Traditionnel	عربى	محمد	الجبر
Arabe Simplifié	عربى	محمد	الجبر
Arabe Tahoma	عربى	محمد	الجبر
Andalus	عربى	محمد	الجبر
	çarabiy~ Arabic	muHam~ad Muhammad	Aljabr Algebra

**Figure 1.13** Mots arabes écrits en différentes topographies**6) Détection de la personne à partir des formes conjuguées des verbes**

Généralement, on utilise des pronoms personnels et des sujets, pour indiquer la personne qui fait l'action dans une phrase. Dans la langue arabe, c'est un peu sophistiqué, puisqu'il y a la variation selon le genre et le nombre, de plus il y a le principe de dualisme, par exemple si nous prenons les deux pronoms 'Tu' et 'Vous', en arabe se traduisent en fonction du genre et du nombre des personnes auxquelles nous nous adressons, i.e. si on s'adresse à :

Pluriel	Duel	Singulier	
نَحْنُ Nous	نَحْنُ Nous	أَنَا Je	1 <sup>ère</sup> personne
أَنْتُمْ Vous	أَنْتَمَا Vous	أَنْثَى Tu	2 <sup>e</sup> personne masculin
أَنْتَنَّ Vous	أَنْتَمَا Vous	أَنْتِ Tu	2 <sup>e</sup> personne féminin
هُمْ Ils	هُمَا Ils	هُوَ Il	3 <sup>e</sup> personne masculin
هُنَّ Elles	هُمَا Elles	هُيَّ Elle	3 <sup>e</sup> personne féminin

**Figure 1.14** Les pronoms personnels de la langue arabe

- une personne du genre masculin on utilise le pronom : "أنت" ;
- une personne du genre féminin on utilise le pronom : "أنت" ;
- deux personnes du genre masculin ou féminin (notion de dualité), on utilise le pronom : "أنتما" ;
- trois personnes ou plus du genre masculin on utilise le pronom : "أنتم" .
- Trois personnes ou plus du genre féminin on utilise le pronom : "أنتن" .

La particularité de la langue arabe est qu'elle a tendance à ne pas utiliser les pronoms personnels, i.e. le sujet d'une phrase est implicite par la forme conjuguée du verbe. Par exemple, "اصلح" (réparer).

"لَذِنْيَا تَضْحَكُ لَكَ" se traduit par "Si tu souris à la vie, elle te sourira", mais les pronoms 'tu' et 'elle', n'apparaissent pas dans la phrase arabe, ils sont sous-entendus par les formes conjuguées des verbes.

## 7) Disponibilité d'ordre mixte

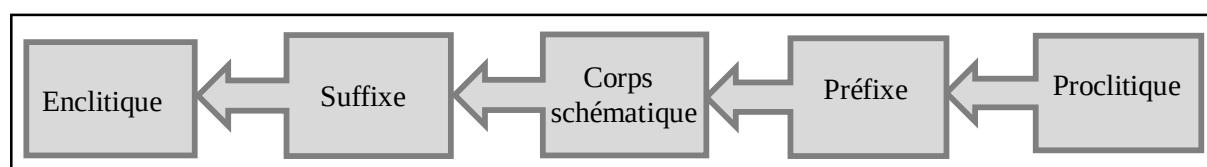
En langue arabe, nous disposons d'un ordre mixte des mots, c.à.d. la syntaxe n'est pas fixe, tels que lorsque nous mélangeons les mots d'une phrase, la phrase reste compréhensible et son sens ne change pas, par exemple, si on prend la phrase verbale « رَبِّي الْمَعْلُمْ أَجِيلًا », elle respecte l'ordre ordinaire VSC (Verbe-Sujet-Complément), mais on trouve notamment, d'autres structures moins fréquentes comme SVC « رَبِّي أَجِيلًا الْمَعْلُمْ », VCS « الْمَعْلُمْ رَبِّي أَجِيلًا » et CVS « أَجِيلًا رَبِّي الْمَعْلُمْ ». Ces structures servent généralement à mettre l'accent sur le sujet.

## 8) Structure du mot arabe

La langue arabe est une langue ayant une morphologie dérivationnelle, flexionnelle et agglutinante, étant donné qu'elle n'est pas une simple concaténation de morphèmes comme c'est le cas pour l'anglais mais c'est une concaténation de droite à gauche des éléments suivants :

- Des prépositions ou des conjonctions (proclitiques).
- Des préfixes, qui sont placés au début des mots arabes.
- Un « corps schématique » composé d'une racine et d'un schème.
- Des suffixes, qui sont placés à la fin des mots arabes.
- Des pronoms affixes (enclitiques).

Les préfixes et suffixes sont rassemblés sous le terme 'Affixes', ils expriment des traits et les rapports grammaticaux et accordent aux mots des éléments syntaxiques, ils marquent l'aspect verbal, le mode du verbe, le genre, le nombre, le cas, la personne, etc.



**Figure 1.15** Structure d'un mot dans la langue arabe (mot maximal)

Cette structure, auprès de l'accumulation des affixes peut aller jusqu'à la formation d'un mot qui signifie toute une phrase, à titre d'exemple le mot « أَسْتَدْعُونَهَا » (« Est-ce-que vous allez vous rappeler d'elle ? »), qui a une forme interrogative :

أ	Proclitique : conjonction d'interrogation.
و	Proclitique : particule du futur.
ت	Préfixe : préfixe verbal exprimant l'aspect inaccompli.
تذکر	Corps schématique : dérivé de la racine ذكر selon le schème ت فعل
و	Suffixe : suffixe verbal exprimant le masculin pluriel.
ن	Suffixe : suffixe verbal exprimant le nominatif.
هـ	Enclitique : pronom affixe exprimant l'accusatif.

**Tableau 1.1** Structuration du mot أستذكرونها

### 1.3.2.4 La morphologie de la langue arabe

En arabe, la majorité des mots du dictionnaire (appelés aussi « lemmes ») sont construits sur la base d'une racine arabe, tout en respectant un schème.

Dans ce qui suit, nous allons définir quatre concepts :

- la racine arabe ;
- le schème ;
- le lemme ;
- le stem.

#### 1) La racine arabe

En arabe, la racine arabe (أصل) représente une suite de consonnes formant le radical du mot. C'est un élément important dans les langues dérivationnelles. En effet, à chaque racine correspond un champ sémantique et à l'aide de différents schèmes, nous pouvons générer une gamme de vocabulaire appartenant à ce champ sémantique.

Pour représenter une racine trilitière, nous écrivons [ف ع ل] trois consonnes qui forment une racine réelle de la langue et qui expriment l'idée d'« agir ». Elles expriment une idée générale et c'est par l'insertion de voyelles brèves, de voyelles longues, de suffixes et de préfixes que se précise cette idée. Et c'est suivant la racine que les mots arabes sont classés traditionnellement dans le dictionnaire.

Le tableau suivant illustre la racine des mots de la phrase arabe suivante.

Mot	الحروب	من	ناقة	الأعرابيّ	استغنم
Racine	حرب	من	نوق	عرب	غم

**Tableau 1.2** Racines trilitères des mots de la phrase “استغنم الأعرابي ناقة من الحروب”

## 2) Le schème

Le schème (وزن) se compose des 3 lettres [ف ع ل] qui correspondent aux lettres de la racine de base فعل، la racine peut être augmentée par des affixes. Les schèmes sont des modèles de différentes structures (moules) qui s'appliquent aux racines pour dériver et créer de nouveaux mots. Chaque application d'un nouveau schème à la même racine donne une nouvelle signification. Le tableau suivant montre quelques schèmes verbaux et schèmes nominaux de la langue arabe.

Mot	كَاتِبٌ	كُتُبٌ	كِتَابٌ	كَاتِبٌ	كَاتِبٌ	يُكَاتِبٌ	كَاشِبٌ	أَكْتَبٌ	يَكْتُبٌ	كَتَبٌ
Schème	فَعَاعِلٌ	فُعْلٌ	فِعَالٌ	فَاعِلٌ	فَاعِلٌ	يُفَاعِلٌ	فَاعِلٌ	فَاعِلٌ	يَفْعَلٌ	فَعَلٌ
Mot	مَكْتَبَاتٌ	مَكَاتِبٌ	إِسْكَنَّاتٍ	إِكْتَبَ	كُوْيِتُبٌ	مُكَاتَبَةٌ	كِتَابَةٌ	كَتَبٌ	مَكْتَبَةٌ	مَكْتُوبٌ
Schème	مَفْعَلَاتٌ	مَفَاعِلٌ	إِسْتَفْعَلٌ	إِفْتَعَلٌ	فَوْيَعْلٌ	مَفَاعِلَةٌ	فِعَالَةٌ	فَعَالٌ	مَفْعَلَةٌ	مَفْعُولٌ

**Tableau 1.3** Quelques modèles de schèmes appliqués à la racine (كتب/KTB/écrire)

## 3) Le lemme

Le lemme est l'entrée lexicale dans un lexique ou un dictionnaire, il s'agit d'une racine (أصل) insérée dans un schème (وزن), donc le fait de connaître le lemme i.e. le couple (racine, schème), nous pouvons déduire les formes fléchies d'un nom ou d'un verbe.

- Pour les verbes, le lemme représente la forme à la 3<sup>ème</sup> personne du masculin singulier (هو) de l'accompli (الماضي).
- Pour les noms variables, le lemme représente le masculin singulier.
- Pour les noms invariables (pronoms personnels) et les particules, il s'agit de leur forme classique vocalisée.

Le tableau suivant illustre les lemmes des mots de la précédente phrase arabe

Mot	الحروب	من	ناقة	الأعرابي	استغنم
Lemme	حرب (اسم)	من (حرف)	ناقة (اسم)	أعرابي (اسم)	استغنم (فعل)

**Tableau 1.4** Lemmes des mots de la phrase ‘استغنم الأعرابي ناقة من الحروب’

## 4) Le stem

Le « stem » est la partie qui reste d'un mot après avoir supprimé ses affixes (préfixes, infixes et suffixes). Les termes « racine arabe » et « stem » désignent souvent les mêmes lettres radicales. En effet, dans l'exemple du mot كاتب, la racine arabe et le stem désignent tous les deux les mêmes lettres radicales, à savoir كتب, mais ce n'est pas toujours le cas. Afin d'illustrer

nos propos, prenons l'exemple du verbe اِتَّصَل. Commençons tout d'abord par le stem. Plusieurs étapes sont nécessaires pour remonter au stem de ce verbe.

- Étape 1 : briser la gémination (chedda) du verbe اِتَّصَل pour obtenir la forme اِتَّصَل.
- Étape 2 : identifier le schème qui a été utilisé pour construire le résultat de la précédente étape. Dans cet exemple, il s'agit du schème verbal اِفْتَعَل.
- Étape 3 : déduire le stem تَصَل, en faisant correspondre les trois lettres [ف ع ل] du schème avec les lettres du résultat de la première étape.

Passons à présent à la racine arabe. Plusieurs étapes sont nécessaires pour y remonter.

- Étape 1 : briser la gémination (chedda) du verbe اِتَّصَل pour obtenir la forme اِتَّصَل.
- Étape 2 : effectuer une transformation morphologique, en substituant la première lettre ت par la lettre و (l'arabe ne permet pas la succession d'une kasra et d'un waw sakin, d'où la substitution de la lettre و originelle par la lettre ت). Nous obtenons alors la forme اوتَّصَل.
- Étape 3 : déduire la racine arabe [و ص ل], en faisant correspondre les trois lettres [ف ع ل] du schème verbal اِفْتَعَل avec les lettres du résultat de la deuxième étape.

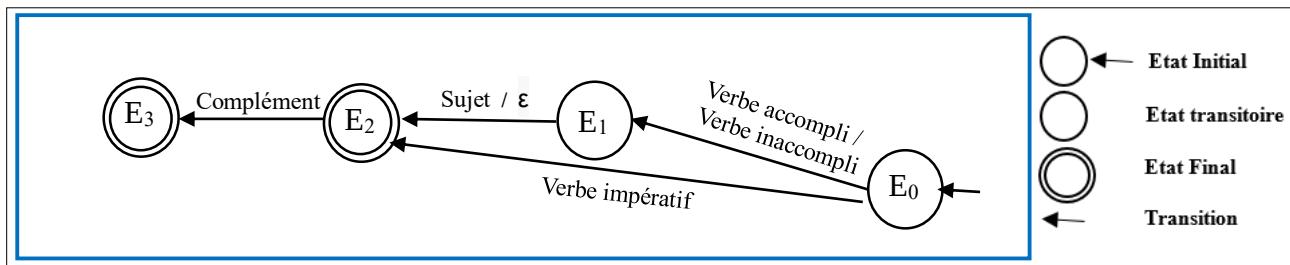
### 1.3.2.5 La syntaxe de la langue arabe

La syntaxe arabe étudie la façon dont on combine les mots pour bien former des phrases. En arabe, nous distinguons trois types de phrases : la phrase verbale, la phrase nominale et la phrase locative [14].

#### 1) La phrase verbale

C'est une phrase qui commence généralement par un verbe qui est le noyau de la phrase verbale, peu importe sa forme et sa conjugaison (accompli/ماضي, inaccompli/مضارع, impératif/أمر, etc.). Elle se constitue dans le cas ordinaire d'un **verbe** (accompli/inaccompli) + **sujet** + **complément**, ou bien d'un **verbe** (impératif) + **complément**.

Un complément, fréquemment peut être un objet/مفعول به, un locatif (circonstanciel/نون), ou combinaison des deux. Un circonstanciel à son tour peut être temporel (بعد/قبل), spatial (في/فوق/تحت), etc. Et il peut être placé n'importe où dans la phrase. Ci-après un transducteur à états finis simplifié montrant la syntaxe de base d'une phrase verbale en arabe.



**Figure 1.16** Transducteur à états finis montrant la syntaxe de base d'une phrase verbale arabe

(راجعا يُراجع) الطالب	(راجعا يُراجع)	راجع
(راجعا يُراجع) الطالب الدرس	(راجعا يُراجع) الدرس	راجع الدرس
(راجعا يُراجع) الطالب في المكتبة	(راجعا يُراجع) في المكتبة	راجع في المكتبة
(راجعا يُراجع) الطالب الدرس في المكتبة	(راجعا يُراجع) الدرس في المكتبة	راجع الدرس في المكتبة
(راجعا يُراجع) في المكتبة الدرس	(راجعا يُراجع) في المكتبة الدرس	راجع في المكتبة الدرس

**Tableau 1.5** Quelques exemples de phrases verbales arabes ayant différentes syntaxes

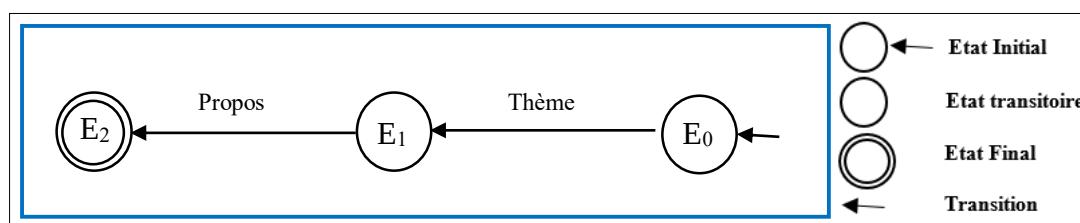
En revanche, l'arabe est une langue flexible, i.e. nous pouvons trouver des phrases verbales sans compléments, sans sujets, sans les deux ou avec les deux.

Comme la langue arabe est une langue d'ordre mixte, nous pouvons trouver d'autres syntaxes de phrases verbales, mais elles sont rarement utilisables.

## 2) La phrase nominale

C'est une phrase qui se compose de deux éléments : le premier est le thème/مبدأ (un nom défini) et le deuxième est le propos/خبر. Ce dernier vient du verbe خبر qui veut dire informer, donc le propos est l'information que nous donnons sur le thème.

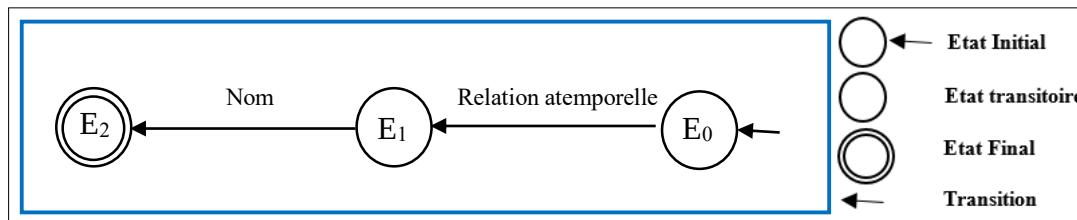
Le propos s'accorde en genre et en nombre avec le thème et il peut être un nom (الطالب نشيط), (الطالب يُراجع الدرس) ou (الطالب في المكتبة), un adjectif (الطالب النشيط), un locatif (الطالب في المكتبة) ou des combinaisons entre ces derniers du genre (الطالب في المكتبة الدرس)، (الطالب في المكتبة يُراجع الدرس)، (يُراجع المكتبة الدرس)، etc. Voici un transducteur d'états finis simplifié montrant la syntaxe de base d'une phrase nominale en arabe.



**Figure 1.17** Transducteur à états finis montrant la syntaxe de base d'une phrase nominale arabe

### 3) La phrase locative

C'est une phrase qui commence par une relation atemporelle ou un locatif, suivie d'un nom, exemple **في الجامعة أستاذة** «». Voici un transducteur d'états finis simplifié montrant la syntaxe de base d'une phrase locative en arabe.



**Figure 1.18** Transducteur à états finis montrant la syntaxe de base d'une phrase locative arabe

Comme la langue arabe est une langue flexible et d'ordre mixte, nous pouvons trouver d'autres syntaxes de phrases nominales et locatives.

#### 1.3.2.6 Les difficultés du TALN Arabe

Dans cette partie nous allons révéler les principaux obstacles qui font ralentir le TALN en langue arabe par rapport à d'autres langues [10] [15].

##### 1) Absence des diacritiques

Les diacritiques sont souvent absents dans les textes arabes, de ce fait c'est au lecteur de déduire les diacritiques au moment de la lecture, ce qui constitue un grand facteur d'ambiguité. Illustrons cette ambiguïté avec le mot **كتب**/ktb qui possède 17 façons de vocalisation et 9 catégories grammaticales ou encore le mot **بين** :

Vocalisation	Prononciation	Signification	Catégorie grammaticale
بَيْنَ	bayyana	a démontré (il)	verbe
بَيْنَ	bayyanna	ont démontré (elles)	verbe
بَيْنَ	bayyin	clair	adjectif
بَيْنَ	bayna	entre / parmi	préposition

**Tableau 1.6** Ambiguïté du mot «**بين**» causée par l'absence de diacritiques

##### 2) Segmentation des textes

La segmentation d'un texte consiste à délimiter les segments de ses éléments de base en des éléments constituants différents niveaux structurels tels que : les paragraphes, les phrases, les

syntagmes, les mots graphiques, etc. Dans la langue arabe, ceci ne se contente pas uniquement des signes de ponctuations et des marqueurs typographiques, car c'est derniers ne sont pas utilisés d'une façon régulière, de plus en langue arabe il n'existe pas de lettres majuscules pour marquer le début d'une nouvelle phrase. Illustrons un cas de cette ambiguïté avec la phrase « تراوح المنحة الجامعية لفصل ما بين 4200,15 د.ج و 5120,15 د.ج ». Dans cette phrase, ni les points, ni les virgules, ni les conjonctions de coordination « و » et « ف » , ne peuvent décider du déclanchement ou de la segmentation de la phrase jusqu'à ce que leur contexte de droit et gauche soient consultés. De ce fait, l'étude des mots et leur voisinage est nécessaire pour désambiguïser les frontières de la phrase.

### 3) Ambiguïtés morphologiques

La morphologie consiste à séparer et identifier des morphèmes semblables aux proclitiques, préfixes, infixes, suffixes et enclitiques dans un mot. Mais le fait que la langue arabe est une langue agglutinante et souvent non vocalisé rend la reconnaissance des entités lexicales dans un texte difficile, ce qui engendre une ambiguïté au cours de l'analyse morphologique. Prenons l'exemple du mot « فصل/FSL » qui peut être :

فصل/Fasala	verbe accompli (il a licencié)	فصل/Fasloune	nom (chapitre/saison)
فصل/Fassala	verbe accompli (il a découpé)	ف+صل/Fa Sil	conjonction + verbe impératif (lier)

Tableau 1.7 Ambiguïté morphologique du mot « فصل »

### 4) Ambiguïtés morphosyntaxiques

L'étiquetage morphosyntaxique (POS tagging en anglais) ou la désignation d'étiquettes (genre, nombre, temps, mode, etc.) aux différents mots du texte (nom, verbe, etc.), peut causer une ambiguïté lorsque ces textes sont partiellement ou totalement non vocalisés. Illustrons l'ambiguïté morphosyntaxique avec le mot كتب/KTB qui étant non vocalisé, admet plusieurs étiquettes :

Mot vocalisé	Prononciation	Signification	Étiquette
كتب	kataba	il a écrit	verbe, 3 <sup>e</sup> personne masculin, singulier de l'accompli actif
كتب	koutiba	il a été écrit	verbe, 3 <sup>e</sup> personne masculin, singulier de l'accompli passif
كتب	koutouboune	des livres	nom, masculin, pluriel
كتب	katboune	un écrit	nom, masculin, singulier
كتب	kattib	fais écrire	verbe, 2 <sup>e</sup> personne masculin, singulier de l'impératif

Tableau 1.8 Ambiguïté morphosyntaxique de mot « كتب »

## 5) Ambiguïtés de lemmatisation

La lemmatisation des pluriels brisés représente des ambiguïtés en langue arabe car :

- Nous pouvons trouver des mots dans leur forme singulière, ayant des schèmes identiques à des mots dans leur forme de pluriel brisé, prenons l'exemple des 2 mots :

قلم				عرب			
Forme	Signification	Schème	Lemme	Forme	Signification	Schème	Lemme
Singulier	crayon	فعل	قلم	Pluriel brisé	arabes	فعل	عربي

**Tableau 1.9** Ambiguïté 1 de lemmatisation

- Un nom à sa forme singulière peut avoir plusieurs formes au pluriel brisé, par exemple le mot suivant le schème فعل، a plusieurs formes au pluriel brisé, tels que : فعل, مفاعل, أفعال : شيوخ، مشايخ، أشياخ.
- Des mots ayant des schèmes identiques au pluriel brisé, n'ont pas les mêmes schèmes à leur forme singulière. Prenons l'exemple des 3 mots arabes dans le tableau suivant.

Mot au pluriel brisé	Schème au pluriel brisé	Mot au singulier	Schème au singulier
وسائل	فعائل	وسيلة	فعيلة
رسائل		رسالة	فعالة
زبائن		زبون	فועל

**Tableau 1.10** Ambiguïté 2 de lemmatisation

- Les cas particuliers du pluriel brisé : les noms irréguliers (أمرأة / نساء /femme(s)), les noms singuliers qui n'ont pas de pluriels (المرء /individu), les noms pluriels qui n'ont pas de singuliers (القوم /tribus).

## 6) Ambiguïtés sémantiques

Une phrase est sémantiquement ambiguë, si nous pouvons lui correspondre aux moins deux sens distincts. Clarifions cette ambiguïté par la phrase arabe « رأت سارة امرأة بـالنظارات », qui veut dire en français : « Sarah a vu une femme avec des lunettes ». Nous pouvons assigner à cette phrase 2 structures syntaxiques et interprétations distinctes.

- 1- Sarah [a vu [une femme] [avec des lunettes]] qui nous dit que c'est au moyen des lunettes que Sarah a vu une femme. Cette phrase peut être désambiguisee par exemple en remplaçant la préposition بـ par le mot : بـواسطة / باستعمال / بـفضل etc.

- 2- Sarah [a vu [une femme [avec des lunettes]]] qui nous dit Sarah a vu une femme qui portait des lunettes. Cette phrase peut être désambiguisee par exemple en remplaçant la préposition بـ par le mot : تلبـ / تـحملـ / تـرتديـ, etc.

### 1.3.3 La synthèse vocale

C'est la dernière étape qui vient conclure le processus d'assistance vocale, une fois le TALN est effectué. La synthèse vocale (SV) est une technique informatique de synthèse sonore permettant de concevoir de la parole artificielle à partir de n'importe quel texte, elle correspond au **feedback** de l'IA qui se détermine à travers une voix synthétique (lecture vocale de la réponse à l'utilisateur), afin de conserver le processus conversationnel de bout en bout.

Cette étape permet de communiquer des réponses à l'utilisateur, symbole d'une interface Homme-Machine complète et aussi d'une expérience utilisateur bien conçue.

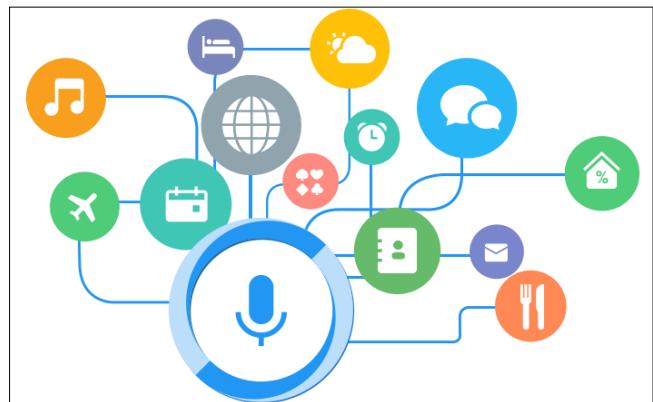
## 1.4 À quoi sert l'assistant vocal ?

Les assistants virtuels sont utilisés pour aider les utilisateurs dans leurs tâches habituelles, ils fournissent une large variété de services, qui croît jour après jour. Ces services ont pour objectif de :

- donner des résultats pour les demandes vocales des utilisateurs de la même manière que pour entrer une requête de recherche dans un navigateur web ;
- lancer des jeux et des applications sur smartphone (Play Store, Maps, YouTube, Facebook, Yassir, Viber et bien d'autres applications) ;
- télécharger des jeux et des applications à partir de Play Store ;
- faire des réservations (billet d'avion, évènement, restaurant, Taxi, etc.) ;
- fournir des informations comme le bulletin météo, les actualités, les taux de change pour la conversion monétaire, des itinéraires ;
- faire des listes de tâches, des listes de courses, des planifications ;
- régler l'alarme, écouter de la musique et des stations radio ;
- acheter des articles sur, par exemple, Amazon ou JUMIA ;
- lire de textes, des emails, des messages, des livres, contenu des sites web à haute voix ;
- passer des appels téléphoniques, envoyer des messages, chercher des contacts ;

- ajuster la luminosité du téléphone, allumer la torche, activer/désactiver la géolocalisation, activer/désactiver le Bluetooth, envoyer des documents, effectuer des calculs numériques et bien d'autres fonctionnalités ;
- effectuer des dialogues avec l'utilisateur (moyen de divertissement).

En résumé, tout ce que l'utilisateur exécutait à partir de son smartphone, en étant en ligne ou hors ligne, aujourd'hui les assistants vocaux virtuels ont la capacité de le faire à sa place d'une manière très élaborée et cela en utilisant seulement la voix, ce qui permet l'économie du temps et de l'effort.



**Figure 1.19** Quelques services fournis par les assistants vocaux virtuels

## 1.5 Conclusion

Ce chapitre a été une introduction à l'objet d'étude de ce mémoire, il a été consacré aux définitions de concepts liés à notre thème. Nous avons commencé par définir les assistants vocaux virtuels, leur fonctionnement de base et leur usage, puis nous avons résumé quelques concepts importants de la langue arabe dont il est nécessaire de maîtriser avant d'entamer n'importe quel projet traitant la langue arabe. Dans le chapitre qui suit, nous allons d'abord analyser différentes méthodes et techniques ayant trait à l'assistance vocale, qui permettront de construire de tels assistants intelligents. Nous ferons ensuite une analyse de l'existant, en passant en revue quelques applications d'assistance vocale disponibles actuellement.



## ÉTAT DE L'ART

Dans ce chapitre nous allons nous focaliser sur la phase d'analyse. Cette analyse comprend deux parties :

La première partie est l'analyse des articles connexes, recherches fondamentales ainsi que les travaux primordiaux, qui sont en lien avec notre sujet. Nous nous sommes inspirés de ces recherches pour adopter quelques techniques et approches qui ont donné des bons résultats dans ce domaine et exploiter celles qui nous semblent adéquates avec nos besoins pour la réalisation de notre travail.

La deuxième partie porte sur l'analyse de l'existant, dans laquelle nous allons étudier des applications d'assistance vocale, qui ont été réalisées dans notre domaine, afin d'en dégager les points forts, les adapter à nos besoins et les intégrer dans notre système. Cette analyse nous permettra aussi d'identifier les imperfections de chaque application, afin de les corriger autant que faire se peut et garantir la meilleure expérience utilisateur possible.

Nous discutons, avant de conclure, de notre assistant vocal et de ce qu'il apportera comme avantages.

### 2.1 Analyse des articles connexes

Dans cette partie, nous allons faire un compte rendu de la littérature sur l'architecture et les algorithmes utilisés dans les assistants virtuels, où nous allons passer en revue, dans l'ordre, les fonctionnalités suivantes :

- les algorithmes de reconnaissance vocale ;
- la fonctionnalité de traitement du langage naturel (TALN) ;
- les approches de génération de réponses ;
- les stratégies de création de bases de connaissances.

Nous conclurons cette partie par une discussion sur les algorithmes de synthèse vocale.

#### 2.1.1 Approches pour la reconnaissance vocale

La conversion de la parole en texte commence par un processus appelé reconnaissance vocale [8]. Le but est d'obtenir une reconnaissance vocale de vocabulaire large indépendante du

locuteur, dont les améliorations peuvent être mesurées selon la taille du vocabulaire, la capacité de reconnaître des locuteurs spécifiques, la capacité à traiter un flux continu de mots, la capacité de filtrer le bruit (le trafic, la musique de fond ou la parole) et la capacité de traiter la parole à différentes distances du microphone. En revanche, il existe 3 outils pour faire correspondre les vecteurs de caractéristiques acoustiques avec les mots correspondants les plus probables :

- **Un modèle acoustique** [16] : ce modèle nous donnera  $P(X|W)$ , étant donné un mot, la probabilité d'entendre un son. Nous trouverons l'argmax<sup>2</sup> sur tous les mots de notre langue w, ou le mot le plus susceptible de représenter ce son. Les modèles acoustiques sont généralement formés sur les enregistrements sonores et les transcriptions qui les accompagnent et qui peuvent être utilisés pour trouver empiriquement ces probabilités. La représentation statistique de chaque phonème généré à partir de l'analyse du corpus sonore est généralement représentée sous la forme d'un modèle de Markov caché (HMM). Les HMM sont un processus de Markov où les états ne sont pas observés / cachés (nous ne connaissons pas les phonèmes réels utilisés et nous approchons de cette information).
- **Un modèle de langue** : il peut nous dire la probabilité d'entendre un mot donné dans notre langue.
- **Un dictionnaire** : contenant une liste de mots et leurs phonèmes.

Étant donné un son, nous pouvons décoder le mot en utilisant la règle de Bayes:

$$W = \text{argmax}_{W \in L} P(W|X) = \text{argmax}_{W \in L} \frac{P(X|W)P(W)}{P(X)} = \text{argmax}_{W \in L} P(X|W)P(W)$$

**Équation 2.1** Fonction de décodage d'un mot (règle de Bayes) [16]

### 2.1.1.1 Approches symboliques

Aussi appelées méthodes traditionnelles de reconnaissance de la parole. Les approches symboliques sont l'une des premières voies tentant d'aboutir à l'IA [17] [18], elles désignent l'ensemble des approches et techniques en IA qui sont fondées sur des représentations symboliques (manipulation des symboles), i.e. lisibles par l'homme.

Ces approches se basent sur des pipelines sophistiqués composés de multiples algorithmes écrits par l'homme, pour associer à chaque phonème un caractère, en se basant sur :

---

<sup>2</sup> En mathématiques, l'argument du maximum, noté argmax, est l'ensemble des points en lesquels une expression atteint sa valeur maximale.

- des systèmes à base de connaissances (systèmes experts) : des outils capables de reproduire les mécanismes cognitifs d'un expert, dans un domaine particulier ;
- des systèmes à base de règles : où nous effectuons un raisonnement à partir de faits et de règles connues ;
- la logique et le raisonnement symbolique ;
- le traitement de la langue naturelle par des grammaires de symboles ;
- les algorithmes de recherche dans les espaces d'états.

### 2.1.1.2 Deep learning (apprentissage profond)

Le deep learning a conduit au développement de méthodes encore plus efficaces pour décoder les sons en phonèmes [19]. Il s'agit des réseaux de neurones avec une couche d'unités visibles stochastiques et N couches d'unités cachées stochastiques. Il n'y a pas de connexions avec chaque couche, mais il existe généralement des connexions entre chaque unité de la couche visible et chaque unité de la couche cachée. Les poids sur chaque bord sont déterminés par une fonction d'activation et sont modifiés et optimisés pendant l'entraînement via la propagation arrière.

Plutôt que d'inclure un HMM pour représenter la séquence de sons qui composent un phonème, les modèles basés sur le deep learning représentent les phonèmes tels que chaque trame sonore qui compose le phonème est une unité visible. [20] ont testé trois variantes de deep learning dans la modélisation acoustique.

- 1) l'inconditionnel ;
- 2) le conditionnel, qui considère la variable visible des périodes précédentes comme entrées conditionnelles ;
- 3) le conditionnel interpolant, qui prend en compte les périodes précédentes et futures comme entrées conditionnelles.

Parmi ceux-ci, ces auteurs trouvent que le conditionnel interpolant génère les meilleurs résultats. Ci-après quelques systèmes de reconnaissance vocale utilisant le deep learning.

**1) DeepSpeech (*Scaling-up end-to-end speech recognition*) de Mozilla :** c'est un système de reconnaissance vocale développé et déployé chez Mozilla [21] [22] à l'aide d'un apprentissage profond (deep learning) de bout en bout. Son architecture repose sur des pipelines de traitements complexes. Le système n'a pas besoin de composants conçus à la main pour

modéliser le bruit de fond, la réverbération ou la variation des haut-parleurs, mais apprend directement une fonction robuste à ces effets. Ainsi, il n'a pas besoin d'un dictionnaire de phonèmes, ni même de concept de «phonème». La clé de cette approche est un système d'entraînement d'un grand réseau de neurones récurrent (RNN) bien optimisé qui utilise plusieurs GPU, ainsi qu'un ensemble de nouvelles techniques de synthèse de données qui permettent d'obtenir une grande quantité de données variées pour l'entraînement.

Le DeepSpeech surpassé les résultats précédemment publiés sur le Switchboard Hub5'00 largement étudié, atteignant 16,0% d'erreur sur l'ensemble de test complet. Le DeepSpeech gère également mieux les environnements bruyants que les systèmes vocaux commerciaux de pointe largement utilisés.

Le RNN de DeepSpeech est composé de 5 couches d'unités cachées. Les 3 premières couches ne sont pas récurrentes. La sortie de la première couche à un instant  $t$ , dépend du fragment des spectrogrammes  $xt$  et d'un contexte de fragment sur chaque côté (droit et gauche). Le reste des couches non récurrentes fonctionnent sur des données indépendantes pour chaque pas. La sortie des 3 premières couches est calculée grâce à une formule spéciale. La 4<sup>ème</sup> couche est bidirectionnelle récurrente. La 5<sup>ème</sup> (non-récurrente) couche prend les unités forward et backward comme entrée. La couche de sortie est une fonction softmax standard qui produit la prédiction de la probabilité d'un caractère pour chaque tranche audio et un caractère de l'alphabet.

L'un des inconvénients inconvenients de cette approche est qu'elle nécessite un corpus de données conséquent, ce qui n'est pas évident de trouver surtout pour une langue comme l'arabe. Par ailleurs, il faut garder à l'esprit que les résultats obtenus grâce à une telle approche sont directement tributaires de la qualité du corpus d'apprentissage. Ce dernier se doit donc d'être représentatif.

**2) Système de reconnaissance vocale automatique (ASR) multilingue de Google :** c'est un système de reconnaissance vocale développé et déployé chez Google à l'aide d'un apprentissage profond (deep learning) [23] de bout en bout, permettant aux utilisateurs d'interagir naturellement avec le système en plusieurs langues. Le système a été évalué et les résultats ont montré que son architecture est capable de gérer plusieurs langues sans impact significatif sur la précision et la latence par rapport aux systèmes de reconnaissance vocale monolingues. La topologie complète du réseau de ce système contient : 1040 unités visibles, 4

couches cachées, 2560 unités cachées, 34 langages de sortie, des unités linéaires rectifiées et une couche de sortie softmax.

Concernant la procédure d'entraînement, le système utilise la descente de gradient stochastique asynchrone et un framework logiciel qui utilise des clusters de calcul avec des milliers de machines. Cette distribution permet d'exploiter de grandes quantités de données pour former de grands modèles.

L'avantage de ce système par rapport à d'autres est qu'il reconnaît la parole même dans la présence du bruit, il est intégrable avec les applications Android et le score de prédiction est toujours élevé.

### **2.1.2 Approches pour le traitement automatique du langage naturel**

L'objectif du TALN est de prendre la sortie non structurée de la RV et de produire une représentation structurée du texte qui contient la compréhension du langage parlé ou, dans le cas de la saisie de texte, la compréhension du langage naturel.

Dans la section suivante, nous allons explorer un certain nombre de méthodes pour extraire des informations sémantiques et leur signification du langage parlé et écrit, afin de créer des structures de données grammaticales qui peuvent être traitées par l'unité de gestion du dialogue.

#### **2.1.2.1 Reconnaissance de l'acte de dialogue**

Une façon d'extraire du sens du langage naturel consiste à déterminer la fonction du texte / de la phrase (question, suggestion, offre, commande, etc.), c'est ce qu'on appelle la reconnaissance de l'acte de dialogue (DA) [24]. Dans les systèmes de reconnaissance des actes de dialogue, un corpus de phrases (données d'apprentissage) est étiqueté avec la fonction de la phrase et un modèle d'apprentissage automatique statistique est construit. Ce modèle prend une phrase et génère sa fonction. Le modèle utilise un certain nombre de caractéristiques différentes pour classer les phrases.

##### **1) Méthodes bayésiennes**

L'idée derrière l'utilisation d'une approche bayésienne des modèles DA est de trouver la probabilité de chaque séquence possible d'actes de dialogue DA qui pourraient représenter une phrase ou un énoncé (U) et de trouver la séquence d'actes de dialogue DA<sub>max</sub> avec la plus forte probabilité de se produire.

$$DA_{Max} = argmax_{DA} P(DA|U) = argmax_c \frac{P(DA)P(U|DA)}{P(U)}$$

$$DA_{Max} = argmax_{DA} P(DA) * P(U|DA)$$

**Équation 2.2** Fonction pour trouver la séquence d'actes de dialogue avec la plus forte probabilité de se produire [16]

En supposant que les N mots individuels de l'énoncé sont connus et que les hypothèses d'indépendance des mots naïfs de Bayes subsistent, cela conduit à :

$$DA_{Max} = argmax_{DA} P(DA) * \prod_{i=1..N} P(W_i|DA)$$

**Équation 2.3** Fonction pour trouver la séquence d'actes de dialogue avec la plus forte probabilité de se produire si les mots d'un énoncé sont connus et les hypothèses d'indépendance des mots naïfs de Bayes subsistent [16]

Il s'agit du modèle unigramme, où  $P(DA)$  et  $P(W_i | DA)$  peuvent être quantifiés à partir des données empiriques. En utilisant ce classifieur Naïve Bayes, [25] ont trouvé un taux de reconnaissance / précision de 74%, pour classer l'acte de dialogue correct à partir d'une phrase donnée. Les modèles N-gram sont fréquemment utilisés pour inclure l'historique des dialogues dans le modèle. Ces modèles estiment  $P(DA | N DA$  historiques) plutôt que  $P(DA)$ . En supposant  $N = 3$ , nous estimerions  $P(DA | DA_{n-1}, DA_{n-2}, DA_{n-3})$ .

Les modèles de Markov cachés peuvent également être utilisés pour modéliser l'historique du dialogue, où chaque état représente un acte de dialogue dans l'historique des conversations de l'assistant. De même, les classifieurs à base de réseaux de neurones peuvent également être formés. Une combinaison de HMM et de réseaux de neurones a atteint une précision de 76% [26].

## 2) Méthodes non bayésiennes

La classification DA est un problème classique d'apprentissage automatique. Un certain nombre d'approches NB (non bayésiennes) ont été utilisées [27], notamment: les réseaux de neurones, les perceptrons multicouches, les k plus proches voisins et les arbres de décision.

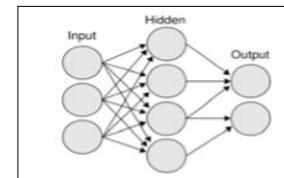
- [28] a mis en œuvre un perceptron multicouche. La couche d'entrée du réseau de neurones MLP a utilisé des caractéristiques suprasegmentales (par exemple, le stress et l'intonation) extraites de l'énoncé, des caractéristiques de durée (c'est-à-dire le temps nécessaire pour

prononcer un mot) et des caractéristiques prosodiques. Le réseau de neurones a utilisé une couche cachée avec 60 neurones et une couche de sortie avec 12 nœuds, correspondant à chacune des 12 configurations / étiquettes DA possibles utilisées. Le réseau a été formé en utilisant la rétro-propagation avec une fonction de coût d'entropie croisée:

$$DA = -\ln \sum_x[y \ln(a) + (1 - y) \ln(1 - a)]$$

**Équation 2.4** Fonction de coût d'entropie croisée [28]

N est le nombre d'éléments de données d'apprentissage, x est les entrées / fonctionnalités d'apprentissage et y est les sorties dans chaque cycle de rétro-propagation.



**Figure 2.1** Exemple d'un MLP [28]

- En revanche, [29] ont utilisé un autre type de réseau de neurones, le réseau de *Kohonen* ou *Self-Organizing Map* (SOM), qui fait correspondre à partir d'un ensemble de données (les caractéristiques d'entrée) sur une grille de sortie bidimensionnelle représentant les DA possibles. Plusieurs fonctionnalités d'entrée ont été utilisées, notamment:

- le locuteur ;
- l'utilisation du mot d'interrogation «wh» ;
- l'utilisation du point d'interrogation.

Ces réseaux utilisent un cluster non supervisé pour créer les étiquettes utilisées pour classer les phrases d'entrée.

- Une troisième approche NB de la classification DA est l'apprentissage basé sur la mémoire (MBL) utilisant K-Nearest-Neighbour (KNN). L'idée derrière MBL est de stocker en mémoire toutes les instances qui ont été vues et classées (c'est-à-dire dans un premier temps, les données d'entraînement). Lorsqu'une instance invisible (nouvelle DA) est vue, le système récupère les k voisins les plus proches (c'est-à-dire les énoncés les plus similaires de l'ensemble d'apprentissage) et l'instance invisible est classée en fonction de la classe majoritaire ou "dominante" de cet échantillon.

[54] a atteint une précision optimale de 72% en utilisant l'approche du plus proche voisin [27].

- Une quatrième approche est l'apprentissage basé sur la transformation dans lequel un classifieur simple est utilisé pour obtenir une solution généralement correcte et des transformations sont appliquées pour obtenir une solution spécifiquement correcte.

### **3) Méthode basée sur le calcul de similarité**

C'est une mesure permettant de comparer des documents textuels et qui consiste à comparer des chaînes de caractères. C'est une métrique qui mesure la similarité ou la dissimilarité entre deux chaînes de caractères dans un même document ou dans des documents différents. Par exemple, les chaînes de caractères "الجَوَّة" et "الجَوَّة" peuvent être considérées comme similaires. Une telle mesure sur les chaînes de caractères fournit une valeur obtenue algorithmiquement. Parmi de telles mesures de similarité, citons par exemple, la distance de Levenshtein (ou distance d'édition), le coefficient de Dice, l'indice de Jaccard, la distance euclidienne, le cosinus, etc.

### **4) Identification de l'intention**

Dans la compréhension du langage parlé, les fonctions / actes de dialogue (DA) sont souvent spécifiques à un domaine [30]. En d'autres termes, au lieu de demander si la fonction de l'énoncé de l'utilisateur est une question ou une réponse, nous demandons si la fonction est, par exemple, de trouver des vols ou d'annuler une réservation dans un programme de réservation de vol. Les actes de dialogue spécifiques au domaine sont appelés intentions. L'identification de l'intention a été la plus utilisée par les robots des centres d'appels, qui demandent à l'utilisateur «comment puis-je vous aider?» et ensuite utiliser l'intention identification pour rediriger l'utilisateur vers l'une des N options de redirection prédéfinies. Un grand nombre des mêmes algorithmes d'apprentissage automatique utilisés pour la classification DA sont utilisés pour l'identification de l'intention.

#### **2.1.2.2 Extraction d'information**

La responsabilité première de la compréhension du langage parlé n'est pas seulement de comprendre la fonction de la phrase, mais de comprendre la signification du texte lui-même. Pour extraire le sens du texte, nous convertissons le texte non structuré, soit la sortie de la RV, soit le texte écrit dans un assistant texte uniquement, en objets de données grammaticales structurés de type (XML, JSON, etc.), qui seront ensuite traités par le gestionnaire de dialogue. La première étape de ce processus consiste à décomposer une phrase en tokens qui représentent chacun de ses composants: mots, signes de ponctuation, nombres, etc. La tokenisation est

difficile en raison de la fréquence des entrées ambiguës ou mal formées, notamment: les phrases (New York), les contractions (ne le **sont** pas), les abréviations (Dr.), les périodes du système horaire de 12 heures (a.m et p.m), etc. Ces tokens peuvent être analysés à l'aide d'un certain nombre d'approches, décrites ci-après, pour créer un certain nombre de structures de données différentes qui seront traitées par le gestionnaire de dialogue [24].

## 1) Approches symboliques

### a) La représentation par sac de mots (Bag of words)

Dans cette approche, nous ignorons la structure, l'ordre et la syntaxe des phrases et comptons le nombre d'occurrences de chaque mot. Nous l'utilisons pour former un modèle d'espace vectoriel, dans lequel les mots vides (par exemple a, le, etc.) sont supprimés et les variantes morphologiques (parler, parlé, parlait, etc.) passent par une lemmatisation des appels de processus et sont stockées sous forme de lemme de base (parler). Dans la phase du gestionnaire de dialogue, en supposant un assistant basé sur des règles, ces mots résultants seront comparés aux documents stockés dans la base de connaissances de l'assistant pour trouver les documents avec des entrées contenant des mots-clés similaires. Cette approche est simple car elle ne nécessite pas de connaissance de la syntaxe, mais, pour cette même raison, elle n'est pas assez précise pour résoudre des problèmes plus complexes. Cette approche présente aussi comme inconvénient la difficulté de bien délimiter les mots de certaines langues comme l'arabe.

### b) Expressions régulières

Les phrases/énoncés peuvent être traités comme des expressions régulières basées sur le concept de « pattern » (en français, « patron ») et peuvent être mises en correspondance avec les documents de la base de connaissances de l'assistant. Il existe plusieurs types / générations de moteurs d'expressions régulières. Il y a des variations syntaxiques, tout dépend de ce que nous volons matcher.

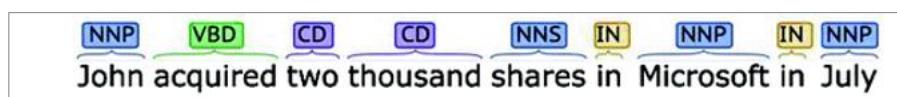
Par exemple, si nous utilisons Python, un des documents de la base de données de connaissances de l'assistant gère le cas où l'utilisateur veut récupérer le nom de l'application, la syntaxe sera "ouvrir l'application .\*". Le méta-caractère .\* indique que cette expression régulière doit être déclenchée chaque fois que l'assistant entend la phrase «ouvrir l'application» suivie de quoi que ce soit. Si l'utilisateur dit «ouvrir l'application météo», cette phrase sera analysée en un certain nombre d'expressions régulières, y compris «ouvrir l'application .\*» et déclenchera la récupération de ce document.

### c) Étiquetage en parties du discours (POS tagging)

L'étiquetage en parties du discours (en anglais « POS tagging » pour Part-Of-Speech tagging) permet d'étiqueter chaque mot de la chaîne d'entrée avec sa partie du discours (par exemple nom, verbe, adjectif, etc.). Ces étiquettes peuvent être basées sur :

- des règles (un ensemble de règles créé manuellement pour spécifier une partie du discours pour les mots ambigus compte tenu de leur contexte) ;
- des automates à états finis, de type transducteurs à états finis – FST – ;
- expressions régulières à base de patrons d'extraction ;
- des grammaires d'extraction contextuelles ;
- des cartouches de règles.

Ils peuvent également être créés en utilisant des modèles stochastiques qui s'entraînent sur des phrases étiquetées en parties du discours correctes. Dans le gestionnaire de dialogue, le POS tagging peut être utilisé pour stocker des informations pertinentes dans l'historique du dialogue. Le POS tagging est également utilisé dans la génération de réponses pour indiquer la partie du discours de la réponse souhaitée.



**Figure 2.2** Étiquetage POS d'une phrase

### d) Reconnaissance d'entités nommées / relations entre entités nommées

Dans la reconnaissance d'entités nommées, les noms de personnes, de lieux, d'organisations, les expressions numériques et les expressions temporelles sont extraits et étiquetés en conséquence dans les phrases. L'extraction d'entités nommées se base sur les mêmes techniques que le POS tagging. En effet, les paires de d'entités nommées peuvent être stockées par le gestionnaire de dialogue dans l'historique des dialogues pour garder une trace du contexte de la conversation de l'assistant. L'extraction des relations entre entités nommées va plus loin dans les relations identitaires (par exemple, «qui a fait quoi, à qui»).



**Figure 2.3** Extraction d'entités nommées d'une phrase

### e) Étiquetage du rôle sémantique

Les arguments d'un verbe sont étiquetés en fonction de leur rôle sémantique (par exemple, sujet, thème, etc.). Dans ce processus, le prédicat (verbe ou groupe verbal) est étiqueté en premier, suivi de ses arguments. Des classifieurs importants pour l'étiquetage des rôles sémantiques ont été formés sur FrameNet et PropBank, des bases de données avec des phrases déjà étiquetées avec leurs rôles sémantiques. Ces paires sémantiques de mots de rôle peuvent être stockées par le gestionnaire de dialogue dans l'historique du dialogue pour garder une trace du contexte.

### f) Création de structures de données grammaticales

Les phrases et les énoncés peuvent être stockés de manière structurée dans des formalismes grammaticaux tels que des grammaires sans contexte (CFG), qui s'appuient sur des règles qui concernent des catégories syntaxiques et des grammaires de dépendance (DG). Les grammaires sans contexte sont des structures de données arborescentes qui représentent des phrases comme contenant des phrases nominales et des phrases verbales, chacune contenant des noms, des verbes, des sujets et d'autres constructions grammaticales. Les grammaires de dépendance, en revanche, se concentrent sur les relations entre les mots.

Les approches symboliques, créées à partir de la connaissance de la situation spécifique en question, ont été utilisées pour extraire un sens structuré d'une phrase ou d'un énoncé. Cependant, ces modèles ont deux problèmes. Premièrement, ils sont coûteux en développement, car de nouveaux modèles doivent être construits avec chaque nouveau système. Deuxièmement, ces modèles ne sont pas robustes souvent aux diverses entrées des utilisateurs, car les utilisateurs ne savent pas quelles structures grammaticales sont gérées par un système spécifique.

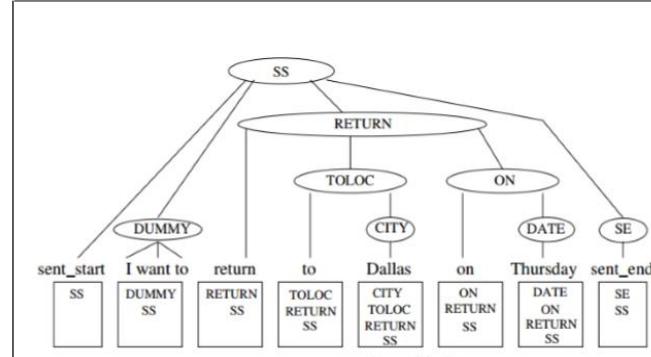
## 2) Approches par apprentissage

Pour résoudre les problèmes des approches symboliques, des modèles statistiques à base d'apprentissage fondés sur les données sont apparus. Ces modèles dérivent la signification M d'un énoncé U qui maximise P (M | U).

### a) État vectoriel caché (Hidden Vector State / HVS)

[31] proposent un modèle statistique à vecteur caché. Étant donné une phrase, l'objectif est de produire automatiquement une signification structurée précise. Considérons la phrase «Je veux retourner à Dallas jeudi». L'arbre d'analyse ci-dessous montre une façon de représenter la signification structurée de la phrase.

SS représente le nœud initial send\_start et SE représente le nœud final send\_end. Nous considérons chaque nœud feuille comme un état vectoriel, décrit par ses nœuds parents: l'état vectoriel de Dallas est [CITY, TOLOC, RETURN, SS].



**Figure 2.4** Arbre d'analyse de HVS [31]

L'arbre d'analyse complet peut alors être considéré comme une séquence d'états vectoriels, représentée par la séquence de carrés ci-dessus. Si chaque état vectoriel est considéré comme une variable cachée, alors la séquence des états vectoriels (par exemple les carrés ci-dessus) peut être considérée comme un modèle de Markov caché, tels que nous commençons à SS et avons certaines probabilités d'atteindre un certain nombre d'états cachés possibles comme l'état suivant. Chaque état vectoriel peut être considéré comme un «automate déroulant» ou une pile. De cette façon, le premier état vectoriel est SS et après cela, nos transitions sont une combinaison d'éléments à pousser ou à tirer pour passer à l'état vectoriel suivant. À chaque étape, nous pouvons effectuer jusqu'à n changements de pile lors de nos transitions. Si nous ne limitons pas n, l'espace d'état augmentera de façon exponentielle (par exemple, n états possibles à partir de S,  $n^2$  états possibles à l'étape suivante, etc.). Pour éviter cela, nous limitons la profondeur maximale de la pile.

Avec ce modèle, [31] soutiennent que la probabilité qu'un ensemble de mots W se produise avec un ensemble correspondant de concepts C et un ensemble d'opérations de transition N, est le produit de :

- la probabilité que la  $t^{\text{ème}}$  opération de pile  $N_t$  se produise étant donné le dernier concept  $C_{t-1}$  ;
- la probabilité que le  $t^{\text{ème}}$  concept du nœud racine  $C_t[1]$  se produise étant donné la séquence de concepts précédents  $C_t[2 \dots D_t]$  ;
- la probabilité que le  $t^{\text{ème}}$  mot  $W_t$  apparaisse étant donné le  $t^{\text{ème}}$  concept  $C_t$ .

pour tout  $t = 1 \dots T$ , où T est l'état final du modèle.

$$P(N, C, W) = \prod_{t=1..T} P(n_t | c_{t-1}) P(c_t[1] | c_t[2..D_t]) P(w_t | c_t)$$

**Équation 2.5** Fonction qui calcule la probabilité qu'un ensemble de mots W se produise avec un ensemble de concepts C et un ensemble d'opérations de transition N [31]

Pour entraîner le modèle, nous créons un ensemble de paires de phrases et leurs annotations de concept structurées et ajustons le paramètre de modèle  $\lambda$  qui nous permet de convertir des phrases en concepts. En effet, pendant entraînement, nous cherchons à maximiser :

$$L(\lambda) = \log P(N, C, W | \lambda)$$

**Équation 2.6** Fonction d'entraînement du modèle de *He et Young* [31]

Lorsque nous exécutons le modèle sur des phrases invisibles (par exemple dans la prédiction / classification), nous appliquons les paramètres du modèle dérivés lors de l'entraînement pour générer la sortie souhaitée (les annotations de concept structurées appropriées).

D'autres modèles similaires incluent: le modèle de transducteur à états finis stochastique, qui utilise des machines à états finis pour déterminer le concept grammatical approprié avec lequel il est possible d'étiqueter un mot donné et des modèles de réseaux bayésiens dynamiques. Tous ces modèles sont des modèles génératifs, i.e. ils cherchent à trouver la distribution de probabilité conjointe  $P(X, Y)$  où X représente les entrées de phrase et Y représente les sorties de concept grammatical structuré. Dans ce qui suit, nous discutons le discriminatif qui calcule la probabilité conditionnelle de  $P(Y | X)$  afin de faire correspondre des phrases avec des concepts.

### b) Modèle de machine à vecteur de support (Support Vector Machine model / SVM)

Les machines à vecteur de support sont un outil d'apprentissage automatique supervisé [32]. Étant donné un ensemble de données d'apprentissage étiquetées, l'algorithme génère l'hyperplan optimal qui divise l'échantillon en leurs propres étiquettes. Traditionnellement, les SVM sont considérés comme résolvant des problèmes de classification binaire, mais plusieurs hyperplans peuvent être utilisés pour diviser les données en plus de deux catégories d'étiquettes. L'hyperplan optimal est défini comme l'hyperplan qui crée la marge ou la distance maximale entre des ensembles de points de données étiquetés différemment. Classiquement, un hyperplan est représenté par  $H(X) = W * X + B = 0$ , où X est les points d'entrée, W est un vecteur de

poids et  $B$  est le terme de biais. Les SVM sont utilisés pour atteindre la compréhension du langage parlé en le traitant comme une séquence de problèmes de classification binaire et en faisant correspondre les mots avec les concepts.

**c) Modèles à champ aléatoire conditionnel (Conditional Random Field models / CRFs)**

Les CRFs sont des modèles statistiques log-linéaires souvent appliqués pour la prédiction structurée. Contrairement au classifieur moyen, qui prédit une étiquette pour un seul objet et ignore le contexte, les CRFs prennent en compte les caractéristiques précédentes de la séquence d'entrée en utilisant des probabilités conditionnelles. Un certain nombre de fonctionnalités différentes peuvent être utilisées pour entraîner le modèle, notamment des informations lexicales, des préfixes et des suffixes, des majuscules et d'autres fonctionnalités.

**d) Deep learning (l'apprentissage profond)**

L'avancement le plus récent dans l'utilisation de modèles statistiques pour la prédiction de concept/structure est le deep learning pour le traitement du langage naturel. Les architectures de réseaux de neurones d'apprentissage profond diffèrent des réseaux de neurones traditionnels. En effet ils utilisent plus de couches cachées. Chaque couche gère des fonctionnalités de plus en plus complexes. En conséquence, les réseaux peuvent apprendre des modèles et des données non étiquetées et l'apprentissage profond peut être utilisé pour un apprentissage non supervisé. Des méthodes d'apprentissage profond ont été utilisées pour générer un étiquetage en parties du discours de phrases (bloc de texte en phrases nominales, phrases verbales, etc.), pour la reconnaissance d'entités nommées et pour l'étiquetage sémantique des rôles.

L'un des inconvénients de ces modèles est qu'ils nécessitent un corpus de données conséquent, ce qui n'est pas évident de trouver surtout pour une langue comme l'arabe. Par ailleurs, il faut garder à l'esprit que les résultats obtenus grâce à une tel modèle sont directement tributaires de la qualité du corpus d'apprentissage. Ce dernier se doit donc d'être représentatif.

### 2.1.2.3 Génération de réponses

La génération de réponses est sans doute le composant le plus central de l'architecture de l'assistant. En entrée, le générateur de réponse (GR) reçoit une représentation structurée du texte parlé. Cela transmet des informations sur qui parle, l'historique du dialogue et le contexte. En sortie, le GR génère une réponse à remettre à l'utilisateur, qu'il fournira au gestionnaire de

dialogue. Le sélectionneur de réponse a accès à trois éléments clés qu'il utilisera pour prendre sa décision sur ce qu'il doit dire:

- 1) une base de connaissances / corpus de données, dont le contenu différera en fonction de la mise en œuvre ;
- 2) un corpus d'historique de dialogue, qui n'existe que dans des modèles plus complexes ;
- 3) une source de données externe, qui fournit à l'assistant une connaissance (par exemple, un chat est un animal). Cette connaissance est souvent obtenue en permettant aux assistants d'accéder et de récupérer des documents à partir des moteurs de recherche.

Dans cette section, nous discuterons en détail des façons dont un assistant virtuel peut récupérer une réponse.

### 1) Modèles basés sur les règles

Dans l'optique d'une approche symbolique, l'idée principale derrière le RG basé sur des règles est qu'un assistant contient une base de connaissances avec des documents, où chaque document contient un <pattern> et un <template>. Lorsque l'assistant reçoit une entrée qui correspond au <pattern>, il envoie le message stocké dans le <template> en tant que réponse. Le <pattern> peut être soit une phrase, comme "Quelle application voulez-vous lancer?", soit un patron comme, «Je veux lancer l'application .\*», où le patron représente une expression régulière. En général, ces paires <pattern> <template> sont conçues manuellement [33].

Le défi principal est de choisir un algorithme à utiliser pour la mise en correspondance de pattern entre la phrase d'entrée et les documents dans le corpus de données. Nous pouvons considérer cela comme un problème de voisinage le plus proche, où notre objectif est de définir la fonction de distance et de récupérer le document le plus proche de la phrase d'entrée. Ci-dessous, nous présentons les méthodes les plus courantes.

#### a) ELIZA (Analyse incrémentale)

Un des premiers assistants créé au MIT entre 1964 et 1966, a utilisé une analyse incrémentale ou une approche de «correspondance directe» pour la correspondance de patrons. ELIZA analyse le texte saisi mot par mot de gauche à droite, en recherchant chaque mot dans le dictionnaire, en lui attribuant un rang d'importance et en le stockant dans une pile de mots-clés. Le mot-clé ayant le rang d'importance le plus élevé serait traité comme un <pattern> et le pattern correspondrait aux documents du corpus pour trouver la réponse de modèle appropriée. S'il n'existe aucun document correspondant à l'entrée, ELIZA dirait «Je vois» ou «Veuillez

continuer» dans le script DOCTOR. ELIZA a été créée avec plusieurs «scripts» qui indiquent différentes réponses <template> aux entrées <pattern> [34].

### b) ALICE (Technique de match direct)

L'architecture ALICE comprend une base de connaissances appelée Graphmaster, qui est un graphe avec des nœuds «nodemappers» et des arêtes. Graphmaster peut être considérée comme un système de fichiers, avec une racine contenant des fichiers et des répertoires. Le nom du chemin d'accès à chaque nœud feuille est la phrase <pattern> de ce nœud feuille. Nous effectuons un filtrage de patrons en utilisant une recherche en profondeur sur la base de connaissances [35][36][37].

### c) VPBot (Technique de correspondance des ensembles de mots-clés)

L'analyse incrémentale et la correspondance directe ne nécessitent qu'un seul mot pour correspondre afin de récupérer une réponse (par exemple, une correspondance de mot-clé dans le premier et une correspondance directe dans le second). Cette flexibilité pourrait conduire à un comportement inattendu. VPBot a modifié cela en une technique permettant aux développeurs d'attribuer entre 1 et 3 mots-clés à un ensemble de mots-clés. Cette technique a exigé que tous les mots-clés de l'ensemble soient présents pour déclencher une réponse.

Dans l'optique d'une approche symbolique, le principal problème avec les systèmes basés sur des règles est que le programmeur doit spécifier toutes les paires «règles» ou <pattern> <template>. Avec l'arrivée de grands ensembles de données tels que les dialogues sur Reddit et Twitter, cela n'est plus devenu nécessaire. En effet, les développeurs pourraient analyser des millions de conversations et stocker ces conversations dans des documents sous forme de paires de <status> <response> pour sélectionner une réponse.

## 2) Modèles basés sur la recherche d'information

Les systèmes basés sur la recherche d'information consistent à faire correspondre ce modèle à l'ensemble des paires <status> <response> à partir d'une phrase d'entrée et de sélectionner une réponse.

Le principal défi des algorithmes de recherche d'information (RI) est de déterminer comment effectuer la correspondance de pattern. Tout d'abord, il faut prendre en considération, étant donné une phrase d'entrée, le fait de correspondre le pattern avec <status> ou <response>. Le premier semble plus intuitif et impliquerait de trouver le <status> le plus similaire dans le

corpus de données. Ce dernier, cependant, est utilisé plus souvent en pratique car il est plus efficace et implique de comparer la phrase d'entrée avec les <responses> dans le corpus de données. La raison pour laquelle faire correspondre à <responses> est plus efficace est que si le mot co-occurre à une phrase d'entrée et une <response>, c'est probablement une bonne réponse. Par contre, si un mot co-occurre à une phrase d'entrée et <status> et si ce n'est pas une correspondance exacte, nous ne pouvons pas savoir si <response> est une bonne réponse. De même, si le système patron correspond aux réponses et non pas aux entrées, il peut simplement utiliser les moteurs de recherche pour avoir un ensemble de réponses similaires à la phrase d'entrée (par exemple, les moteurs de recherche récupèrent des ensembles de réponses). Indépendamment de la correspondance à <status> ou <response>, nous devons choisir un algorithme de correspondance des patrons. Ici, Nous pourrions utiliser une approche par mots-clés de base, dans laquelle nous recherchons des mots-clés qui co-ocurrent à la phrase d'entrée et le document, cette méthode est parfois utilisée [38].

Cependant, les modèles de RI les plus complexes utilisent généralement un ensemble de fonctions de score pour calculer le classement global de chaque document en tant que moyenne pondérée de ces scores et renvoient le document ayant le score le plus élevé. Étant donné un status (S) - la phrase d'entrée - et un corpus de documents (D) dans la base de connaissances, ces algorithmes renvoient :

$$Rank(S, D) = \sum_k \lambda_k * score_k(S, D)$$

**Équation 2.7** Fonction de classement global des documents [38]

Dans cette formule, la sortie de la  $k^{\text{ième}}$  fonction de score  $score_k(S, D)$  est multipliée par le poids reçu par cette fonction de score ( $\lambda_k$ ). La somme de toutes les fonctions de score donne le classement général de chaque document.

Les modèles de recherche d'information sont trop complexes pour être exécutés sur chaque réponse candidate et nécessitent une base de données de réponses. [39] ont utilisé une approche de filtrage basée sur des règles pour sélectionner les cinq premières réponses candidates. Ces réponses sont exécutées via le réseau de neurones récurrent (RNN) et reclasées pour sélectionner la première réponse.

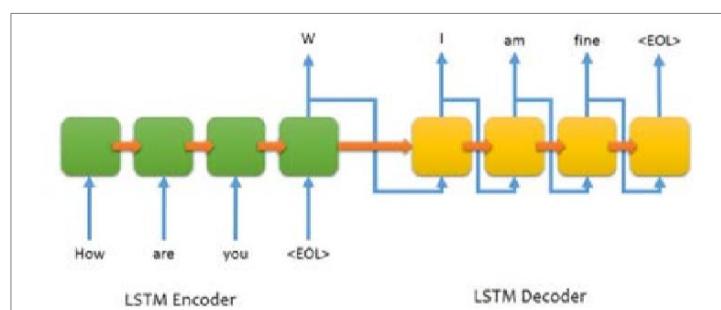
### 3) Modèles génératifs de traduction

Les modèles de recherche d'information nécessitent une base de données de réponses possibles parmi lesquelles choisir. Les modèles génératifs, en revanche, construisent des réponses à la volée. La plupart le font à l'aide de techniques d'apprentissage automatique : les classificateurs sont formés sur la base de véritables dialogues et utilisés pour générer des réponses aux utilisateurs en traduisant les entrées en réponses. Les modèles de traduction automatique statistique (SMT) sont parmi les modèles les plus récents et les plus efficaces utilisés pour générer des réponses de l'assistant.

La traduction automatique statistique est un domaine de la traduction automatique qui analyse les corpus bilingues de texte et utilise l'analyse statistique pour dériver des transactions exactes entre des mots, des phrases et des caractéristiques individuelles des textes. L'avantage de la traduction automatique par rapport à la traduction humaine est avant tout des économies de ressources. De plus, SMT a accès à de nombreux corpus parallèles permettant aux analyses de générer des traductions entre plusieurs langues. SMT génère des économies de coûts par rapport aux autres approches automatisées dans la mesure où les systèmes de traduction basés sur des règles nécessitent la création manuelle de règles. Ceci est coûteux et crée des règles qui ne fonctionnent généralement que pour une paire de langues car elles ne se généralisent pas bien pour les autres traductions de langues [40].

### 4) Modèles séquence à séquence (Seq2Seq)

C'est une variante du modèle génératif qui a été proposée par [41] en 2014. Elle utilise les avancées du deep learning pour atteindre une plus grande précision. Leur modèle encodeur-décodeur utilise deux réseaux de neurones récurrents (RNN). Le premier encode le <status>, ou phrase d'entrée et le second décode le <status> et génère la <response> souhaitée. Le modèle est capable de traduire entre les langues (traduction automatique statistique) et peut également être utilisé pour permettre aux assistants de convertir entre le document <status> et le document <response>. Le modèle Seq2Seq est la meilleure pratique actuelle pour la génération de réponses et est largement utilisé.



**Figure 2.5** Exemple de Seq2Seq RNN [41]

Les réseaux de neurones récurrents (RNN) sont des réseaux de neurones avec un état caché  $h$  et une sortie  $y$ , qui est fonction d'une séquence  $x_1 \dots x_n$ . Dans notre cas, la séquence est la chaîne d'entrée de l'assistant. À chaque pas de temps, l'état caché du RNN est mis à jour par la fonction ci-dessous.

$$h_t = f(h_{t-1}, x_t)$$

**Équation 2.8** Fonction de māj de l'état caché du RNN [41]

$f$  est une fonction d'activation telle que la fonction sigmoïde ou l'unité de mémoire à long terme (LSTM).

Le but de l'encodeur RNN est de prédire la fonction suivante :  $P(y_1 \dots y_{tx} | x_1 \dots x_{tx})$ , tels que  $x_1 \dots x_{tx}$  est la séquence d'entrée,  $y_1 \dots y_{ty}$  est la séquence de sortie et  $tx$  et  $ty$  sont respectivement la longueur de l'entrée et la longueur de la réponse, ces dernières peuvent différer. À chaque étape, l'état caché du RNN est mis à jour jusqu'à ce qu'il atteigne  $x_{tx}$  et contienne des informations relatives à la chaîne entière que nous appellerons  $c$ . Le but du décodeur RNN est l'inverse, il prédit la chaîne de réponse en fonction de la chaîne encodée de l'état caché créée par l'encodeur et des informations finales sur la chaîne d'origine  $c$ . La fonction cachée du décodeur n'est qu'une fonction :

$$h_t = f(h_{t-1}, y_{t-1}, c)$$

**Équation 2.9** Fonction cachée du décodeur [41]

Les deux RNN sont entraînés pour maximiser la probabilité que le modèle produise la séquence de réponse  $y$ , étant donné la séquence d'entrée  $x$ , comme le montre l'équation ci-dessous.

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^N \log P_{\theta}(y_n | x_n)$$

**Équation 2.10** Fonction pour maximiser la probabilité de production d'une séquence de réponse étant donné une séquence d'entrée [41]

$\theta$  sont les paramètres du modèle. Le modèle est entrainé sur des dialogues réels avec des paires <input><response>. Les RNN résultants pourraient être utilisés pour deux fonctions: dans SMT, ils pourraient être utilisés pour générer une traduction <response> d'une entrée, c'est ainsi qu'ils sont utilisés ici. Deuxièmement, ils pourraient être utilisés pour évaluer une réponse dans la

mesure où elle correspond à une entrée, c'est ainsi que Seq2Seq est utilisé dans les processus de recherche d'information.

L'approche de [41] est maintenant la base de la plupart des générations de réponses SMT des assistants, mais elle est construite sur une longue histoire de techniques SMT basées sur les réseaux neuronaux. *Schwenk et al.*, en 2012, ont utilisé des réseaux de neurones à réaction avec une entrée limitée à 7 mots pour générer des réponses. *Devlin et al.*, en 2014, ont avancé l'utilisation des réseaux de neurones à réaction directe, mais ont maintenu la nécessité de limiter la longueur d'entrée a priori. Seq2Seq est largement utilisé dans la génération de réponses car il ne nécessite pas de longueur d'entrée définie et offre des performances plus élevées que les modèles précédents.

Bien que le modèle soit efficace, Seq2Seq présente un certain nombre d'inconvénients.

Premièrement, Seq2Seq entraîne le complexe codeur-décodeur pour générer la meilleure réponse à un seul <input> en maximisant la probabilité conditionnelle log.

$$\max_{\theta} \frac{1}{N} \sum_{n=1}^N \log P_{\theta}(y_n | x_n)$$

**Équation 2.11** Fonction pour maximiser la probabilité conditionnelle log [41]

Bien que cette fonction offre une approximation d'une bonne réponse, elle n'atteint pas le véritable objectif de l'assistant : simuler une interaction homme-homme en «fournissant des commentaires intéressants, diversifiés et informatifs qui maintiennent l'engagement des utilisateurs». Par exemple, les robots répondront fréquemment aux entrées avec la phrase "Je ne sais pas" en raison de la fréquence à laquelle cette réponse est utilisée dans les ensembles d'entraînement. Deuxièmement, Seq2Seq reste souvent bloqué dans des boucles infinies, comme le montrent [42] dans des simulations de conversations entre deux assistants Seq2Seq.

## 5) Reinforcement learning (apprentissage par renforcement) avec Seq2Seq

Pour résoudre ces problèmes, nous devons définir des fonctions de récompense à long terme qui encouragent les assistants à prendre des mesures qui mèneront à de bonnes conversations tels que l'apprentissage par renforcement. Les approches d'apprentissage par renforcement définissent le dialogue comme un processus de décision de Markov avec un espace d'états S, des transitions T, un ensemble d'actions A et une fonction de récompense R [43].

#### 2.1.2.4 Création d'une base de connaissances

Plus la base de connaissances est grande, plus l'assistant est intelligent. De ce fait, la collecte de données d'entraînement utilisées pour former des classifieurs d'apprentissage automatique est essentielle pour réaliser des interactions entre l'humain et la machine.

Faire progresser les corpus de données utilisés par les assistants est un complément au développement d'algorithmes pour améliorer la précision de ces assistants. Cette section se concentre sur les méthodes de collecte de données pour les assistants. Initialement, les approches basées sur des règles étaient complétées par des bases de connaissances créées manuellement. Plus tard, les développeurs ont commencé à utiliser de grands ensembles de données de dialogue annotés par l'homme. Ces dernières années, le raclage de millions de dialogues en ligne et la création automatisée de bases de connaissances utilisées par les assistants ont beaucoup augmenté.

##### 1) Corpus annotés par l'homme

Reconnaissant la valeur des corpus de dialogue déjà existants, [44] ont développé des algorithmes pour convertir des dialogues annotés par l'homme en XML, AIML et d'autres formats compatibles avec les assistants. Leurs programmes comportent quatre étapes de base:

- 1) lire les dialogues à partir des corpus existants ;
- 2) supprimer tous les textes qui se chevauchent (par exemple, deux orateurs parlent à la fois) et les éléments de remplissage non verbaux ;
- 3) traiter le texte pour considérer chaque première ligne dans un dialogue comme <pattern> et la deuxième ligne comme <template> ;
- 4) transformer ces dialogues en fichiers XML, AIML, etc. ou en d'autres formats compatibles avec les assistants afin que ces dialogues puissent être utilisés pour former des bases de connaissances.

##### 2) Forums de discussion

[45] ont généré le premier forum de connaissances sur l'assistant extrait des forums de discussion en ligne. L'avantage d'utiliser les forums est qu'ils contiennent différentes pages de thèmes de discussion, ce qui permet d'accéder à des dialogues sur un large choix de sujets. Ainsi, ces dialogues peuvent impliquer plusieurs réponses à la même question, que l'assistant peut traiter comme des paires <input> <response> séparées. L'inconvénient est que :

- 1) les forums de discussion sont des communications asynchrones, durant lesquelles les utilisateurs ne répondent pas en temps réel aux requêtes des autres comme l'exigent les robots, ce qui peut affecter le style de communication ;
- 2) le contrôle de la qualité est un problème important, tels que les réponses sur les forums sont rédigées, brèves et peuvent contenir des erreurs ;
- 3) il n'y a aucune garantie que la <response> soit une réponse à l'entrée <input>.

### **3) Conversations par e-mail**

[46] ont utilisé la conversation par e-mail pour extraire des paires question-réponse. Cette approche bénéficie de la prévalence des conversations par e-mail, mais présente un certain nombre d'inconvénients, notamment :

- les longs monologues utilisés ;
- les e-mails sont asynchrones contrairement au chat ;
- les différences stylistiques dans le chat par rapport à la communication par e-mail ;
- le manque de clarté quant aux paires question-réponse spécifiques dans les e-mails plus longs.

#### **2.1.3 Approches pour la synthèse vocale**

La synthèse vocale (SV) est la dernière étape du processus et convertit la réponse générée en parole, qui est renvoyée à l'utilisateur. La première étape de la synthèse vocale est l'analyse de texte, dans laquelle le texte est converti en phonèmes avec hauteur et durée. La deuxième étape est la synthèse de forme d'onde, dans laquelle des segments de parole enregistrée correspondant à chaque phonème sont concaténés pour former la parole.

##### **2.1.3.1 Analyse de texte**

L'analyse de texte commence par un processus de normalisation dans lequel le texte est divisé en composants, les fragments de texte sont divisés en phrases, les phrases en mots et ponctuations et les mots en phonèmes [47]. La segmentation des phrases implique la recherche d'une ponctuation de fin de phrase typique, telle que le point d'exclamation, le point d'interrogation, les deux points, les points-virgules ou le point. Le défi est que ces ponctuations ne marquent pas toujours la fin d'une phrase. Pour résoudre ce problème, les classificateurs d'apprentissage automatique sont entraînés sur du texte pour apprendre la différence entre les ponctuations de fin de phrase et les ponctuations sans fin de phrase. La normalisation comprend également la conversion de mots non standards en langage naturel. Les exemples incluent des

nombres, des abréviations et des acronymes qui peuvent être prononcés de différentes manières. Enfin, la normalisation implique la levée de l'ambiguïté des homographes, ou des mots qui sont prononcés différemment et qui contiennent une signification différente en fonction du contexte. Pour ces mots (par exemple «J'ai utilisé ma voiture» ou «J'ai mis ma voiture en utilisation»), une analyse de la partie du discours (POS) peut être exécutée pour marquer la partie du discours de l'homographe. La deuxième étape de l'analyse de texte est la prononciation. Cela implique l'accès à un lexique de prononciation qui comprend des correspondances entre les mots / phonèmes et leurs prononciations, ainsi que des lexiques de noms, qui associent les noms à leurs prononciations. Le processus de mise en correspondance de caractère avec phonème était initialement basé sur des règles, mais il a progressé en utilisant aujourd'hui des algorithmes avancés pour générer les mises en correspondance les plus précises en fonction du contexte. [48] ont utilisé des arbres de décision pour le processus et [49] ont entraîné un réseau neuronal feed-forward. La prononciation seule conduirait le discours à ressembler à une chaîne de mots. Un certain nombre de classificateurs d'apprentissage automatique ont été développés et entraînés pour fournir une correspondance appropriée entre les caractères et les informations prosodiques.

### 2.1.3.2 La synthèse de forme d'onde (Waveform synthesis)

La synthèse de longueur d'onde est la voix des sons générés dans l'analyse de texte et peut se produire en utilisant un certain nombre de processus différents [47]. La synthèse formée implique un certain nombre de résonateurs connectés en série ou en parallèle qui transmettent la sortie de la production de fréquences l'un à l'autre (généralement trois sont nécessaires pour un son compréhensible et cinq pour une production sonore de haute qualité). La synthèse articulatoire est théoriquement destinée à générer «la parole par modélisation directe du comportement de l'articulateur humain», mais ne produit pas de résultats de haute qualité en pratique. La synthèse concaténative, en revanche, génère de la parole en connectant un certain nombre d'unités vocales préenregistrées. L'unité la plus couramment utilisée est le di-phone qui commence au milieu d'un phonème et s'étend au milieu d'un deuxième, capturant à la fois les sons et la transition. Le problème de la synthèse concaténative est que les unités risquent de ne pas s'aligner. Pour résoudre ce problème, la synthèse de sélection d'unité stocke plusieurs instances de chaque unité, chacune avec une prosodie différente et sélectionne l'unité à livrer en fonction du contexte. Le stockage de plusieurs instances de chaque unité peut être peu efficace. Des modèles statistiques (par exemple des modèles de synthèse HMM) existent pour résoudre ce problème en mettant en correspondance une instance sur la variante avec la prosodie correcte.

**Conclusion :** d'après l'analyse des recherches et des articles connexes, nous avons pu diviser les approches et les méthodes en deux classes principales, les approches symboliques et les approches par apprentissage. Chacune de ces deux classes présente un ensemble d'avantages et d'inconvénients (cités précédemment).

En effet, les approches symboliques sont très coûteuses et complexes à implémenter, cependant, elles fonctionnent indépendamment des données. Quant aux approches par apprentissage, elles sont simples et faciles à implémenter, cependant, elles nécessitent beaucoup de données d'apprentissage et de grandes puissances de calcul.

En conclusion, nous avons constaté que la plupart des travaux récents utilisent les approches par apprentissage (apprentissage automatique et profond) dans les différentes phases de traitement. Cependant, il ne faut pas négliger l'importance et l'utilité des approches classiques symboliques dans certains cas.

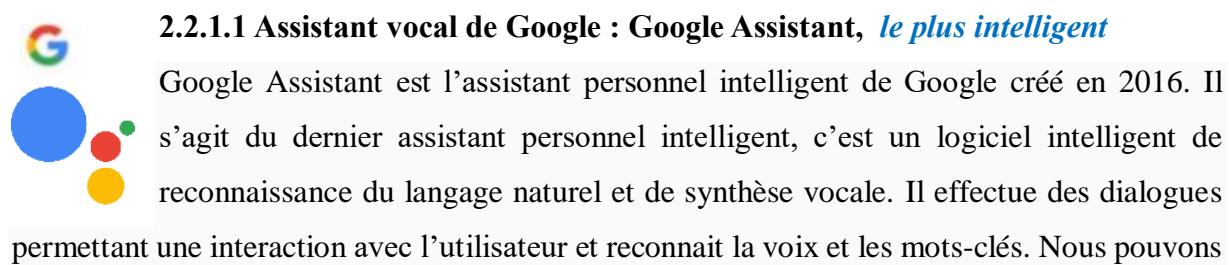
Dans la conception de notre solution, nous essayerons de collecter les avantages de chaque approche pour construire un système hybride capable de donner de meilleurs résultats.

## 2.2 Analyse de l'existant

### 2.2.1 Revue de quelques applications d'assistance vocale disponibles sur le marché

La majorité des assistants vocaux virtuels ont des tâches et fonctionnalités communes, néanmoins, nous trouvons toujours au moins une caractéristique ou fonctionnalité qui distingue un assistant donné des autres assistants.

De ce fait, le passage en revue certains assistants vocaux virtuels qui occupent le marché actuel est une étape nécessaire [4], afin de dégager les points forts et les points faibles de chacun, dans le but d'intégrer dans notre assistant le maximum de leurs points forts et faire en sorte de corriger dans la mesure du possible, leurs points faibles.



communiquer avec l'assistant soit oralement, il suffit de commencer la requête par le mot magique « Ok Google » pour activer son écoute, soit par écrit, en saisissant le texte, soit la question sur le tchat. Sa compatibilité avec le moteur de recherche le plus populaire « Google » a fait de l'assistant vocal la meilleure option pour obtenir des réponses précises aux questions. Ainsi, l'avancement très vite de sa technologie et son évolution en permanence fait de l'assistant de Google un investissement à long terme.

Google Assistant est présent sur plus de 500 millions de dispositifs dans le monde : téléphones Android, enceintes intelligentes, téléviseurs, véhicules, casques audio, montres, etc. Il est compatible avec : Google Home Mini, Assistant Google Home ou Assistant vocal Google Home, Google Home Max, Android et iPhone via l'application à télécharger sur Google Assistant et plus de 1000 autres services, marques et applications. Ainsi, il prend en charge 4 langues : l'anglais, le français, l'allemand et le japonais.

**Points forts** : intelligence, précision, mobilité, facilité d'installation sous Android, évolution constante, excellente compatibilité avec Google et YouTube, exécution simultanée de 2 commandes depuis un Assistant Google Home, mais le plus grand avantage caractérisant cet assistant est qu'il est le plus intelligent en le comparant aux autres assistants existants.

**Points faibles** : la plus grande limite de Google Assistant que nous jugeons importante est qu'il ne supporte pas la langue arabe, mais aussi son incapacité d'utilisation en étant hors ligne, sa faible compatibilité avec des appareils et services domotiques et ses réponses ne sont pas toujours pertinentes.

### 2.2.1.2 Assistant vocal d'Amazon : Alexa, *la meilleure compatibilité*



Alexa est l'assistant personnel vocal intelligent d'Amazon lancé en novembre 2014. Il s'agit d'un logiciel modelé permettant d'effectuer des dialogues avec un appareil, afin d'obtenir des réponses, des informations, des services et des actions sur d'autres objets connectés par le biais de la voix. À sa sortie, Alexa était présent uniquement dans les enceintes intelligentes sur la gamme d'appareils *ECHO* d'Amazon, leurs fonctions varient selon les modèles (DOT, PLUS, DOT KIDS EDITION, SPOT, SHOW, LOOK) et ont la capacité d'obéir à la voix humaine, de dialoguer et d'interagir avec l'utilisateur, ainsi ces appareils peuvent être connectés à des objets domotiques pouvant être contrôlés par la voix. De ce fait, de nombreux produits intelligents compatibles intègrent Alexa car il est certainement le

choix pour rendre une maison intelligente, en permettant un contrôle vocal des appareils (smartphones, interrupteurs, prises, ampoules, volets, caméras et vidéosurveillances, électroménagers, serrures et systèmes de sécurité pour les maisons, chauffages, thermostats, hub domestiques, appareils à air-conditionné ou encore automobiles.) depuis notre domicile. Alexa est idéal pour transformer notre maison en smart-home. De plus, à l'ère où l'e-commerce est privilégié par les consommateurs, le dispositif d'Amazon est parfait pour effectuer des achats en ligne dans des meilleures conditions.

Alexa est compatible avec : tous les appareils ECHO d'Amazon, Android et iPhone via l'application de l'assistant vocal Alexa et plus de 7400 autres services, marques et applications. Ainsi, il prend en charge 4 langues : l'anglais, le français, l'allemand et le japonais.

**Points forts :** intelligence, précision, mobilité, meilleure expérience d'achat, mais l'avantage distinct qu'Alexa possède par rapport aux autres assistants est sa compatibilité avec les appareils, les applications et les services domotiques avec une énorme liste plus longue que celle des autres assistants intelligents.

**Points faibles :** le principal inconvénient d'Alexa que nous trouvons important est qu'il ne supporte pas la langue arabe, de plus son incapacité d'utilisation en étant hors ligne, ses réponses ne sont pas toujours pertinentes et l'installation et l'activation d'Alexa sur smartphones n'est pas aussi intuitive que Google et Siri, etc.

#### 2.2.1.3 Assistant vocal d'Apple : Siri, *le meilleur pour l'écosystème Apple*



Siri est l'assistant personnel vocal intelligent d'Apple, c'est le plus ancien et le plus divertissant des assistants. En 2010, Apple a proposé Siri en tant qu'application autonome sur l'*App Store*. Puis en 2011, elle l'a intégré dans le système d'exploitation iOS d'Apple. Il s'agit d'un assistant vocal intelligent personnalisé qui effectue une multitude de tâches et collecte des informations provenant des services d'Apple.

Siri est disponible uniquement sur les appareils Apple, comme l'iPhone, l'iPad, l'Apple Watch, l'AirPod, le MacBook, l'iMac, le haut-parleur HomePod, ainsi que les véhicules ayant Apple CarPlay à bord. Ce dernier nécessite un appareil compatible avec iOS. L'assistant virtuel Siri est donc parfait pour les utilisateurs d'Apple, ce qui fait la différence entre ce dernier et les autres assistants.

Siri prend en charge 21 langues adaptées à 36 pays, nous citons : l'arabe, le chinois, le danois, le néerlandais, l'anglais (Australie, Canada, Inde, Nouvelle-Zélande, Singapour, Royaume-Uni, Etats-Unis), le finnois, le français, l'allemand, le hébreu, l'italien, le japonais, le coréen, le malais, le norvégien, le portugais, le russe, l'espagnol, le suédois, le thaïlandais, le turc et le cantonais.

**Points forts :** intelligence, précision, mobilité, Compatibilité avec Home-Kit (service domotique d'Apple), Capacité multilingue (Support linguistique), mais l'avantage distinct pour Siri est son expérience et son intégralité inégalée avec les dispositifs de l'écosystème d'Apple.

**Points faibles :** l'un des grands soucis qui nous fait rebouter de Siri, c'est bien qu'il supporte la langue arabe mais la façon dont elle est parlée est horrible et pleine d'erreurs. Ajouté à cela, l'incapacité d'utilisation en étant hors ligne, des réponses pas toujours pertinentes, des applications vocales limitées. Ainsi, Siri n'est pas compatible avec des services populaires comme Spotify. Ajoutons à cela, les prix élevés des dispositifs d'Apple, à titre d'exemple HomePod est cinq fois plus cher qu'Amazon Echo et trois fois plus cher que Google Home.

#### 2.2.1.4 Assistant vocal de Microsoft : Cortana, **le meilleur pour les pros**



Initialement intégré à Windows et aux téléphones sous Windows en 2014, Cortana est l'assitant personnel intelligent de Microsoft. C'est un logiciel intelligent de reconnaissance du langage naturel et de synthèse vocale. En 2015, Cortana a été porté sous forme d'une application à télécharger pour les téléphones sous Android. Cortana s'appuie sur le moteur de recherche Bing pour chercher des informations sur Internet. Ainsi, il a un nombre faible d'applications, de services et d'appareils compatibles par rapport aux autres assistants concurrents, comme Google Assistant et c'est la raison pour laquelle Microsoft a annoncé qu'il serait bientôt possible de lancer Google Assistant depuis Cortana. Par ailleurs, Cortana prend en charge 8 langues adaptées à 13 pays (l'anglais, le français, l'allemand, l'espagnol, l'italien, le portugais, le chinois et le japonais).

**Points forts :** intelligence, précision, mobilité, mais le grand avantage de Cortana est que dans le l'avenir proche, Microsoft compte se tourner vers un usage professionnel de son assistant vocal. En effet, son intégration aux applications Microsoft Office 365 en est la preuve. De

plus, afin de rassurer les professionnels, Cortana va leur garantir une meilleure sécurité des données.

**Points faibles :** la principale raison pour laquelle Cortana n'a pas connu le succès escompté auprès du grand public malgré son intelligence est qu'il se limite au système d'exploitation Windows 10 et à l'enceinte connectée *Harman Kardon Invoke*. Un autre point faible réside dans le fait qu'il ne supporte pas la langue arabe. De plus, son incapacité d'utilisation en étant hors ligne et ses réponses pas toujours pertinentes lui portent préjudice.

#### 2.2.1.5 Assistant vocal arabe : RafiQ



RafiQ est une application Android d'assistance vocale qui comprend les commandes vocales arabes. Il est disponible sous forme d'une voix activée et est intégré comme un chatbot dans des applications mobiles, des sites web, des systèmes Interactive Voice Response (IVR) et des chatbots Messenger Facebook. RafiQ soutient la langue arabe et l'arabe dialectal égyptien et peut faire des contrôles téléphoniques ainsi que des intégrations de services tiers.

**Points forts :** mobilité et prise en charge de la langue arabe.

**Points faibles :** la principale raison pour laquelle RafiQ n'est pas trop utilisé est qu'il génère des réponses qui ne sont pas adéquates avec les requêtes des utilisateurs, et ce, à cause de sa construction à base de règles (IA symbolique). RafiQ souffre également de bugs et de problèmes sur beaucoup de smartphones ayant un système Android comme le Samsung A70, le Samsung Grand Prime (d'après les commentaires sur Play Store). RafiQ ne fonctionne pas hors ligne et les informations qu'il donne (météo, journal, etc.) se limitent à l'Égypte.

#### 2.2.1.6 Autres assistants vocaux virtuels en développement

Il existe d'autres assistants virtuels qui sont en cours d'élaboration pour concurrencer Google Assistant, Amazon Alexa et Siri, comme M de Facebook, Célia de Huawei, bixby de Samsung, mais ces derniers ne supportent pas la langue arabe.



Il existe également plusieurs problèmes avec les produits d'assistance vocale, tels que les contrôles de confidentialités et de sécurité qui devront être améliorés.

### 2.2.2 Critères pour choisir un assistant vocal

Il s'agit des critères qu'il faut prendre en compte avant d'utiliser ou d'installer une application d'assistance vocale.

- **a) Intelligence :** un bon assistant doit se doter d'intelligence pour qu'il puisse donner les meilleures réponses appropriées à la demande des utilisateurs.
- **b) Support de la langue désirée (ici l'arabe) :** la langue que nous utilisons souvent est un facteur très important qui doit être présent dans l'assistant choisi.
- **c) Compatibilité et adaptabilité :** les assistants vocaux ont été fabriqués de manière à s'adapter à l'environnement dans lequel ils se trouvent. Ainsi il est impératif de s'assurer que l'assistant soit compatible avec nos autres appareils connectés.
- **d) Fiabilité :** la fiabilité est un critère qui doit être présent dans n'importe quel logiciel.
- **e) Pertinence des résultats :** un bon assistant vocal doit retourner des résultats corrects et adéquats aux requêtes des utilisateurs dans la mesure du possible.
- **f) Flexibilité :** l'assistant vocal doit être capable de répondre aux différents besoins en constante évolution.
- **g) Facilité d'utilisation (interface) :** l'utilisateur doit pouvoir se servir aisément des assistants intelligents, même s'il n'est pas un professionnel de la high-tech.
- **h) Temps d'exécution et de réponse :** le temps d'exécution et de réponse d'un assistant vocal compte beaucoup pour continuer à l'utiliser. En effet, plus le temps de réponse est court, plus l'utilisation de cet assistant accroît.
- **i) Sécurité et protection de données :** l'assistant doit être apte à protéger les données et les informations de ses utilisateurs contre les accès ou manipulations non autorisées.
- **j) Qualité du son et capacité de reconnaissance vocale :** la qualité sonore d'un assistant vocal et sa capacité de reconnaître la voix devront être irréprochables. Ce sont des éléments essentiels pour des assistants boostés à l'Intelligente Artificielle. L'idéal est d'opter pour un assistant vocal qui reconnaît différents timbres de voix.
- **k) Absence de bugs :** Les bugs est un paramètre négatif dans l'utilisation d'un assistant vocal.
- **l) Non cherté / gratuité :** le budget est un paramètre non négligeable dans la sélection d'un assistant vocal. Si l'assistant n'est pas très cher ou gratuit, il attirera plus d'utilisateurs.

## 2.2.3 La solution de l'assistant vocal proposé « FASSIHA »

### 2.2.3.1 Comparatif des assistants vocaux existants et FASSIHA

Les caractéristiques, les avantages et les inconvénients des produits d'Assistance vocale disponibles sur le marché, nos besoins, ainsi que les critères de choix d'un assistant vocal, nous ont permis de dresser le tableau ci-après, qui compare l'assistant virtuel proposé (فاصيحة/**Fassiha**) avec quelques assistants virtuels existants sur le marché, où :

	Intelligence	Support de l'arabe	Adaptabilité	Fiabilité	Pertinence des résultats	Flexibilité	Facilité d'utilisation	Réduction de temps de réponse	Sécurité et protection des données	Qualité du son	Absence des bugs	Gratuité
	Google Assistant	Alexa	Siri	Cortana	RafiQ	Fassiha						
Google Assistant	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Alexa	✓	✗	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Siri	✓	✓	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓
Cortana	✓	✗	✗	✓	✓	✓	✓	✗	✓	✓	✓	✓
RafiQ	✗	✓	✗	✓	✗	✓	✓	✗	✗	✗	✓	✓
Fassiha	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

**Tableau 2.1** Comparatif des assistants vocaux existants et Fassiha

- **Les lignes i** : représentent les principaux assistants virtuels disponibles sur le marché avec Fassiha à la dernière ligne.
- **Les colonnes j** : représentent les principaux critères qui doivent être présents dans un bon assistant vocal (cités plus haut).
- **Les cellules [i, j] vertes** : critère j présent dans l'assistant vocal i.
- **Les cellules [i, j] rouges** : critère j absent dans l'assistant vocal i.

### 2.2.3.2 Description de l'assistant vocal FASSIHA, *le meilleur pour tous*

Les assistants vocaux exigent la mobilité, pour cela nous avons proposé une application mobile Android intelligente "Fassihah", capable de reconnaître parfaitement la voix humaine, de traiter automatiquement le langage naturel et de synthétiser la voix.

L'application est facile d'utilisation, personnalisable, ayant un bon support la langue arabe et donne des résultats pertinents seulement en quelques secondes, tout en protégeant les données et informations des utilisateurs. C'est un logiciel sophistiqué, sécurisé, fiable, flexible, gratuit, compatible et fonctionne parfaitement sous la plateforme Android.

Fassihah est une bonne apprenante de la langue arabe, elle s'adapte au fur et à mesure aux recherches, habitudes et préférences de l'utilisateur. Elle est très précise en termes de réponses et de résultats et effectue des dialogues, permettant une interaction avec l'utilisateur. En effet, elle est capable de reconnaître facilement la voix. Il est également possible de communiquer avec notre assistant par écrit.



**Figure 2.6** Logo de Fassihah

## 2.3 Conclusion

Ce chapitre nous a permis, d'une part, d'assimiler les différentes façons dont les assistants ont été conçus et d'autre part, de présenter des assistants occupants le marché actuel. Ceci nous a permis d'examiner et de mentionner les points forts et faibles de chaque assistant, afin de pouvoir proposer un système d'assistance vocale adéquat aux besoins d'une certaine catégorie d'utilisateurs, qui utilisent la langue arabe. Dans le chapitre qui suit, nous allons voir la conception de notre assistant vocal Fassihah.

Chapitre  
**3**

# CONCEPTION

Comme tout projet informatique, notre projet d'assistance vocale arabe nécessite une phase de conception qui est un procédé permettant de formaliser les étapes préliminaires du développement d'un système. Dans ce chapitre, nous nous intéressons à la conception, où nous apportons tous les détails à notre solution, tout en cherchant à décrire le système en associant la forme et l'architecture correspondante

## 3.1 Architecture globale de l'assistant vocal « FASSIHA »

Après avoir analysé les différents articles et examiné les derniers assistants vocaux qui occupent le marché, nous avons pu concevoir une architecture pour notre assistant vocal virtuel supportant la langue arabe « Fassiha ». Bien que notre système partage des modules et composants nécessaires avec les autres systèmes, nous avons ajouté notre contribution personnelle dans chaque étape de traitement.

L'architecture globale détaillée de notre système Fassiha est représentée dans la figure qui vient ci-après (Figure 3.1). Ce dernier est divisé en deux sous-systèmes :

**1) L'interface utilisateur (Front-end) :** c'est l'application mobile, à travers laquelle l'utilisateur peut interagir et communiquer avec le système (réception des requêtes vocales des utilisateurs et lecture/affichage des réponses et des résultats aux utilisateurs).

Nous avons choisi d'intégrer notre système dans une application mobile pour le besoin de mobilité. En effet, les assistants vocaux exigent la portabilité et la mobilité vu que les utilisateurs utilisent leurs smartphones presque partout et tout le temps, donc ils auront sûrement besoin d'utiliser leurs assistants vocaux pour lancer des requêtes ou des conversations à n'importe quel moment et endroit.

**2) Le système intelligent d'arrière-plan (Back-end) :** représente le « cerveau » du système, car c'est là où l'intelligence intervient et les différents traitements s'exécutent sur la requête vocale de l'utilisateur pour générer une réponse adéquate à cette requête.

Notre système est construit à base de modules (architecture modulaire). **L'interface utilisateur** comprend 2 modules (M1, M7) et **le système intelligent d'arrière-plan** comprend 5 modules (M2, M3, M4, M5, M6).

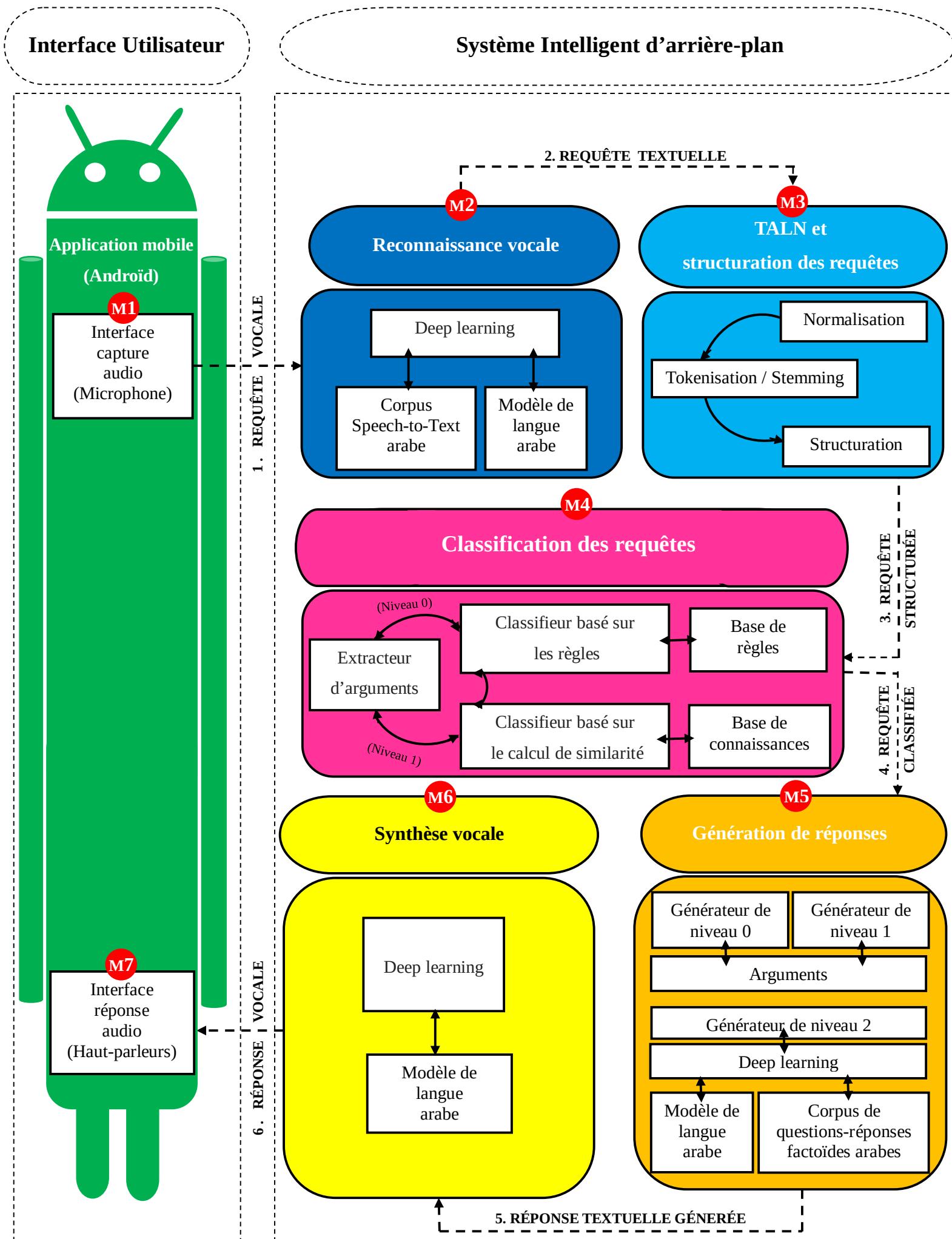


Figure 3.1 Architecture globale du système « FASSIHA »

Nous avons opté pour l'architecture par module pour leur facilité de modification et de mise à jour vu que chaque module est indépendant des autres modules. Ainsi, nous pouvons modifier entièrement l'architecture interne d'un module sans se soucier du problème de compatibilité, il suffit juste de garder le même format et structure de ses entrées et sorties.

Le système se déclenche lorsque l'utilisateur lance une commande vocale, cette commande passe dans l'ordre dans les modules (M1, M2, M3, M4, M5, M6, M7). L'entrée du module  $i+1$  est la sortie du module  $i$ , avec  $1 \leq i < 7$ . La communication entre ces modules assure la coopération, la coordination et la compatibilité. Dans ce qui suit, nous allons expliquer comment les différents modules des deux sous-systèmes cités plus haut procèdent, pour que la machine (smartphone) puisse générer une réponse adéquate à la commande vocale prononcée par l'utilisateur. Nous verrons également l'architecture détaillée de chaque sous-système.

## 3.2 L'interface utilisateur (Front-end)

Ce sous-système est le point d'interaction de l'utilisateur avec le système (l'application mobile), il agit comme une interface entre l'utilisateur et le reste des modules du système, i.e. l'interface utilisateur est un niveau d'abstraction entre le Back-end et l'utilisateur, qui ne doit dans aucun cas être exposé au fonctionnement interne du système.

L'interface utilisateur en fait, ce n'est qu'une interface graphique englobant 2 modules d'interaction utilisateur-système (interface de capture audio et interface de réponse audio).

### 3.2.1 L'interface graphique

Nous avons discuté la raison pour laquelle nous avons opté pour une application mobile. Dans une telle application, l'élément principal est l'interface graphique, car elle désigne la manière dont un logiciel est présenté à l'écran pour l'utilisateur ainsi que le positionnement des éléments (menus, boutons, etc.). Nous nous sommes basés sur la simplicité, l'ergonomie et l'intuition pour concevoir l'interface graphique de FASSIHA, afin que l'utilisateur puisse comprendre tout de suite comment exploiter l'application.

### 3.2.2 Interface de capture audio (M1)

L'interface de capture audio est un module de l'interface utilisateur et c'est le premier module du système qui se déclenche pour recevoir la requête, cette dernière est passée oralement par l'utilisateur (commande vocale). L'interface de capture audio doit donc attentivement capturer

le son pour assurer sa précision et passer la bonne et correcte commande (signal audio) au module suivant (M2) pour le traitement. Cette opération est assurée à l'aide des entrées/capteurs audio (microphones) préinstallés sur les Smartphones. Android permet aux développeurs d'utiliser ces capteurs pour récupérer les commandes vocales des utilisateurs. Ce processus est implémenté comme un Listener qui se lance à la demande et continue à entendre la commande vocale de l'utilisateur. Android permet aussi la configuration de ce Listener et donne le choix au développeur de personnaliser l'événement (comme par exemple le choix du temps d'attente pour que l'utilisateur entre sa commande), ce qui permet de donner plus d'ergonomie à l'application et de bien la configurer selon le besoin de chaque utilisateur.

### 3.2.3 Interface de réponse audio (M7)

L'interface de réponse audio est le deuxième module de l'interface utilisateur et c'est le dernier module du système qui se réplique, pour communiquer la réponse du système à l'utilisateur. Après le traitement de la requête dans la partie interne (Back-end) du système, une réponse est générée pour cette requête, puis est communiquée à l'utilisateur. Pour simuler un assistant réel, la meilleure façon de délivrer la réponse à l'utilisateur, c'est à travers le son (signal audio). Pour ce faire, nous avons utilisé les sorties audio préinstallées avec le smartphone. Ces sorties peuvent être des haut-parleurs, des écouteurs avec ou sans fil ou n'importe quelle autre sortie disponible fournie par Android.

Pour des raisons d'accessibilité, nous avons aussi fourni une méthode d'entrée textuelle (commande écrite) et une méthode de sortie textuelle (un affichage de résultats en format texte).

### 3.2.4 Architecture et conception de l'interface utilisateur (Front-end)

#### Pourquoi le système d'exploitation Android ?

Lors du développement d'une application mobile, deux systèmes d'exploitation mobiles majeurs s'imposent : Android et IOS, bien qu'il soit toujours préférable d'implémenter l'application sur les deux systèmes, nous avons été limités par le temps et les ressources. De ce fait, nous étions obligés de choisir un de ces deux systèmes d'exploitation mobile, pour qu'il serve comme plateforme à notre application. Nous avons opté pour Android pour les raisons suivantes:

- **Système open-source** : Android est un système d'exploitation open-source fondé sur le fameux noyau Linux<sup>3</sup>, bien que ce système soit maintenu par Google, il offre une grande

---

<sup>3</sup> [en.wikipedia.org/wiki/Android\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Android_(operating_system))

liberté aux développeurs pour l'implémentation de leurs applications, la personnalisation et l'utilisation de toute fonctionnalité offerte par le smartphone et assurée par Android, en gardant la sécurité et la fiabilité de Linux.

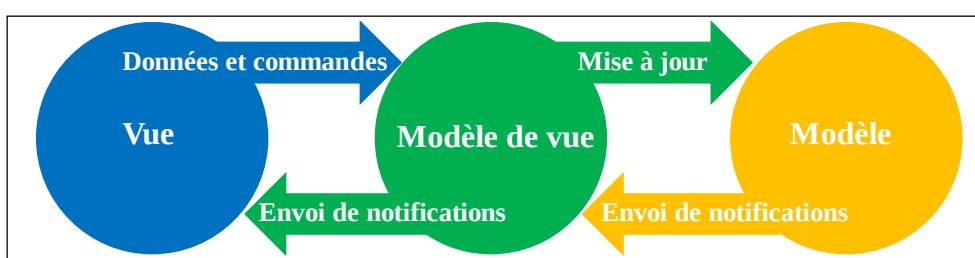
- **Un seul système, plusieurs appareils** : contrairement à IOS qui est supporté uniquement par iPhone. Android est adopté par la majorité des constructeurs des smartphones, ce qui offre une grande et variante base d'utilisateurs, et un ensemble varié de smartphones pour tester l'application.
- **Support de Google** : Android est supporté par Google, ce qui offre gratuitement des outils très puissants pour le développement (comme Android Studio et Gradle).

### Pourquoi l'architecture logicielle MVVM ?

Pour concevoir une application Android, plusieurs architectures peuvent être implémentées, pour notre application nous avons choisi le Pattern MVVM [50] (Model-View-ViewModel) ou (Modèle-Vue-VueModèle). C'est une architecture logicielle qui se concentre principalement sur la séparation de l'interface utilisateur graphique ou front-end (la vue) du traitement interne du système ou back-end (le modèle) par le biais d'un modèle de vue, de sorte que la vue ne dépende d'aucune plateforme de modèle spécifique. Cette architecture s'adapte parfaitement à notre besoin, puisque notre traitement de l'information est totalement séparé de l'interface utilisateur. MVVM est aussi une architecture standard pour les applications Android, ce qui facilite le développement et la mise à jour de notre application.

En résumé, le pattern MVVM se compose de trois modules principaux:

- 1) La vue (View)** : chargée de l'affichage des différents éléments graphiques (tout ce que nous voyons).
- 2) Le modèle (Model)** : représente l'implémentation et le traitement nécessaire en Back-end du logiciel (tout ce que nous ne voyons pas).
- 3) Le modèle de vue (ViewModel)** : chargé de la conversion des objets (données) entre la vue et le modèle d'une manière à ce que ces objets soient facilement gérés et présentés. Ainsi, ce module assure la compatibilité de la communication entre les deux modules précédents.



**Figure 3.2** Architecture MVVM [50]

### 3.3 Le système intelligent d'arrière-plan (Back-end)

Le back-end est la partie essentielle du système car c'est là où l'intelligence intervient. Ce sous-système regroupe cinq modules (la reconnaissance vocale, le TALN et la structuration des requêtes, la classification des requêtes, la génération des réponses et la synthèse vocale). Ces modules sont chargés du traitement de la requête de l'utilisateur et la génération de la bonne et adéquate réponse par la suite. Cette partie est une boîte noire pour l'utilisateur, i.e. il ne peut pas voir le processus de traitement de sa requête; ce qui facilite l'utilisation de l'application et familiarise l'utilisateur avec cette dernière. Dans ce qui suit, nous allons expliquer le fonctionnement du Back-end, notre conception et l'architecture de chacun de ces 5 modules.

#### 3.3.1 Module de reconnaissance vocale (M2)

Comme mentionné plus haut, l'assistant vocal est conçu pour recevoir principalement des requêtes et des commandes vocales. De ce fait, pour déclencher le processus de traitement, l'utilisateur n'a qu'à lancer une nouvelle commande vocale en parlant à son appareil. Pour traiter cette commande vocale, il faut d'abord la convertir en un texte compréhensible par notre machine. C'est pour cette raison, que nous avons mis en place un module de la reconnaissance vocale (Speech-To-Text), comme premier module du back-end. Ce module consiste à convertir l'entrée vocale en un texte avec un minimum taux d'erreur, car n'importe quelle erreur dans la conversion de la requête de l'utilisateur va se propager dans les étapes suivantes du traitement, il faut donc développer ce module avec précaution et essayer de maximiser l'efficacité de ce dernier dans la mesure du possible.

Pour le module de reconnaissance vocale, nous avons choisi d'intégrer l'Application Programming Interface (API) de Google SpeechRecognizer et de l'adapter à la langue arabe. C'est le système de reconnaissance vocale automatique multilingue de Google. Ce choix est fait après avoir essayé de développer notre propre modèle, une chose qui demande beaucoup de temps mais surtout de moyens de calcul et de corpus Speech-To-Text, mais de tels corpus sont rares pour la langue arabe. Le meilleur corpus que nous avons pu trouver est le corpus de Mozilla sous le projet Open Voice<sup>4</sup>, ce corpus contient 12 heures d'enregistrements sonores pour l'arabe. Or, le minimum pour atteindre une bonne précision de reconnaissance vocale est 100 heures. Vu les difficultés du développement d'un système de reconnaissance vocale, nous avons opté pour l'API SpeechRecognizer. Cette dernière est développée par Google, donc elle est parfaitement intégrable avec les applications Android, ce qui atteste la fiabilité de notre système.

---

<sup>4</sup> <https://commonvoice.mozilla.org/en/datasets>

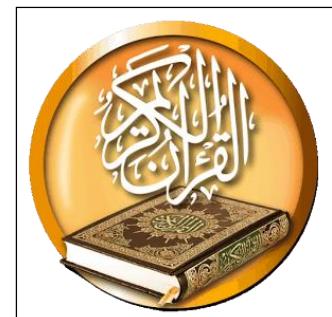
**Architecture de l'API SpeechRecognizer de Google :** Google fournit une API de reconnaissance vocale avec les APIs de développement Android. Cette API est une interface pour l'utilisation du système de reconnaissance vocale Google Cloud, qui est un système fourni par Google pour permettre la reconnaissance vocale et la diffusion de l'audio sur le Cloud, ce qui minimise l'utilisation des ressources de l'appareil de l'utilisateur et donne plus de précision. La consommation de ce service à partir d'Android ne nécessite que l'utilisation de l'interface SpeechRocgnizer pour envoyer une requête vocale et recevoir la transcription textuelle.

Google Cloud a une architecture séquence à séquence [23], plus précisément, c'est un modèle basé sur l'intention nommé LAS (Listen Attend and Spell). Cette architecture permet la construction d'un grand modèle d'apprentissage sans avoir besoin de séparer les étapes de traitement de reconnaissance vocale, en assurant la meilleure précision des systèmes traditionnels.

**Modèle de langue :** le modèle a à sa disposition l'un des meilleurs modèles de langue de Google [51], qui permet la construction de l'HMM intégré avec le modèle de reconnaissance vocale pour améliorer la précision. Il faut noter aussi que la langue arabe est supportée nativement par Google.

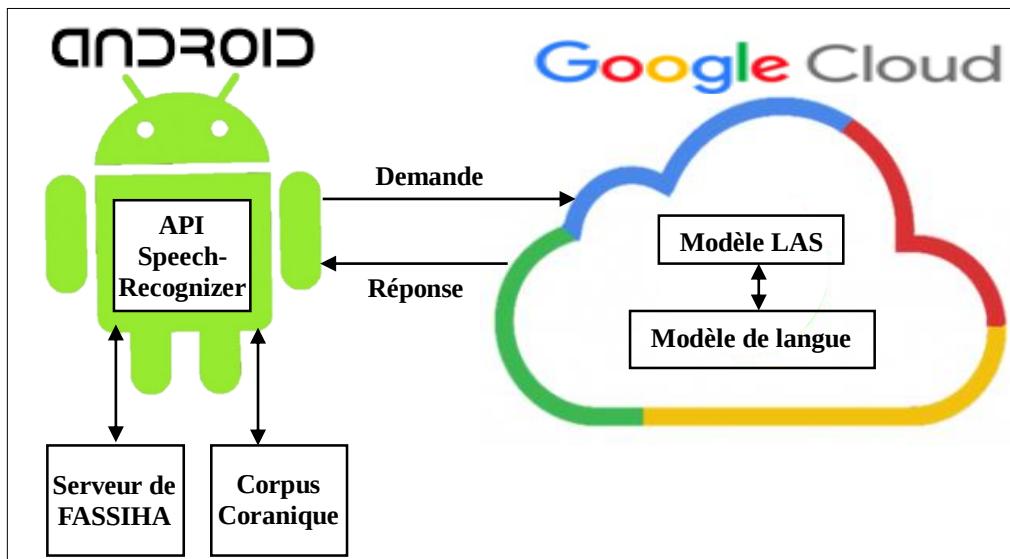
**Corpus d'apprentissage :** Google Cloud permet à l'utilisateur d'intégrer son propre corpus d'apprentissage pour améliorer le rendu de la reconnaissance vocale. Pour cette raison, nous avons construit notre propre corpus, il s'agit du corpus coranique (texte sacré), composé de 114 chapitres nommés (*sourates*), chaque *soura* est composée de versets nommés (*ayat*). Les versets sont au nombre de 6236. La disponibilité de plusieurs versions audio et textuelles facilite la construction d'un tel corpus. Par ailleurs, le Coran est connu pour la richesse de son vocabulaire arabe, la chose qui a permis à notre système de bien améliorer la reconnaissance vocale.

Le modèle communique avec le corpus coranique et continue l'apprentissage et l'évaluation sur ce corpus en parallèle avec le fonctionnement habituel de l'API.



**Figure 3.3** Image du livre sacré (corpus coranique)

La figure qui vient ci-après, montre l'architecture du module de reconnaissance vocale :



**Figure 3.4** Architecture du module de reconnaissance vocale

**Sortie :** Après la phase de reconnaissance vocale, la commande vocale de l'utilisateur est transformée en format texte. Le module alors envoie le fichier texte contenant cette commande au module suivant qui est le traitement automatique du langage naturel est la structuration de la requête pour enchaîner l'étape suivante.

### 3.3.2 Module de TALN et structuration des requêtes (M3)

Ce module est la base du processus, car c'est à ce niveau-là que les commandes textuelles reçues de M2 sont traitées. Ces commandes ont le même format texte (grâce à M2) certes, mais n'ont pas forcément la même structure, car le langage naturel utilise un vocabulaire non exhaustivement défini, avec une sémantique ambiguë et des règles syntaxiques nombreuses à la complexité variable. Ainsi, chaque utilisateur constitue et synthétise sa requête de la façon qu'il souhaite. En effet, nous pouvons demander à notre smartphone d'ouvrir une même application, d'une manière implicite ou explicite, avec mille et une requêtes, et ce :

- en utilisant des mots synonymes (الجوّ / الطقس) etc.) ;
- en employant des mots avec différents proclitiques et affixes (الجوّ / الجوي / الجوية) etc.) ;
- en conjuguant les verbes aux différents temps (l'accompli, l'inaccompli, l'impératif, etc.) ;
- en formulant la requête sous forme d'un ordre ou d'une demande polie (أمرك بفتح / افتح) ( أمرك بفتح / افتح من فضلك / هل بإمكانك فتح؟ // افتح من فضلك / هل بإمكانك فتح؟ ) etc.) ;

- en employant des mots qui ont relation avec l'application ou le service (des mots du champ lexical). Il s'agit d'un ensemble de noms, d'adjectifs et de verbes liés au domaine de l'application par leur sémantique (la manière implicite).

Pour structurer ces requêtes, le module M3 commence d'abord par normaliser la requête, puis la traite, pour qu'il puisse enfin la structurer et cela dans le but de représenter la requête dans un format homogène, cohérent et compréhensible dans la mesure du possible par la machine et par le développeur. Ainsi, ce dernier peut facilement modifier et mettre à jour le module, dans le but de faire collaborer ce module avec les modules suivants.

**1) La normalisation :** c'est la première étape du module M3, où la requête textuelle va être normalisée, i.e. éliminer/ignorer tous ce qui est inutile et non essentiel et ne garder pour le traitement que l'essentiel qui donne une signification à la requête. C'est une sorte de nettoyage de la requête.

La normalisation de nos requêtes comprend 2 étapes.

- **La négligence des mots vides (*stop words*) :** pour le traitement des commandes basiques, nous avons négligé tous les mots vides qui ont tendance à être tellement communs qu'il est inutile de les prendre en considération que ce soit dans une recherche ou dans un traitement. Parmi les mots vides évidents nous citons (و / من /إلى /على /لكن /أو /إذن /لأن ) etc.). Par ailleurs, dans le traitement des commandes avancées, ces mots vides nous ont été utiles pour faire un bon apprentissage de l'arabe, la raison pour laquelle nous les avons négligés et non pas les supprimés.
- **La suppression des signes de ponctuation :** que ce soit dans les commandes basiques ou avancées, le traitement de la ponctuation ne nous apporte rien. Les signes de ponctuation sont plutôt importants lors de la lecture, ou dans des projets spécifiques qui traitent par exemple des phrases interrogatives (?), exclamatrices (!), etc.

**2) Le TALN :** c'est une étape importante car elle nous a permis de faire la liaison entre la linguistique, l'informatique et l'intelligence artificielle. Une fois la requête normalisée, nous passons à son traitement d'une façon automatique. Pour ce faire, nous avons appliqué à chaque requête : une analyse morphologique qui est la tokenisation et une analyse morphologique qui est le stemming, dans le but de parvenir automatiquement à la compréhension la plus complète de la requête de l'utilisateur et d'exploiter les ressources linguistiques massives et complexes pour lever les ambiguïtés des requêtes textuelles.

- **La tokenisation** : nous avons segmenté chaque requête en unités linguistiques/lexicales (tokens). Nous avons fait segmentation de bas niveau où nous nous sommes intéressés aux mots (il est également possible de tokeniser en clauses, phrases et paragraphes). Cette tokenisation est basée sur l'espace; le marqueur de séparation privilégié à l'écrit, mais nous sommes restés vigilants dans **certaines exceptions**, comme les noms détachés dont le lien est implicite (فوس قرح, etc.).

Par exemple dans la phrase افتح تطبيق الطقس, la tokenisation corresponds à :

فتح / تطبيق / الطقس

- **Le stemming (racinisation)** : une fois les unités lexicales repérées, il est possible de normaliser la forme des tokens en passant par une étape de racinisation. Nous avons donc fait une racinisation pour les mots de la requête, en attribuant à chaque mot son stem. La racinisation consiste à supprimer tous les affixes (préfixes, infixes et suffixes) du mot, en ne gardant que ses lettres radicales. Cette étape facilite la reconnaissance des mots d'une requête donnée et le rapprochement des mots similaires, i.e. le stemming ramène les 3 mots rapprochés suivants : طقس/طقس/طقسي : à leur stem, qui est : طقس.

Par exemple, si nous reprenons notre précédente phrase: افتح تطبيق الطقس, le stemming correspond à: فتح طبق طقس

Nous stockons le stemming pour chaque requête, mais l'utilisation de ce stemming n'est évoquée que dans la gestion des cas de conflits (nous expliquerons ce point dans le module de classification des requêtes de niveau 1).

Ces étapes entament le processus de compréhension des requêtes de l'utilisateur, formulées en langage naturel. Elles sont le point de départ d'une analyse et d'un traitement long et complexe, permettant à une Intelligence Artificielle de comprendre la requête et d'y apporter une réponse adaptée.

**3) La structuration** : la structuration des données textuelles de la requête est importante afin que ces dernières soient uniformes et lisibles aussi bien par des humains que par des machines. Pour ce faire, nous avons opté pour le langage de balisage : eXtensible Markup Language (XML) pour les raisons suivantes.

- Il permet de décrire tout type de données grâce à sa structure arborescente et extensible.
- Il est facile à parser (analyser syntaxiquement) par les logiciels étant lisible entre les humains.
- Il est facile d'y faire des recherches.
- Il est utilisable dans tout contexte et sur tout système grâce à son universalité.

- Il permet de stocker une grande quantité de données dans un petit espace mémoire.

Par exemple, pour la requête افتح تطبيق الطقس , sa structuration équivaut au document XML suivant :

```
<?xml version="1.0" ?>
<root>
  <tokenization>
    <word id="1" value="فتح" stop_word="False" stem="فتح" />
    <word id="2" value="تطبيق" stop_word="False" stem="طبق" />
    <word id="3" value="الطقس" stop_word="False" stem="طقس" />
  </tokenization>
</root>
```

**Figure 3.5** Document XML associé à la requête افتح تطبيق الطقس

XML est un format fondé sur des balises comme HTML, Ces balises peuvent avoir aussi des attributs et cela fait penser aux langages de programmation orientés objets.

Nous avons un document XML pour chaque requête, chaque document contient :

- une déclaration XML contenant la version du langage XML utilisé ;
- un nœud d’élément racine `<root>`, englobant toutes les autres nœuds ;
- un nœud d’élément `<tokenization>`, contenant tous les mots de la requête de l’utilisateur ;
- des nœuds d’élément `<word>` (mot), qui sont au nombre de mots de la requête. Chaque nœud d’élément `<word>` contient 4 attributs : i) un attribut représentant l’identifiant de chaque mot de la requête (position), ii) un attribut représentant le mot de la requête, iii) un attribut représentant un booléen pour savoir si le mot est un mot vide (stop word) et enfin, iv) un dernier attribut représentant le stem équivalent à chaque mot de la requête.

**Sortie** : ce module fournit donc une structure commune pour toutes les requêtes, afin de les préparer à la prochaine étape qui va être implémentée spécifiquement sur la structure résultante de ce module. Ce module communique donc le fichier structuré de la requête au système global qui assure la transmission de ce fichier au module suivant, celui de la classification des requêtes.

### 3.3.3 Module de classification des requêtes (M4)

Comme nous avons montré dans l'état de l'art, il existe plusieurs approches et méthodes de conception des assistants virtuels. Chaque approche présente un ensemble d'avantages et un ensemble d'inconvénients. Le choix d'adopter une méthode nous a été difficile, ce qui nous a poussés à proposer notre propre approche en essayant de créer un système hybride qui prend les avantages des approches majeures existantes et en essayant de réduire les inconvénients dans la mesure du possible. En fait la spécificité majeure du système FASSIHA c'est cette hybridation, qui apparaît clairement dans le classifieur de requêtes.

Durant notre analyse des utilisations des assistants vocaux, nous étions exposés à plusieurs manières d'interactions user-assistant. Ces interactions varient d'une simple requête dont le résultat n'est qu'une exécution d'un simple processus (exemple : (افتح تطبيق الطقس :), هل يمكنني ارتداء معطف غداً؟), afin de pouvoir par la suite générer une réponse adéquate à la demande (ici, dans les exemples, nous demandons l'application météo).

Le traitement des différents niveaux de requêtes avec une seule approche est évidemment une mauvaise idée, c'est pour cela que nous avons eu l'idée d'implémenter ce module (La classification des requêtes). Ce module prend comme entrée le fichier XML du module précédent contenant la requête structurée, dans le but de séparer les différents niveaux de commandes. Cette séparation nous permet ensuite de générer la réponse correspondante selon les niveaux. Le processus de génération des réponses pour les commandes simples va être différent des commandes avancées, ce qui permet non seulement de donner de meilleurs résultats, mais aussi de minimiser le traitement et le temps de réponse pour les simples commandes (pas besoin de passer par un traitement complexe pour une commande simple).

En effet, nous avons construit 3 classificateurs distincts, dont chacun est chargé d'un niveau de requêtes (Niveau 0, Niveau 1 et Niveau 2).

#### 3.3.3.1 Classifieur de niveau 0

Ce classifieur regroupe les commandes les plus simples (commandes impératives) où l'intention de l'utilisateur est **explicite** dans sa requête (exemple : (افتح تطبيق الطقس :)). Ces commandes font déclencher un simple processus d'exécution d'une application ou l'accomplissement d'une simple tâche. Les commandes de ce niveau partagent la même structure et syntaxe (phrase verbale à l'impératif), ce qui nous a aidé à construire un analyseur à base de règles (grammaire) pour identifier de telles commandes. L'analyseur prend comme

entrée la requête structurée et analyse sa validité, en exécutant une analyse Bottom-up (du bas vers le haut / ascendante), utilisant des règles prédéfinies, 7 non terminaux et un nombre considérable de terminaux.



**Figure 3.6** Extrait de la base de règles associée aux requêtes de niveau 0

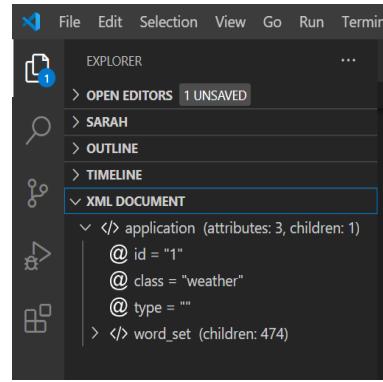
Si la requête respecte les règles, elle est classifiée donc comme étant une requête de niveau 0, le classifieur envoie donc la classe de la requête (0) et l'application correspondante directement au prochain module, sinon il la passe au prochain niveau de classification.

### 3.3.3.2 Classifieur de niveau 1

Contrairement au niveau précédent, ce niveau est plus complexe, car il inclut toutes les autres requêtes de l'utilisateur, que son intention soit **explicite** (exemple هل حالة الطقس ستكون معتدلة ؟) ou **implicite** (هل باستطاعتي لبس معطفى ؟). Ces dernières sont difficiles à comprendre par la machine vu la complexité de formalisation des requêtes. Donc, elles nécessitent un traitement avancé pour extraire la vraie intention de l'utilisateur. Pour cette raison, nous avons implémenté ce niveau.

Pour identifier l'intention de l'utilisateur (l'application souhaitée) et pouvoir classifier ce genre de requêtes, nous avons mis en place un système basé sur le calcul de similarité entre la requête de l'utilisateur et une base de connaissances (BC) que nous avons construite. Celle-ci inclut des fichiers XML structurés (Datasets). Chaque fichier est associé à une application précise et englobe un bon nombre de termes, que leur relation soit forte ou faible avec le domaine de l'application.

Par exemple, pour l'application **Météo**, nous avons représenté un maximum de termes ayant une relation de près ou de loin avec la météo, ces termes sont au nombre de 474.



**Figure 3.7** Caractéristiques du dataset associé à l’application Météo

Les termes à leur tour, sont divisés en 2 catégories : termes composés et termes non composés (2<sup>e</sup> attribut), pour assurer une meilleure génération de réponse par la suite.

```

<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<! --Météo -->
<application id="1" class="weather" type="">
  <word_set>
    <!-- Non Composés -->
    <word id="1" composed="false" value="طقس" />
    <word id="2" composed="false" value="جو" />
    <word id="3" composed="false" value="فتن" />
    <word id="4" composed="false" value="بطر" />
    <word id="5" composed="false" value="نَجَّ" />
    <word id="6" composed="false" value="رِيح" />
    <word id="7" composed="false" value="زَرْد" />
    <word id="8" composed="false" value="رطوبة" />
    <word id="9" composed="false" value="غَمَّ" />
    <word id="10" composed="false" value="رَمَاد" />
    <word id="11" composed="false" value="سَحَابَة" />
    <word id="12" composed="false" value="عَذَقَ" />
    <word id="13" composed="false" value="زَوْبَانَة" />
    <word id="14" composed="false" value="عَصَارٍ" />
    <word id="15" composed="false" value="فَلَاجَةً" />
    <word id="16" composed="false" value="سَبَبَ" />
    <word id="17" composed="false" value="مَيْدَان" />
    <word id="18" composed="false" value="أَرْضَة" />
    <word id="19" composed="false" value="رَعَى" />
    <word id="20" composed="false" value="فَهْفَانَة" />
    <word id="21" composed="false" value="لَوْنَانَ" />
    <word id="22" composed="false" value="جَعْلَفَانَ" />
    <word id="23" composed="false" value="شَفَانَ" />
    <word id="24" composed="false" value="فَنَانَ" />

```

**Figure 3.8** Termes non composés du dataset associé à l’application Météo

```

<!-- Composés -->
<word id="1" composed="true" value="درجة حرارة" />
<word id="2" composed="true" value="حركة مدار" />
<word id="3" composed="true" value="ملاءة و مدد بمحرك" />
<word id="4" composed="true" value="غاف جوي" />
<word id="5" composed="true" value="علوم ارض" />
<word id="6" composed="true" value="مسشار اسلانى" />
<word id="7" composed="true" value="ديناميكية حرارية" />
<word id="8" composed="true" value=" مجرد طبور" />
<word id="9" composed="true" value="تساقط اوراق شجر" />
<word id="10" composed="true" value="قياس تدفق" />
<word id="11" composed="true" value="بحر أبيض متواتر" />
<word id="12" composed="true" value="طبقه ازوفون" />
<word id="13" composed="true" value="بحر هادئ" />
<word id="14" composed="true" value="بحر مادق" />
<word id="15" composed="true" value="بحر مقطورة" />
<word id="16" composed="true" value="سماء رمادية" />
<word id="17" composed="true" value="سماء زرقاء" />
<word id="18" composed="true" value="إحتباس حراري" />
<word id="19" composed="true" value="موسم حصاد" />
<word id="20" composed="true" value="موسم زراعة" />
<word id="21" composed="true" value="موسم فلاح" />
<word id="22" composed="true" value="موسم حرب" />
<word id="23" composed="true" value="موسم جدب" />
<word id="24" composed="true" value="جندي زيتون" />
<word id="25" composed="true" value="គ្រាន់តិចិត្តី" />
<word id="26" composed="true" value="បូយុត បាស៊ិកីរី" />
<word id="27" composed="true" value="ស្ថរុណ មិន" />
<word id="28" composed="true" value="ខ្លួន ដុំ" />

```

**Figure 3.9** Termes composés du dataset associé à l’application Météo

Le calcul de similarité entre la requête et chaque dataset contenant les mots clés relatifs à une application (Météo, Alarme, Calculatrice, etc.), donne un score qui varie entre 0 et 1. Le système affecte donc à la requête, l’application correspondante ayant le score le plus élevé. Le calcul de similarité est un simple processus faisant l’intersection entre les mots de la requête et ceux de la BC. En effet, nous avons simplifié ce processus, puisque le nettoyage et la

structuration des requêtes annulent déjà la plupart des conflits qui peuvent apparaître dans le calcul de similarité.

**Cas de conflits** : une requête peut contenir plusieurs mots appartenant à plusieurs domaines. Le système arrive à bien classifier les requêtes (quand les scores de similarité associés à la requête sont différents), mais dès que ces scores sont égaux, le système ne va pas bien classifier la requête à cause de la multitude d'affectation, il affectera plutôt deux applications ou plus à la même requête selon le nombre de domaines sous-entendus dans cette dernière.

Par exemple, si nous prenons la phrase : هل ستحدث عاصفة عند المقابلة؟

- « عاصفة » est un mot clé du dataset relatif à l'application Météo.
- « مقابلة » est un mot clé du dataset relatif à l'application Alarme.

La requête sous-entend 2 domaines (météo et alarme) et les 2 scores de similarité sont de 0.33 (égalité), ce qui fait deux applications vont être affectées à la même requête (Météo et Alarme).

Selon le sens de la phrase c'est bien clair que c'est l'application Météo qui doit être affectée à cette requête, mais la machine n'est pas assez intelligente pour le constater.

**Résolution du cas de conflits** : pour faire recours à ce genre de conflits, notre solution était de calculer la similarité avec stemming de termes de la requête (exploitation de la racinisation stockée dans le module M3). Ceci nous a réglé le problème de multitude d'affectation dans la plupart des cas, par-ce-qu' après un stemming, il ne nous reste que les mots appartenant certainement à un domaine, donc seuls ces mots qui seront pris en considération.

```
Requête: هل ستحدث عاصفة عند المقابلة
class: 1
Application résultat sans stemming: alarm_clock + weather. Avec score:0.3333333333333333
Application résultat avec stemming: weather. Avec score:0.3333333333333333

Process finished with exit code 0
```

**Figure 3.10** Applications affectées à une requête de niveau 1 avant et après stemming

Notons ainsi, qu'une application n'est affectée à une requête que si le score de similarité est supérieur à 0.25, ce score est estimé après un processus d'expérimentations, et ce,

pour empêcher le système d'affecter une application à une commande dans le cas où cette commande est très faiblement liée à un domaine.

Si le classifieur de niveau 1 réussit à affecter une et une seule application à la requête, cette dernière sera classifiée comme étant une requête de niveau 1, il envoie donc l'information concernant la classe (1) et l'application correspondante directement au prochain module, sinon, si le problème de multitude est toujours présent, nous faisons appel au prochain niveau de classification.

### 3.3.3.3 Classifieur de niveau 2

Bien que dans la plupart des cas, l'utilisateur a l'intention d'exécuter des commandes et services précis, mais il existe certains cas où il voudrait interagir avec l'assistant, soit pour effectuer une conversation, soit pour pouvoir avoir des réponses à ses questions ou bien pour exploiter d'autres comportements offerts par l'IA et voir les limites de cette dernière. Ce niveau se déclenche automatiquement lorsque la requête n'est classifiée ni comme étant une requête du niveau 0, ni de niveau 1. En effet, il utilise des techniques avancées de l'IA pour générer des réponses.

**Sortie :** après ce module, nous avons la requête classifiée et l'intention de l'utilisateur bien claire. Nous obtenons ainsi 2 sorties : les informations relatives à la classe de la requête (la classe 0, 1 ou 2) et l'application correspondante à la requête (pour les classes 0 et 1), nous passons maintenant à l'étape suivante pour extraire les arguments de la requête et générer les réponses.

### 3.3.4 Module de génération de réponses (M5)

Ce module est très important car il représente le « cerveau » du back-end, c'est à ce niveau-là que les arguments de la requête sont extraits et les réponses sont générées, par conséquent, la qualité de notre système dépend fortement de la qualité et de l'efficacité de ce module.

Ce module prend comme entrée le fichier structuré ayant la classe de la requête (niveau 0, niveau 1 ou niveau 2) et l'application à lancer (pour les niveaux 0 et 1).

Chaque niveau est associé à son propre générateur de réponse, ce qui fait que nous avons 3 générateurs :

### 3.3.4.1 Générateur de réponses de niveau 0

Générateur à base de règles. Une fois que nous avons l'application désirée, nous extrayons l'action et les arguments à partir de la requête pour générer la réponse adéquate. Pour l'exemple précédent : افتح تطبيق الطقس في الجزائر :

- L'action correspond au premier mot de la requête (verbe à l'impératif), qui correspond à l'un des terminaux de la première règle : *verb* → اغلق | افتح | etc, ici افتح.
- Les arguments correspondent aux terminaux des autres règles, ici الجزائر .

Après cette extraction, nous générerons donc la réponse adéquate « فتح تطبيق الطقس في الجزائر », puis un processus d'exécution simple et spécifique sera suffisant pour répondre à la requête (pour l'exemple, nous affichons à l'utilisateur la météo à Alger).

### 3.3.4.2 Générateur de réponses de niveau 1

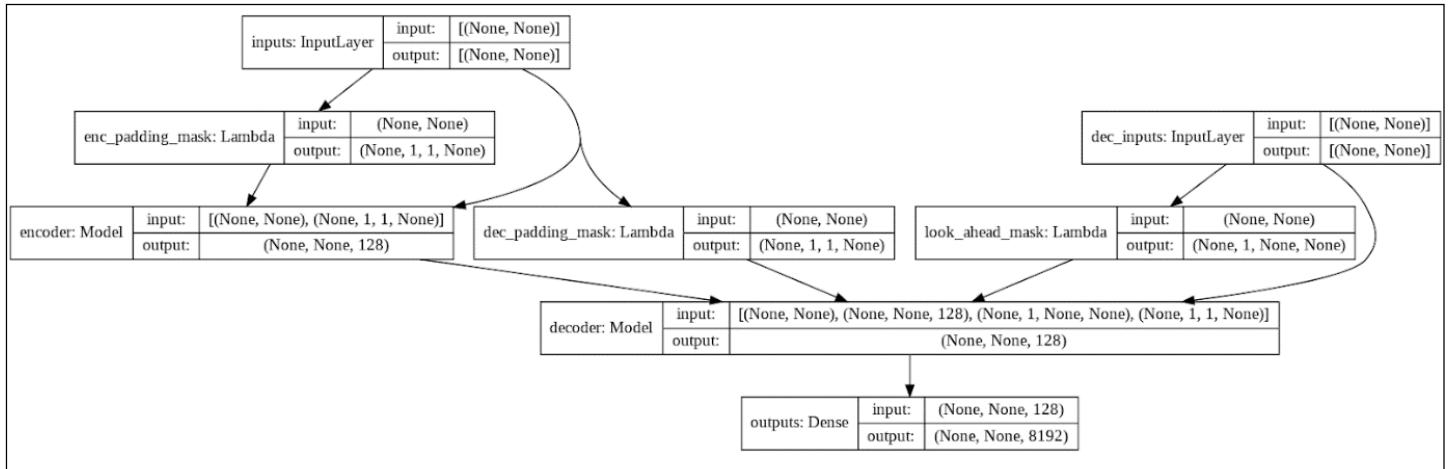
Ce générateur est plus complexe que le précédent car la façon dont les utilisateurs forment leurs requêtes est différente et n'ayant pas la même structure, donc nous avons essayé de rapprocher son traitement du traitement précédent. Nous extrayons les arguments des requêtes de niveau 1, en faisant un étiquetage morphosyntaxique que nous expliquerons dans l'implémentation.

Par exemple dans la requête : هل باستطاعتي ليس معطفي غداً ؟ , le seul argument est la conjonction de temps « غداً », pour savoir qu'il faut afficher la météo de demain.

### 3.3.4.3 Générateur de réponses de niveau 2

Dans certains cas, l'utilisateur souhaite avoir une conversation *Human-like*, donc il formalise des requêtes et commandes assez complexes. Il est presque impossible de les traiter avec les méthodes traditionnelles de génération de réponses, en même temps, il est défavorable de restreindre l'utilisateur et de l'obliger à suivre un langage formalisé pour passer ses commandes. Pour cela, nous avons décidé d'ajouté donc un autre niveau de génération des réponses, chargé du traitement des conversations *Human-like*. Ce niveau se base sur un modèle complexe de l'apprentissage profond (deep learning). En effet, nous avons réalisé un modèle séquence-à-séquence (Seq2Seq) basé sur l'**attention** [52] pour ce niveau. Dans ce qui suit, nous allons fournir tous les détails concernant ce modèle, mais tout d'abord, commençons par définir quelques termes importants à ce propos.

- **Un transformateur** : un transformateur est un modèle de deep learning conçu pour le traitement des données séquentielles comme le langage naturel, mais contrairement aux autres modèles traitant ce type de donnée (comme les RNN), les transformateurs n'exigent pas le traitement séquentiel de ces données, ce qui permet plus de parallélisme et réduit le temps d'apprentissage. Le transformateur utilisé dans notre modèle est représenté dans la figure qui vient ci-après [52].



**Figure 3.11** Transformateur de notre réseau deep learning [52]

L'autre avantage de cette architecture, c'est le fait que ce modèle peut apprendre des longues dépendances (les dépendances entre le début et la fin de la phrase, ou entre 2 phrases du même locuteur), ce qui permet de garder le contexte d'une conversation. Ainsi, ces dépendances peuvent être calculées sans avoir besoin de passer par des couches récurrentes, ce qui réduit le temps d'apprentissage.

- **Une attention** : comme son nom l'indique, l'attention est le fait que la machine donne plus d'importance (être attentive à) à des parties du texte ou du dialogue, ce comportement s'inspire des humains et est adopté par les chercheurs dans le TALN. Notre modèle utilise « *Scaled dot product attention* » pour repérer le plus important dans le dialogue, la formule de calcul est représentée ci-après, avec Q, représentant la requête, K, la clé de l'encodage du mot dans la requête et V, la valeur de ce mot.

$$\text{Attention}(Q, K, V) = \text{softmax}_k\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

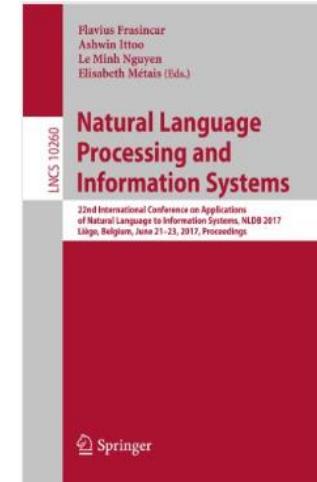
**Équation 3.1** Formule de calcul de l'attention [52]

Après avoir défini les principaux concepts utilisés dans ce modèle, nous allons expliquer le processus de traitement et d'apprentissage.

**Corpus d'apprentissage :** l'étape la plus difficile lors du traitement de la langue arabe, est de trouver de bonnes ressources (corpus et outils). En effet, le manque de ressources arabes, était la raison principale qui nous a motivés à réaliser ce projet.

Le corpus qu'on avait utilisé pour faire l'apprentissage et générer des réponses aux requêtes de niveau 2, est le corpus TALAA-AFAQ [53]. Ce dernier est un corpus de réponses aux questions factoïdes en arabe, pour un système de réponse aux questions.

Nous avons construit un corpus à partir de TALAA-AFAQ, et ce, en ne gardant que les questions et leurs réponses. En effet, nous avons éliminé tous les autres attributs de TALAA-AFAQ, puisque un modèle séquence-à-séquence n'a besoin que du texte brut de la conversation pour pouvoir apprendre.



**Figure 3.12**

Corpus de questions-réponses arabes factoïdes TALAA-AFAQ [53]

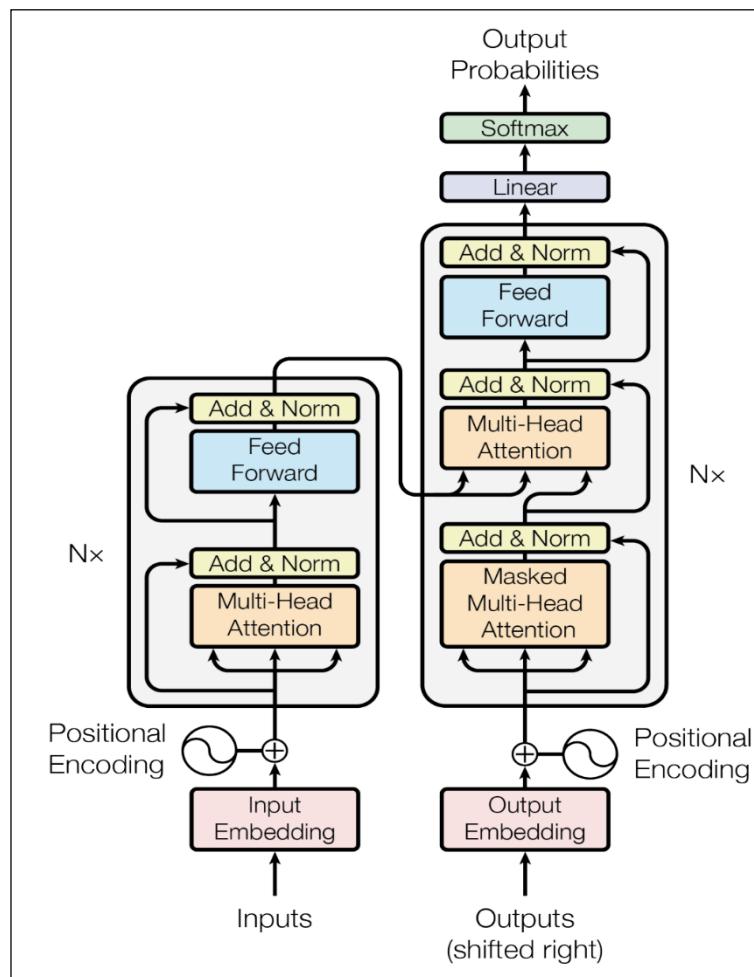
Ci-après le fichier XML montrant la nouvelle structure de ce corpus.

```
<?xml version="1.0" ?>
<corpus>
    <instance id="1">
        <q> من هو مخرج فيلم الملكة الافريقية؟ </q>
        <a>جون هيستون</a>
    </instance>
    <instance id="2">
        <q> من هو الملك الفرنسي الذي وقع التحالف العسكري الامريكي الفرنسي عام 1778 ؟ </q>
        <a>لويس السادس عشر</a>
    </instance>
    <instance id="3">
        <q> من أسس منظمة ; مؤسسة المسجد الاسلامي ؟ </q>
        <a>مالكوم اكس</a>
    </instance>
    <instance id="4">
        <q> من أسس الدولة الرستمية ؟ </q>
        <a>عبد الرحمن بن رستم</a>
    </instance>
    <instance id="5">
        <q> من اخترع التلسكوب الانكاري ؟ </q>
        <a>جاليليو</a>
    </instance>
    <instance id="6">
        <q> من ألف كتاب الميدلة بن تساو ؟ </q>
        <a>شن تونج</a>
    </instance>
```

**Figure 3.13** La nouvelle structure du corpus questions-réponses factoïdes arabes TALAA-AFAQ

Le corpus contient 2001 questions-réponses, mais cela n'a pas été assez suffisant pour l'apprentissage d'un modèle complexe, nous allons discuter ce problème dans le chapitre qui suit.

**Processus d'apprentissage** : une fois le corpus passé au transformateur, l'apprentissage commence. En effet, les entrées textuelles sont encodées par rapport à leur position dans la chaîne de caractères, avec les équations (Équation 3.2), pour donner au modèle des informations sur leurs positions. Le vecteur d'encodage de positions est donc ajouté à l'entrée, puis, il est passé au capteur d'attention, pour ensuite continuer vers l'étape d'apprentissage qui génère une sortie. Cette dernière est considérée comme une entrée pour la prochaine itération/epoch d'apprentissage (dernière itération).



**Figure 3.14** Architecture du transformateur utilisé [52]

$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

**Équation 3.2** Formules d'encodage de position [52]

Le résultat après l'apprentissage est un modèle qui prend une phrase textuelle arabe et génère une réponse correspondante en langage naturel. Une évaluation de ce modèle sera faite dans le dernier chapitre.

**Sortie :** le module de génération de réponse fournit 2 sorties selon la classification des requêtes, soit une réponse basique, qui consiste à exécuter directement la requête avec un petit message correspondant (pour les requêtes de niveau 0 et 1 ayant l'intention de lancer une application donnée), soit une réponse avancée (pour les requêtes de niveau 2 ayant l'intention de démarrer une conversation, ou d'avoir des réponses à des questions).

### 3.3.5 Module de synthèse vocale (M6)

L'assistant vocal n'a un sens que s'il garde le processus conversationnel de bout en bout. De ce fait, nous avons ajouté le module de synthèse vocale (Text-To-Speech), qui correspond au feedback du back-end, pour offrir une interface dans le but d'interagir avec l'utilisateur. En effet ce module permet de lui communiquer les réponses du système à travers une voix synthétique.

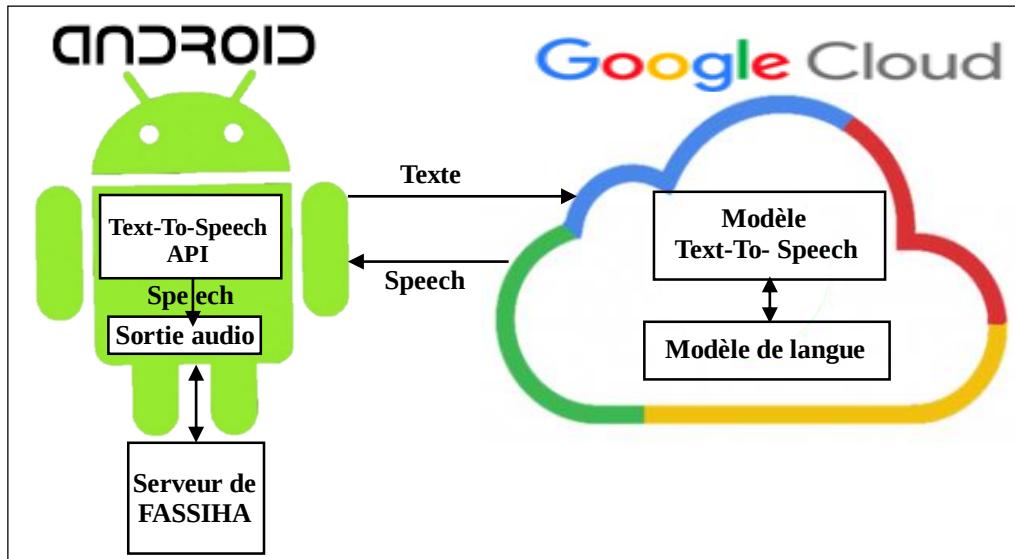
Pour déclencher le processus de synthèse vocale, la réponse doit être déjà générée par le module précédent (M5).

Ce module consiste à convertir les entrées textuelles en une voix synthétique. Pour ce faire, nous avons utilisé l'Application Programming Interface (API) TextToSpeech de Google qui fournit cette fonctionnalité, nous l'avons juste adaptée à la langue arabe.

Cette API TextToSpeech est développée par Google, donc elle est parfaitement intégrable avec les applications Android, une chose qui atteste encore la fiabilité de notre système (procédure inverse du module de reconnaissance vocale).

**Architecture de l'API TextToSpeech de Google :** Google fournit une API pour la synthèse vocale avec les APIs de développement Android, cette API est une interface pour l'utilisation du système de synthèse vocale Google Cloud. L'API TextToSpeech est basée sur un modèle de deep learning pour générer le speech (discours). En effet, la consommation de ce service à partir d'Android ne nécessite que l'utilisation de l'interface TextToSpeech pour générer une réponse audio.

**Modèle de langue** : le modèle a à sa disposition l'un des meilleurs modèles de langue de Google [3], ce modèle de langue permet la construction de l'HMM intégré avec le modèle de synthèse vocale pour améliorer la précision et notons que la langue arabe est supportée nativement par Google.



**Figure 3.15** Architecture du module de synthèse vocale

**Sortie** : ce module transforme la réponse générée par le module de génération de réponses en une réponse audio. Par la suite, cette réponse est envoyée au module (M7) de l'interface utilisateur (Front-end), pour la communiquer à l'utilisateur à travers une voix synthétique (lecture vocale).

Notons que la réponse en format texte sera aussi affichée dans l'interface utilisateur pour des raisons déjà discutées.

### 3.4 Conclusion

Ce chapitre a donné une vision sur notre travail et a montré l'aspect conceptuel de notre assistant vocal « FASSIHA » à travers les différentes architectures des modules du front-end et du back-end. Le chapitre qui suit fera l'objet de l'implémentation de notre application.

# 4

Chapitre

# IMPLÉMENTATION ET ÉVALUATION

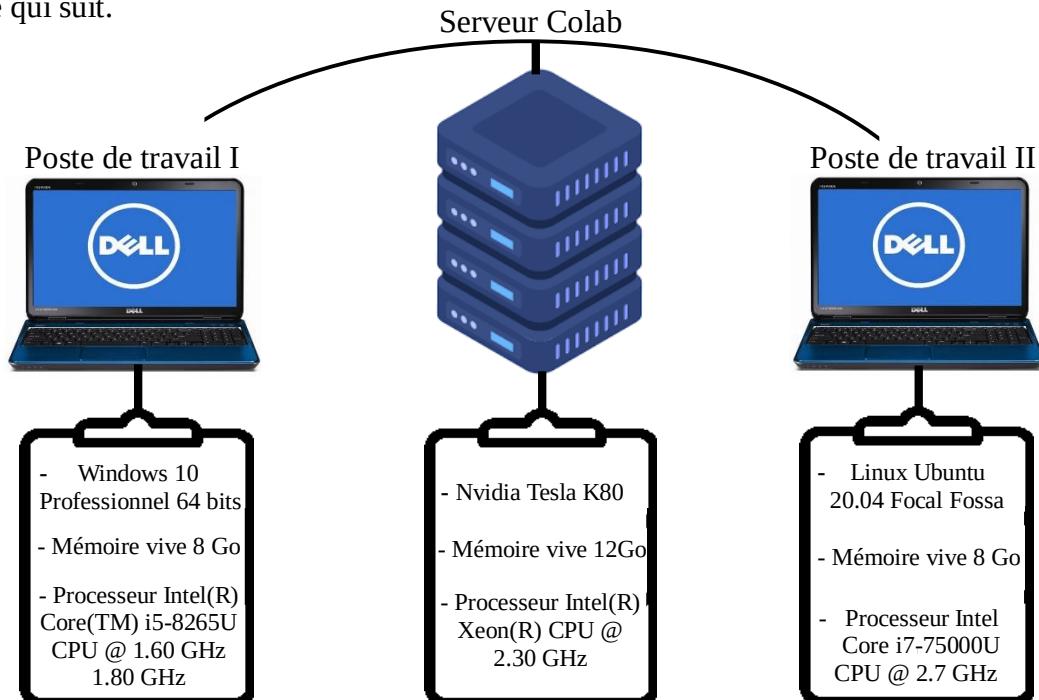
Après avoir achevé l'étape de la conception, ce chapitre vient traiter les aspects techniques liés à l'implémentation et la mise en œuvre de notre assistant vocal virtuel supportant la langue arabe «FASSIHA», ainsi que les aspects expérimentaux permettant l'évaluation de notre système. Nous présentons d'abord l'environnement matériel et l'environnement logiciel utilisé pour le déploiement de « FASSIHA ». Ensuite nous détaillerons les méthodes et la façon dont nous avons implémenté chaque module cité dans le chapitre précédent. Enfin, nous allons évaluer les modules essentiels et discuter les différents résultats.

## 4.1 Environnement de développement

Pour la réalisation de ce projet, nous avons eu recours aux environnements suivants.

### 4.1.1 Environnement matériel

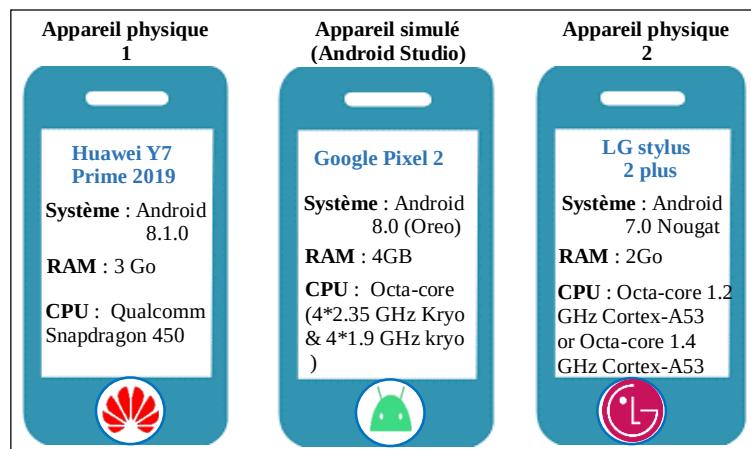
Le développement de notre système passe par plusieurs étapes (apprentissage automatique, développement des modules, développement de l'interface Android, etc.). Ces étapes nécessitent un matériel (machines) ayant différentes architectures et puissances de calcul). En effet, les caractéristiques des machines utilisées pour le développement sont illustrées dans la figure qui suit.



**Figure 4.1** Caractéristiques des machines

Pour tester l'application mobile, il faut s'assurer de sa couverture pour la majorité des versions Android intégrées dans différents appareils mobiles des différents fabricants. Pour ce faire, nous avons testé notre application sur plusieurs appareils physiques et simulés.

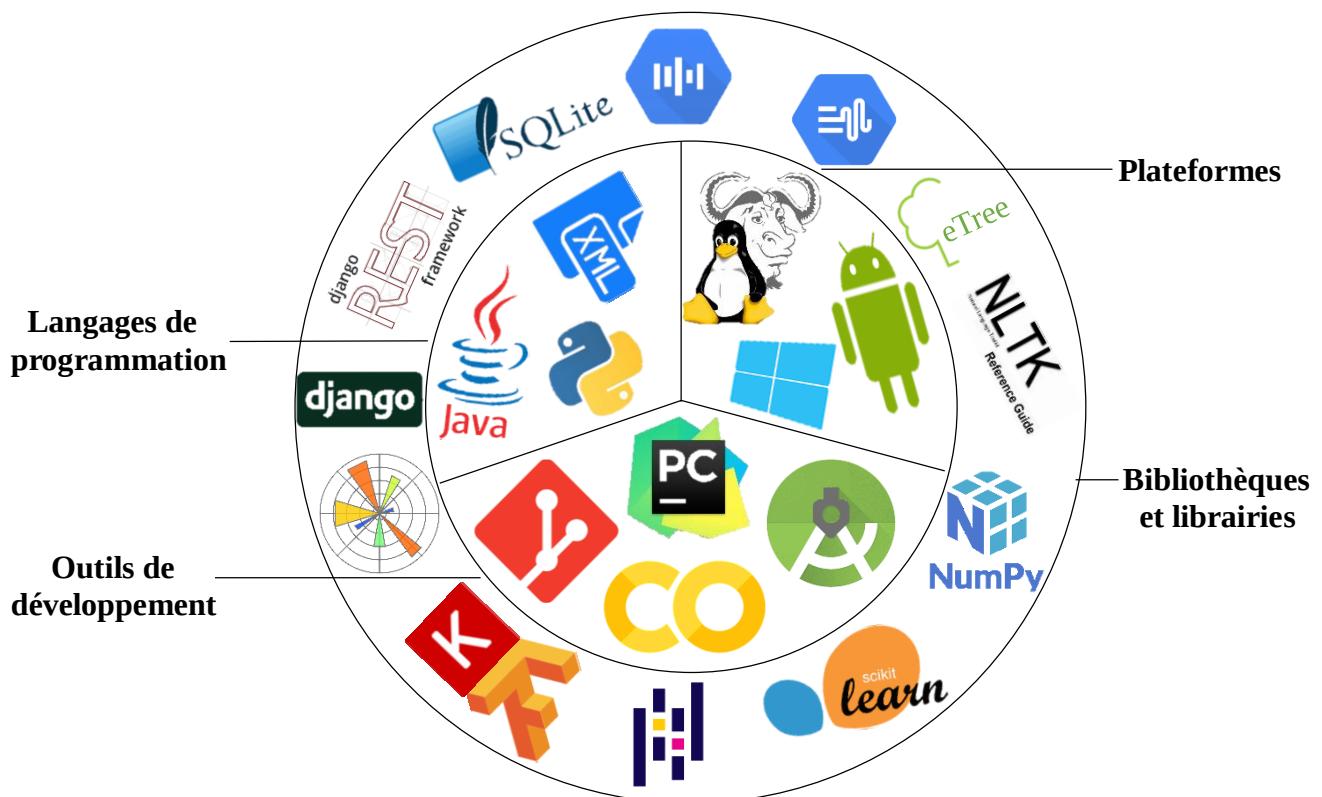
L'ensemble des appareils de test est dans la figure ci-après.



**Figure 4.2** Caractéristiques des appareils de test

#### 4.1.2 Environnement logiciel

Pour la partie logicielle, une liste non exhaustive est présentée juste après, qui ne mentionne que les logiciels et bibliothèques les plus utilisés.



**Figure 4.3** Logos des différents logiciels utilisés

#### 4.1.2.1 Plateformes

##### GNU/Linux

GNU/Linux<sup>5</sup> est une famille des systèmes d'exploitation UNIX-Like, totalement open-source, il offre une alternative complète des systèmes propriétaires à l'exemple de Windows. GNU/Linux est maintenu principalement par FSF (Free Software Foundation), mais c'est toute une communauté open-source qui aide au développement de ce système. GNU/Linux est le système le plus utilisé dans les serveurs dû à sa rapidité et à sa sécurité. En effet, il offre un système complet au développeur pour tester et lancer les différents types d'applications (web, mobile, etc.) avec une architecture simple et une interface facile d'utilisation.

##### Android

Android<sup>6</sup> est un système d'exploitation mobile, développé et maintenu par Google. Android est basé sur un Kernel de Linux. Le système d'exploitation le plus utilisé actuellement dans le monde. En effet, la plus part des fabricateurs mobiles ont adopté Android comme un système par défaut dans leurs appareils. Google offre une variété de bibliothèques pour le développement Android, ce qui facilite les tâches aux développeurs. Android exécute une version spéciale de JVM (Java virtual machine), ce qui fait de Java le langage standard pour le développement sur cette plateforme (Kotlin est ajouté récemment comme un langage standard Android, s'exécutant sur la même JVM que Java).

#### 4.1.2.2 Langages de programmation

##### Python

Python<sup>7</sup> est un langage de programmation interprété de haut niveau, structuré et open source créé par Guido van Rossum. Python se caractérise par la lisibilité de son code et la facilité de sa maintenance, ce qui permet au développeur de se concentrer sur la qualité du logiciel sans perdre du temps avec les opérations de bas niveau (récupération de mémoire, allocation de mémoire, etc.), c'est pour cette raison que python est connu pour la rapidité de son cycle de

---

<sup>5</sup> <https://en.wikipedia.org/wiki/Linux>

<sup>6</sup> [https://en.wikipedia.org/wiki/Android\\_\(operating\\_system\)](https://en.wikipedia.org/wiki/Android_(operating_system))

<sup>7</sup> [https://en.wikipedia.org/wiki/Python\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))

développement. Ainsi, puisque python est open-source, il a permis de développer un écosystème de bibliothèques (packages), variant des bibliothèques dédiées au développement d'applications mobiles, web et réseaux jusqu'aux bibliothèques très avancées dédiées à l'apprentissage automatique et à l'apprentissage profond. En effet, python aujourd'hui est le numéro un dans le développement des modèles de l'IA.

## Java

Java<sup>8</sup> est un langage de programmation compilé et orienté objet. Il est développé avec la philosophie WORA (Write Once Run Anywhere), qui veut dire qu'un code java peut être exécuté sur plusieurs plateformes sans avoir besoin de récrire le code source. Java est développé initialement par James Gosling chez Sun Microsystems et lancé en 1995, il est considéré comme un langage standard de développement des applications Android, et ce, est dû à sa rapidité et portabilité, tels qu'il permet l'utilisation des ressources et des bibliothèques disponibles dans les appareils mobiles, tout en cachant la partie de compilation et de l'exécution du code au développeur, ce qui permet à ce dernier de se concentrer uniquement sur la création de son application.

## XML

XML (Extensible Markup Language)<sup>9</sup> est un langage de balisage qui définit un ensemble de règles d'encodage des documents dans un format lisible par l'être humain et la machine en même temps. Il permet aussi de stocker les données dans un format compact, réduisant la taille de ces dernières. Ainsi, il existe de très rapides bibliothèques de manipulation de XML. En effet, XML est le langage de définition des interfaces sur Android.

### 4.1.2.3 Bibliothèques et librairies

#### SpeechRecognizer

SpeechRecognizer<sup>10</sup> est une API d'Android permettant l'interaction avec le service de reconnaissance vocale de Google Cloud, elle offre la possibilité de personnaliser les différentes variables de reconnaissance (Langue, Temps d'audition, etc.).

<sup>8</sup> [https://en.wikipedia.org/wiki/Java\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Java_(programming_language))

<sup>9</sup> <https://en.wikipedia.org/wiki/XML>

<sup>10</sup> <https://developer.android.com/reference/android/speech/SpeechRecognizer>

## TextToSpeech

TextToSpeech<sup>11</sup> est une API d'Android permettant l'interaction avec le service de synthèse vocale de Google Cloud.

## ElementTree

ElementTree<sup>12</sup> est une bibliothèque de python dédiée à la création et à la manipulation des fichiers XML, elle est rapide et facile d'utilisation et offre ainsi une implémentation en langage C (cElementTree), qui est rapide aussi dans la manipulation de XML.

## NLTK

NLTK<sup>13</sup> est un ensemble de bibliothèques python conçu pour le traitement statistique et symbolique du langage naturel. Il propose une très grande variété de tâches pour le TALN permettant au développeur de traiter le langage naturel en écrivant un minimum de code.

## Numpy

Numpy<sup>14</sup> est une bibliothèque de python spécialisée dans la manipulation des grandes masses de données numériques, notamment les vecteurs unidimensionnels (tableaux) et multidimensionnels (matrices). Les opérations sur ces derniers s'exécutent avec une très grande rapidité et efficacité de calcul. Numpy est très utilisée dans l'apprentissage automatique et se base généralement sur le calcul matriciel. En effet, elle offre des structures de données compatibles avec beaucoup d'autres librairies telles que Keras et Tensorflow.

## Scikit-Learn

Scikit-Learn<sup>15</sup> est une bibliothèque open-source de python conçue pour le développement rapide des modèles de l'apprentissage automatique, elle est équipée avec une variété d'algorithmes de classification, de régression et de regroupement. Scikit-Learn est principalement utilisée pour ses nombreux outils de prétraitement de données (normalisation, codification, filtrage, etc.). En effet, cette bibliothèque est idéale pour le développement rapide des modèles de test.

<sup>11</sup> <https://developer.android.com/reference/android/speech/tts/TextToSpeech>

<sup>12</sup> <https://docs.python.org/2/library/xml.etree.elementtree.html>

<sup>13</sup> <https://www.nltk.org/>

<sup>14</sup> <https://numpy.org/>

<sup>15</sup> <https://scikit-learn.org/stable/>

## Pandas

Pandas<sup>16</sup> est une bibliothèque de python conçue pour la manipulation et l'analyse de données. Elle est très utilisée dans le prétraitement des données d'apprentissage automatique.

## Tensorflow/Keras

Tensorflow<sup>17</sup> est une bibliothèque open-source développée par Google qui est dédiée à l'apprentissage automatique et plus particulièrement aux réseaux de neurones et l'apprentissage profond. Optimisée pour exécuter des opérations massivement distribuées sur un réseau. Tensorflow offre la possibilité d'implémenter une grande variété d'architectures de modèles avec un maximum d'efficacité en exploitant le maximum de ressource disponibles, comme les cartes graphiques de hautes performances, ce qui permet le développement de modèles très complexes en un temps minimum. Quant à Keras<sup>18</sup>, elle est généralement utilisée comme interface, pour rajouter de l'abstraction à Tensorflow. En effet, Keras est un package python permettant l'interaction avec Tensorflow pour faciliter le développement des modèles d'apprentissage, tout en offrant la possibilité de personnaliser les fonctionnalités par défaut.

## Matplotlib

Matplotlib<sup>19</sup> est une bibliothèque de python conçue pour le traçage des courbes et des fonctions mathématiques. Elle permet la visualisation des données et du processus de calcul (par exemple l'apprentissage automatique).

## Django

Django<sup>20</sup> est un framework web python basé sur une architecture MVC (Model-View-Controller). Il est conçu pour le développement rapide des sites et des applications web, le processus de développement est entièrement en python. En effet, Django est l'idéal pour concevoir des applications web interagissant avec des bases de données.

## Django REST Framework

---

<sup>16</sup> <https://pandas.pydata.org/>

<sup>17</sup> <https://www.tensorflow.org/>

<sup>18</sup> <https://keras.io/>

<sup>19</sup> <https://matplotlib.org/>

<sup>20</sup> <https://www.djangoproject.com/>

Django REST Framework<sup>21</sup> est un framework conçu pour la création des API REST (Application Programming Interface REST) dans le cadre de Django. En effet, la création de telles API est similaire à la création d'une vue (View) avec Django. Il permet aussi la manipulation totale des caractéristiques de l'API REST (Port réseau ouvert, type des données des requêtes, etc.).

## SQLite

SQLite<sup>22</sup> est le système de gestion des bases de données relationnelles (SGBDR). Il est intégré par défaut avec Django. SQLite est parfaitement intégrable avec les applications sans avoir besoin d'être installé en mode autonome.

### 4.1.2.4 Outils de développement

#### PyCharm

PyCharm<sup>23</sup> est un environnement de développement intégré et spécialisé pour programmer en langage python. Il est équipé d'un ensemble de fonctionnalités permettant de développer tout un projet en python sans avoir besoin d'un autre outil. PyCharm permet l'analyse de code en continu et propose un débogueur intégré pour exécuter le code instruction par instruction. Ainsi, il supporte le framework web Django et le gestionnaire des versions Git.

#### Android Studio

Android Studio<sup>24</sup> est l'environnement officiel pour le développement d'applications Android. Il est supporté par Google et est équipé de tout ce qui est nécessaire pour le développement Android (Debugger, Interface Designer, Android Library support, etc.). Android Studio supporte le développement en Java ou en Kotlin, les deux langages officiels d'Android. En effet, l'une des fonctionnalités les plus importantes d'Android Studio est l'émulateur, ce dernier permet de tester des applications sur des appareils logiques simulés par le système, cette fonctionnalité aide le développeur dans le test des applications sur des différentes architectures, versions et types d'appareils, et ce, sans avoir besoin d'avoir un appareil physique.

---

<sup>21</sup> <https://www.djangoproject.com/>

<sup>22</sup> <https://www.sqlite.org/index.html>

<sup>23</sup> <https://www.jetbrains.com/pycharm/>

<sup>24</sup> <https://developer.android.com/studio/>

## Google Colaboratory

Google Colaboratory<sup>25</sup> ou Colab, est un environnement python pour le développement et l'exécution de Jupyter Notebooks<sup>26</sup> sur le Cloud. Il offre la puissance de calcul des serveurs de Google avec un environnement de développement idéal pour accélérer l'apprentissage. En effet, Colab ne nécessite aucune configuration et son utilisation peut énormément réduire le temps d'apprentissage et donner un espace de stockage et de traitement de données sur le Cloud.

## Git

Git<sup>27</sup> est un système de gestion des versions décentralisées permettant entre autres de garder les toutes les versions d'un projet pendant son développement, de tester chaque version avant d'appliquer les changements, de garder l'historiques des modifications effectuées et surtout d'assurer un environnement de collaboration et de travail en parallèle entre les développeurs d'un projet.

Dans le reste de ce chapitre, nous allons détailler la phase d'implémentation de chacun des modules de notre assistant vocal Fassiha, munie d'une phase d'évaluation pour les modules de reconnaissance vocale et de gestion de dialogue, ainsi que pour l'application. La phase d'évaluation permettra d'estimer l'efficacité de notre système, et cela, en utilisant des métriques spécifiques pour chaque module, ce qui va être un précurseur pour les futures améliorations.

## 4.2 Module de reconnaissance vocale

Comme nous l'avons mentionné dans la conception, nous avons exploité le service *Google Cloud* pour l'implémentation du module de reconnaissance vocale. *Google Cloud* service peut être demandé à partir de n'importe quelle plateforme (Windows, Linux, Android, etc.). Ce service utilise un système d'interfaces (API's) pour interagir avec les différentes plateformes. Puisque nous développons notre application sous Android, nous avons utilisé l'API d'Android « *SpeechRecognizer* ». L'application doit avoir un accès à Internet pour pouvoir envoyer les requêtes reçues de l'utilisateur au service Cloud. En effet la permission d'accès à Internet doit être accordée à l'application en rajoutant la balise :

---

<sup>25</sup> <https://colab.research.google.com/>

<sup>26</sup> <https://jupyter.org/>

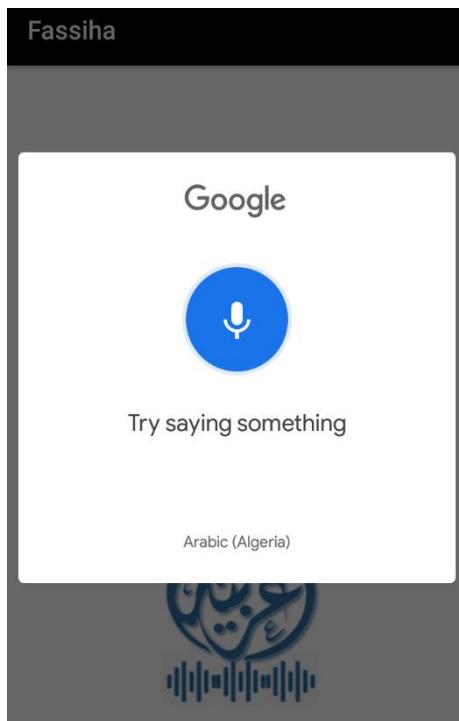
<sup>27</sup> <https://git-scm.com/>

<uses-permission android:name="android.permission.INTERNET/> dans le fichier **AndroidManifest.xml**. Pour s'initier à la reconnaissance vocale, il faut lancer un simple *Intent* (intention) d'Android. Un événement doit être associé à l'intention (clic sur un bouton par exemple), en spécifiant ainsi la langue désirée.

```
Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
                    RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE, "ar");
```

**Algorithm 4.1** Démarrage du processus de reconnaissance vocale arabe

L'intention démarre donc à chaque occurrence de l'événement spécifié. Une boîte de dialogue se lance en demandant à l'utilisateur une entrée vocale.



**Figure 4.4** Boîte de dialogue demandant une entrée vocale

L'entrée vocale est ensuite transmise au service de la transcription textuelle. L'application attend la réception de la réponse. La réception de la réponse peut échouer dans le cas d'absence ou d'insuffisance du service Internet, ou dans certains anciens appareils et versions Android qui ne supportent pas des entrées vocales, sinon, la réponse est capturée et est transmise au prochain module.

## 4.2.1 Expérimentations et évaluation

La reconnaissance vocale représente le point d'entrée de l'application. De ce fait, chaque erreur dans la reconnaissance de la commande vocale de l'utilisateur, va mener à la mauvaise transcription de cette dernière, ce qui va influencer négativement sur les prochaines étapes de traitement. Pour cette raison, il est nécessaire de tester ce module et de bien l'évaluer pour s'assurer de sa fiabilité et être certain qu'il ne va pas causer des inconsistances dans les étapes suivantes.

### 4.2.1.1 Données expérimentales (ensemble de test)

Les résultats de la reconnaissance vocale peuvent être affectés par plusieurs facteurs, dont voici les plus importants : le bruit, le dialecte, l'accent et la vitesse de la parole de l'utilisateur. Ce qui rend la construction d'un ensemble de test complet très difficile. Bien que plusieurs recherches effectuées sur le système de reconnaissance vocale de Google aient démontré la fiabilité de ce dernier, il était clair aussi que notre système présente des particularités, dont la première est l'utilisation de la langue arabe et la deuxième concerne le type des requêtes (des courtes commandes). Nous avons donc décidé de construire un mini corpus pour l'évaluation minimal de notre système. Notre corpus est composé de 201 enregistrements vocaux au format (**.wav**) et la longueur maximale de chaque enregistrement est de 5 secondes. En effet, nous avons choisi d'attribuer 67 enregistrements pour chaque niveau de requêtes (niveau 0,1 et 2). Les requêtes ont été enregistrées équitablement dans 2 environnements distincts (50% bruyant, 50% non bruyant) et ont été prononcées équitablement par 2 types de locuteurs (50% féminin et 50% masculin). Chaque enregistrement vocal est associé à sa transcription textuelle correcte (**.txt**), afin de pouvoir faire l'évaluation.

### 4.2.1.2 Métriques d'évaluation

La métrique la plus utilisée pour évaluer les systèmes de reconnaissance vocale est la métrique WER (Word Error Rate), La formule de calcul du WER est la suivante:

$$WER(x; y) = \frac{S+I+D}{N} .$$

**Équation 4.1** La métrique WER pour l'évaluation des systèmes de reconnaissance vocale

x : La chaîne de caractère référence.

y : La chaîne de caractère prédite (hypothèse).

S : Nombre de substitutions en mots entre l'hypothèse et la référence.

I : Nombre d'insertions en mots entre l'hypothèse et la référence.  
 D : Nombre de suppressions en mots entre l'hypothèse et la référence.  
 N : Longueur en mots de la séquence de mots.

Chaque enregistrement est sauvegardé dans un fichier ayant un nom (l'ID de l'enregistrement) et correspondant à l'ID du fichier contenant la transcription textuelle.

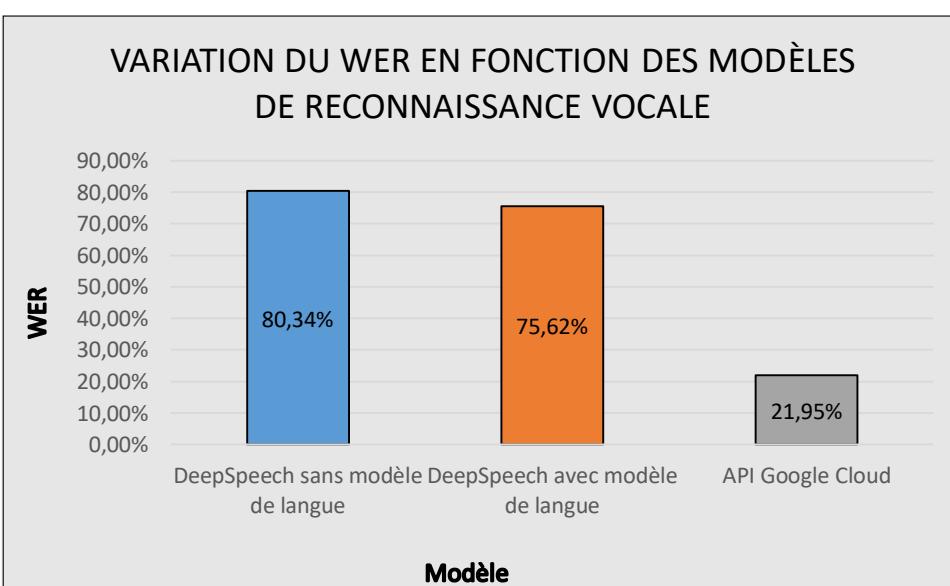
Un script python est lancé pour calculer le WER et évaluer chaque instance (enregistrement). En effet, la valeur du WER varie entre 0 et 1. Plus cette valeur est petite et se rapproche de 0, plus la précision est grande et meilleure. Notons ainsi ces deux cas particuliers :

- Si  $WER=0$  alors la commande vocale (l'enregistrement) est transcrit en texte correctement à 100% (0 substitutions, 0 insertions, 0 suppressions).
- Si  $WER=1$  la commande est mal transcrit en texte à 100%.

#### 4.2.1.3 Résultats et discussion

Pour pouvoir évaluer n'importe quel système, il faut toujours le comparer à d'autres systèmes similaires. De ce fait, pour pouvoir évaluer notre modèle de reconnaissance vocale « *API Google Cloud* », nous l'avons comparé à deux modèles analogues que nous avons développés et testés auparavant mais qui n'avaient pas donné des résultats satisfaisants. Ceci est dû à l'entraînement de ces deux modèles sur le corpus « *Common Voice by Mozilla* », mais les données de ce corpus n'étaient pas suffisantes pour atteindre les résultats souhaités. Ces deux modèles sont ceux de *DeepSpeech*, dont le premier est sans modèle de langue et le deuxième est avec le modèle de langue.

Nous avons calculé le WER moyen pour chacun des 3 modèles cités et nous avons eu ces résultats représentées dans l'histogramme venant ci-après.



**Figure 4.5** Histogramme montrant la variation du WER en fonction de 3 modèles de reconnaissance vocale

L'histogramme montre clairement que le modèle que nous avons adopté dans la reconnaissance vocale « *API Google Cloud* » est le meilleur et le plus performant d'après les valeurs du WER. La valeur du WER calculée pour « l'*API de Google Cloud* » est à peu près 4 fois plus petite que celles calculées pour les modèles de *DeepSpeech*. Nous interprétons ceci comme signifiant que « l'*API de Google Cloud* » donne des résultats approximativement 4 fois plus pertinents et améliorés que les deux autres modèles de *DeepSpeech*. Cette forte pertinence de résultats obtenus par notre système de reconnaissance vocale « *API Google Cloud* » est justifiée par le fait que Google est parfaitement intégrable avec Android.

### 4.3 Module de TALN et structuration des requêtes

Nous avons discuté dans les chapitres précédents la complexité de la langue arabe, qui nécessite une phase de normalisation et de simplification des requêtes textuelles. Le module de TALN et structuration des requêtes chargé d'effectuer cette tâche.

L'entrée de ce module est la transcription textuelle de la commande vocale de l'utilisateur fournie par *Google Cloud*, cette commande est écrite dans un fichier ayant une extension .txt. Ce fichier est ensuite communiqué au 1<sup>er</sup> sous-module, celui de la normalisation des requêtes. Les commandes transmises par le module de reconnaissance vocale sont simples et ne contiennent ni de ponctuation, ni de diacritiques. De ce fait, la normalisation consiste à éliminer de la Hamza (ء) et le Alif Lam (ا) et la gestion des mots vides. Les mots vides ont tendance à causer des incohérences dans le traitement du langage et la recherche d'information. Pour cette raison, nous les avons étiquetés, afin de les omettre dans les étapes suivantes du traitement de la requête et ainsi, ces derniers n'auront pas de poids lors de la génération de réponse et ne vont pas affecter la pertinence des résultats finaux retournés à l'utilisateur. À cet effet, nous avons utilisé l'ensemble des mots vides pour l'arabe de la bibliothèque NLTK de Python.

Après nous sommes passés au 2<sup>ème</sup> sous-module, celui de la tokenisation et de la racinisation (stemming). Nous avons utilisé la fonction **WordPunctTokenizer** de la bibliothèque NLTK pour tokeniser la commande. Cette fonction donne des résultats assez satisfaisants pour la langue arabe. Ensuite nous avons associé à chaque token un stem. Pour faire le stemming, nous avons utilisé **ISRIStemmer** de NLTK.

Ces informations (commande tokenisée et racinisée) sont ensuite passées au 3<sup>ème</sup> sous-module, celui de la structuration des requêtes, qui assure le regroupement de toutes ces informations

sous un format rapidement accessible, en minimisant la taille de stockage (la limite des ressources c'est l'une des choses qu'il faut garder en tête lors du développement d'une application mobile). Nous avons donc décidé de stocker les informations concernant la requête dans un fichier XML, pour les retrouver facilement plus tard. La figure 3.5 montre la structure finale de l'exemple précédent (فتح تطبيق الطقس) (اقتحم تطبيق الطقس) à la fin de cette étape.

## 4.4 Module de classification des requêtes

Nous avons illustré dans le chapitre précédent notre méthode pour le traitement des requêtes. En effet, Nous avons classifié les requêtes en trois niveaux selon leur complexité, dans le but d'implémenter une méthode de traitement indépendante et différente pour chaque niveau. Ceci a été l'objectif du module de classification. Dans cette section, nous allons discuter l'implémentation de ce module en listant les avantages et les inconvénients de chaque niveau.

L'entrée de ce module c'est bien le fichier structuré généré dans l'étape précédente (Figure 3.5). La classification est lancée par un module python principal. Ce module lit le fichier XML lié à la requête, extrait la classe de la requête et l'application correspondante, puis envoie ces informations aux deux générateurs : générateur de niveau 0 et générateur de niveau 1.

### 4.4.1 Classifieur de niveau 0

C'est un classifieur symbolique, i.e. il se base sur des règles pour identifier les requêtes les plus fondamentales qui respectent une structure basique, claire et précise (ce type de commandes serait utile pour les utilisateurs qui veulent rapidement exécuter une tâche simple et bien précise depuis leur appareil). Donc ce type de requêtes va être exécuté très rapidement. Cette rapidité est assurée par le module de classification qui fonctionne en synergie avec le module de génération de réponses.

```
"""This file provide a set of rules to be parsed by the level0
requests classifier"""

RULES = [ 'S->VERBAL_PHRASE',
          'VERBAL_PHRASE->VERB/VERB OBJECTS',
          'OBJECTS->TIME_OBJECT/PLACE_OBJECT/APP_OBJECT/OBJECTS'
        ]

# FINALS should be without 'ال' and without the Hamza
FINALS = [ 'VERB-' + 'اخفي/اعرض/انقر/اقفل/ابدا/اطلق/اخراج/افتح/اغلاق/ادخل/ابحث',
           ' TIME_OBJECT-' + 'سيت/بارحة/يوم/بعد غد/بارحة/غدا/امس/الآن' ]
```

```

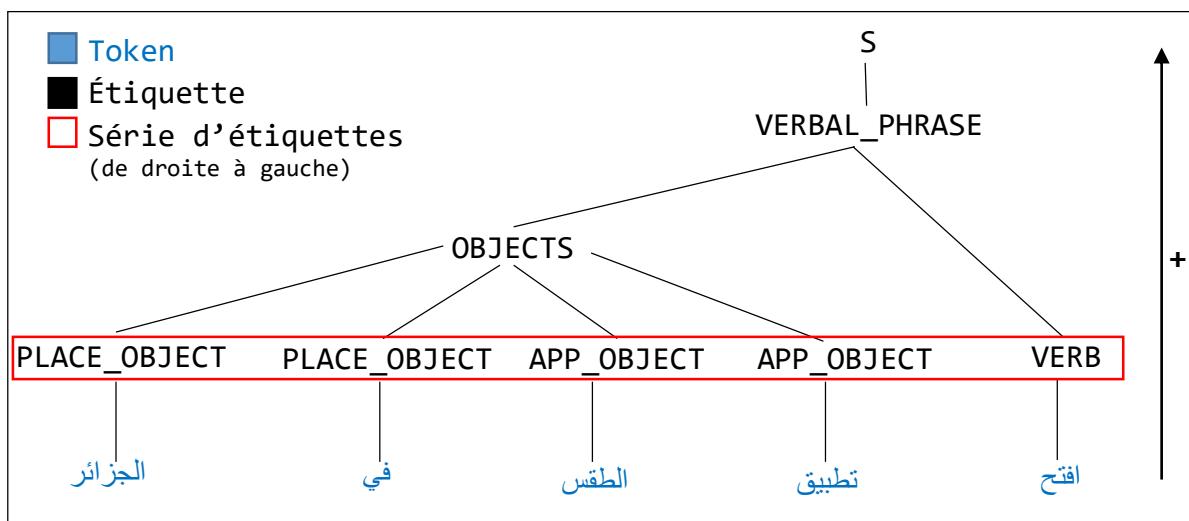
    ، الواحدة/ ثانية/ دقيقة/ ساعة/ على/ في/ شهر
    / صحراء/ خارج/ داخل/ ساحل/ عاصمة/ هنا/ جزائر/ امريكا->
    ، إلى/ في/ من
    'تطبيق/ اتصال/ اذاعة/ آلة حاسبة/ موسيقى/ منبه/ طقس/ ساعة/ لعبة->
    ]

```

**Algorithm 4.2** Extrait de la base de règles associée au classifieur de niveau 0

La liste des terminaux et des non terminaux peut être mise à jour à n'importe quel moment, cette mise à jour n'affecte pas le système.

Dans la base de règles, des étiquettes sont associées aux tokens, et ainsi, le classifieur effectue une analyse de bas en haut (bottom-up) pour reconstruire une série d'étiquettes et vérifier la validité de la requête. Si la requête est valide (respecte les règles), il envoie la classe 0, l'application correspondante ainsi que la série d'étiquettes associée à la requête au module suivant pour en extraire les arguments, sinon la requête sera jugée comme non valide pour le niveau 0, elle sera envoyée donc au classifieur du prochain niveau (niveau 1).



**Figure 4.6** Analyse Bottom-up sur un exemple de requête de niveau 0

#### 4.4.2 Classifieur de niveau 1

C'est un classifieur à base de connaissances, qui s'appuie sur une base de connaissances extensible que nous avons nous-même construite. Cette base se compose de plusieurs fichiers XML, dont chacun est associé à une application précise (météo, réveil, calculatrice, etc.). Ces fichiers XML contiennent des mots clés (composés et non composés) liés au domaine de l'application (Figures 3.8 et 3.9).

En effet, comme nous avons précisé dans la conception, le classifieur de niveau 1 calcule le score de similarité en deux manières distinctes selon l'absence ou la présence des conflits.

1) **Cas d'absence de conflits** : nous calculons les scores de similarité entre les mots de la requête et les mots des fichiers associés aux applications, avec la formule  $N/T$  où

$N$  : est le nombre de tokens de la requête qui figurent dans les fichiers XML (datasets) associés aux applications.

$T$  : est le nombre total des mots de la requête (mots vide négligés).

2) **Cas de présence de conflits** : nous avons gardé la même formule de calcul de similarité précédente, sauf que cette fois-ci les mots de la requête et ceux des datasets sont racinisés. Nous avons testé plusieurs approches et nous avons constaté qu'en faisant le stemming, les conflits disparaissent dans la plupart des cas, parce que le stemming a tendance à rapprocher les mots du même domaine.

Après avoir calculé les scores de similarité, Si la requête est de niveau 1, alors le module envoie la classe 1 et l'application correspondante au module suivant pour en extraire les arguments, sinon la requête sera envoyée au classifieur du prochain niveau (niveau 2).

#### 4.4.3 Classifieur de niveau 2

Dans le cas où la requête n'est ni de niveau 0 ni de niveau 1, elle sera automatiquement affectée au niveau 2, pour faire un apprentissage automatique (section suivante).

Toutes les informations extraites dans ce module (pour les requêtes de niveau 0 et 1) sont envoyées au module suivant pour extraire les arguments de la requête et générer une réponse.

### 4.5 Module de génération de réponses

Après avoir classifié la requête, il est temps d'extraire ses majeurs arguments et générer une réponse pour cette requête afin d'accomplir la tâche demandée par l'utilisateur et satisfaire sa demande.

Le module de génération de réponse produit le résultat final qui sera communiqué à l'utilisateur. Donc, la qualité de ce module a un impact non négligeable sur la qualité de l'application et sur l'impression de l'utilisateur envers l'application.

En effet, puisque nous avons classifié nos requêtes en 3 classes selon leur complexité, nous avons aussi décomposé le générateur de réponses aussi en 3 générateurs distincts, dont chacun est responsable d'une classe de requêtes. Dans ce qui suit, nous allons détailler l'implémentation et évaluer chaque générateur à part, pour discuter ensuite les résultats et la qualité de réponses générées.

#### 4.5.1 Générateur de réponses de niveau 0

La génération de réponses pour les requêtes de la classe de niveau 0 est très simple ; le module reçoit du classifieur l'application à exécuter, il la lance, puis il génère une réponse standard appropriée à l'application. Android autorise le développeur à lancer d'autres applications externes déjà installées dans l'appareil, en utilisant un Intent. Le code ci-après représente montrer comment exécuter une application externe sous Android (ici météo). Notons qu'il faut avoir le nom du package de l'application pour pouvoir l'exécuter. Ceci est fait à travers le répertoire des applications installé sur Android, qui fournit les noms de tous les packages.

```
Intent launchIntent =
getPackageManager().getLaunchIntentForPackage("com.package.weather")
;
if (launchIntent != null) {
    startActivity(launchIntent);
}
```

**Algorithm 4.3** Lancement d'une application externe sous Android

Après l'exécution de l'application demandée, la réponse standard générée est envoyée directement au prochain module (Text-To-Speech) pour la communiquer à l'utilisateur sous forme d'une voix synthétique.

#### 4.5.2 Générateur de réponses de niveau 1

La génération de réponse pour les requêtes de la classe de niveau 1 est similaire au niveau 0, sauf que ce niveau de requêtes donne plus de choix et de liberté à l'utilisateur. En effet, ce dernier peut effectuer des commandes flexibles avec plus d'arguments.

La requête	L'application à lancer	Les arguments
ما حال الطقس في الجزائر	Météo (à Alger)	الجزائر
توقيت الشّروق في تمنراست	Météo (à Tamenrasset)	تمنراست

عندِي مقابلة مهمة غدا على الثامنة صباحا	Alarme (à 8 a.m.)	غدا، الثامنة، صباحا
كم حاصل قسمة 4 على 2	Calculatrice	قسمة، 4، 2

**Tableau 4.1** Exemple d'arguments tirés et d'applications lancées à partir des requêtes de niveau 1

Pour traiter ce type de commandes, nous avons fait un **étiquetage morphosyntaxique**, pour extraire les entités nommées présentes dans la commande (les lieux, les expressions temporelles, etc.). Nous avons utilisé à cet effet l'étiqueteur morpho-syntaxique de Stanford<sup>28</sup>.

Ces entités représentent les arguments de la requête, une fois ces derniers tirés, ils seront communiqués à l'application correspondante. Ce genre d'exécution sous Android nécessite d'avoir pour chaque application, une API qui lui permet de recevoir ces arguments.

Mais l'API n'est pas présente dans toutes les applications. Pour pallier ce problème (absence de l'API sur l'application souhaitée), nous avons redirigé les requêtes vers une recherche Web pour fournir à l'utilisateur les résultats souhaités.

#### 4.5.3 Générateur de réponses de niveau 2

Les requêtes peuvent aller de la plus simple à la plus compliquée, et ce, selon leur quantité d'ambiguïtés et ainsi, les requêtes peuvent ne pas être affectées aux deux niveaux précédents. De ce fait, nous avons développé un générateur de réponse de niveau 2, le niveau le plus élevé supportant n'importe quelle requête, quelle que soit la quantité d'ambiguïtés dans ces requêtes. Ce générateur est un générateur à base d'apprentissage. Dans le chapitre précédent nous avons dégagé l'architecture du modèle d'apprentissage (deep learning) de ce générateur. Dans ce chapitre, nous allons voir l'implémentation de ce modèle en exploitant les dernières bibliothèques dédiées au domaine. Nous allons ensuite discuter le processus d'apprentissage et évaluer le modèle.

Tout le processus d'apprentissage est exécuté sur l'environnement Colab, décrit précédemment. Tout d'abord nous commençons par l'installation des bibliothèques nécessaires pour l'exécution de notre code.

```
import tensorflow as tf
assert tf.__version__.startswith('2')
tf.random.set_seed(1234)

!pip install tensorflow-datasets==1.2.0
import tensorflow_datasets as tfds
```

<sup>28</sup> <https://nlp.stanford.edu/projects/arabic.shtml>

```
import os
import re
import numpy as np

import matplotlib.pyplot as plt
```

**Algorithme 4.4** Préparation de l'environnement pour l'apprentissage

Ensuite, nous avons préparé le corpus TALAA-AFAQ en éliminant la ponctuation et en gardant que les questions et leurs réponses. Un *Tokenizer* est ensuite construit à l'aide de la bibliothèque *Tensorflow Datasets*. Le Tokenizer :

- extrait chaque token unique dans les questions et les réponses présentes dans le corpus ;
- associe à chaque token un identifiant entier et unique, qui sera l'entrée correspondante à ce token dans le réseau profond ;
- ajoute deux tokens « START » et « END » pour marquer le début et la fin d'une entrée (question ou réponse).

```
# Build tokenizer using tfds for both questions and answers
tokenizer = tfds.features.text.SubwordTextEncoder.build_from_corpus(
    questions + answers, target_vocab_size=2**13)

# Define start and end token to indicate the start and end of a
sentence
START_TOKEN, END_TOKEN = [tokenizer.vocab_size],
[tokenizer.vocab_size + 1]

# Vocabulary size plus start and end token
VOCAB_SIZE = tokenizer.vocab_size + 2
```

**Algorithme 4.5** Construction du *Tokenizer*

Un *Padding* est ensuite exécuté sur les données pour assurer que toutes les entrées ont la même longueur de caractères (longueur maximale de 100 caractères par entrée). Après cette phase de préparation du corpus, nous avons eu 2001 paires de question-réponse, avec 6700 mots distincts.

Nous commençons maintenant la préparation de l'apprentissage, l'utilisation de l'API *tf.data.Dataset* nous a offerte des fonctionnalités comme le cache et la pré-lecture pour accélérer l'apprentissage.

Notre modèle d'apprentissage profond est un transformateur qui utilise la méthode *teacherforcing*. Cette méthode force le modèle à passer la sortie (la réponse dans notre cas) correcte à la prochaine étape de l'apprentissage au lieu de la réponse prédite.

Implémentons maintenant la fonction de calcul de l'attention (déjà définie dans le chapitre précédent), permettant au modèle de capturer le plus essentiel des entrées.

```
def scaled_dot_product_attention(query, key, value, mask):
    """Calculate the attention weights."""
    matmul_qk = tf.matmul(query, key, transpose_b=True)

    # scale matmul_qk
    depth = tf.cast(tf.shape(key)[-1], tf.float32)
    logits = matmul_qk / tf.math.sqrt(depth)

    # add the mask to zero out padding tokens
    if mask is not None:
        logits += (mask * -1e9)

    # softmax is normalized on the last axis (seq_len_k)
    attention_weights = tf.nn.softmax(logits, axis=-1)

    output = tf.matmul(attention_weights, value)

    return output
```

**Algorithme 4.6** Calcul de l'attention du produit scalaire

Définissons maintenant la couche d'attention qui va compléter la fonction précédente.

```
class MultiHeadAttention(tf.keras.layers.Layer):

    def __init__(self, d_model, num_heads,
name="multi_head_attention"):
        super(MultiHeadAttention, self).__init__(name=name)
        self.num_heads = num_heads
        self.d_model = d_model

        assert d_model % self.num_heads == 0

        self.depth = d_model // self.num_heads

        self.query_dense = tf.keras.layers.Dense(units=d_model)
        self.key_dense = tf.keras.layers.Dense(units=d_model)
        self.value_dense = tf.keras.layers.Dense(units=d_model)

        self.dense = tf.keras.layers.Dense(units=d_model)

    def split_heads(self, inputs, batch_size):
        inputs = tf.reshape(
            inputs, shape=(batch_size, -1, self.num_heads, self.depth))
        return tf.transpose(inputs, perm=[0, 2, 1, 3])
```

```
def call(self, inputs):
    query, key, value, mask = inputs['query'], inputs['key'],
inputs[
    'value'], inputs['mask']
batch_size = tf.shape(query)[0]

# linear layers
query = self.query_dense(query)
key = self.key_dense(key)
value = self.value_dense(value)

# split heads
query = self.split_heads(query, batch_size)
key = self.split_heads(key, batch_size)
value = self.split_heads(value, batch_size)

# scaled dot-product attention
scaled_attention = scaled_dot_product_attention(query, key,
value, mask)

scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1,
3])

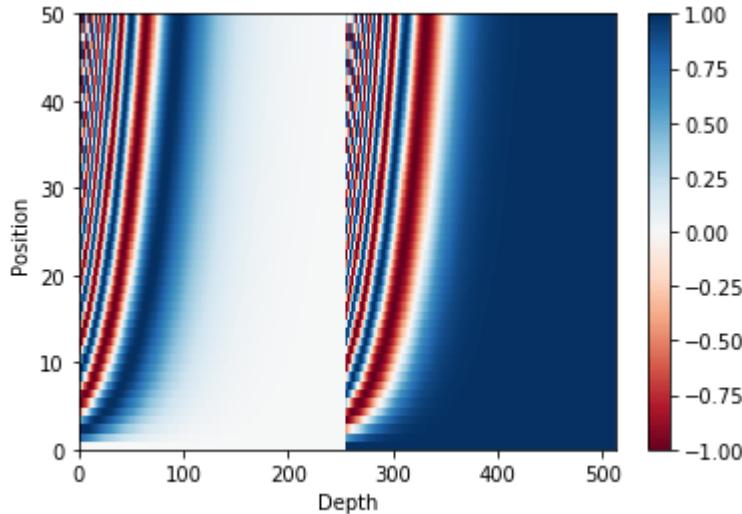
# concatenation of heads
concat_attention = tf.reshape(scaled_attention,
                                (batch_size, -1, self.d_model))

# final linear layer
outputs = self.dense(concat_attention)

return outputs
```

**Algorithme 4.7** Implémentation de la couche d'attention

Puisque notre modèle ne contient ni de couches récurrentes ni de convolutions, il nous est nécessaire d'encoder les positions des tokens pour permettre au modèle d'avoir l'information des positions de ces derniers (Figure 4.7).



**Figure 4.7** Exemple d'encodage des *tokens*

Comme expliqué dans la conception, le transformateur se base sur les couches d'encodeur et celles du décodeur. Le code qui suit montre leur implémentation.

```

def encoder(vocab_size,
            num_layers,
            units,
            d_model,
            num_heads,
            dropout,
            name="encoder"):

    inputs = tf.keras.Input(shape=(None,), name="inputs")
    padding_mask = tf.keras.Input(shape=(1, 1, None),
name="padding_mask")

    embeddings = tf.keras.layers.Embedding(vocab_size,
d_model)(inputs)
    embeddings *= tf.math.sqrt(tf.cast(d_model, tf.float32))
    embeddings = PositionalEncoding(vocab_size, d_model)(embeddings)

    outputs = tf.keras.layers.Dropout(rate=dropout)(embeddings)

    for i in range(num_layers):
        outputs = encoder_layer(
            units=units,
            d_model=d_model,
            num_heads=num_heads,
            dropout=dropout,
            name="encoder_layer_{}".format(i),
        )([outputs, padding_mask])

    return tf.keras.Model(

```

```

        inputs=[inputs, padding_mask], outputs=outputs, name=name)

def decoder(vocab_size,
            num_layers,
            units,
            d_model,
            num_heads,
            dropout,
            name='decoder'):
    inputs = tf.keras.Input(shape=(None,), name='inputs')
    enc_outputs = tf.keras.Input(shape=(None, d_model),
                                 name='encoder_outputs')
    look_ahead_mask = tf.keras.Input(
        shape=(1, None, None), name='look_ahead_mask')
    padding_mask = tf.keras.Input(shape=(1, 1, None),
                                  name='padding_mask')

    embeddings = tf.keras.layers.Embedding(vocab_size,
                                           d_model)(inputs)
    embeddings *= tf.math.sqrt(tf.cast(d_model, tf.float32))
    embeddings = PositionalEncoding(vocab_size, d_model)(embeddings)

    outputs = tf.keras.layers.Dropout(rate=dropout)(embeddings)

    for i in range(num_layers):
        outputs = decoder_layer(
            units=units,
            d_model=d_model,
            num_heads=num_heads,
            dropout=dropout,
            name='decoder_layer_{}'.format(i),
            )(inputs=[outputs, enc_outputs, look_ahead_mask, padding_mask])

    return tf.keras.Model(
        inputs=[inputs, enc_outputs, look_ahead_mask, padding_mask],
        outputs=outputs,
        name=name)

```

#### Algorithme 4.8 Implémentation des couches d'encodeur et de décodeur

Il ne nous reste maintenant que l'implémentation du transformateur et le lancement d'apprentissage.

```

def transformer(vocab_size,
                num_layers,
                units,
                d_model,
                num_heads,

```

```

        dropout,
        name="transformer"):
inputs = tf.keras.Input(shape=(None,), name="inputs")
dec_inputs = tf.keras.Input(shape=(None,), name="dec_inputs")

enc_padding_mask = tf.keras.layers.Lambda(
    create_padding_mask, output_shape=(1, 1, None),
    name='enc_padding_mask')(inputs)
# mask the future tokens for decoder inputs at the 1st attention
block
look_ahead_mask = tf.keras.layers.Lambda(
    create_look_ahead_mask,
    output_shape=(1, None, None),
    name='look_ahead_mask')(dec_inputs)
# mask the encoder outputs for the 2nd attention block
dec_padding_mask = tf.keras.layers.Lambda(
    create_padding_mask, output_shape=(1, 1, None),
    name='dec_padding_mask')(inputs)

enc_outputs = encoder(
    vocab_size=vocab_size,
    num_layers=num_layers,
    units=units,
    d_model=d_model,
    num_heads=num_heads,
    dropout=dropout,
)(inputs=[inputs, enc_padding_mask])

dec_outputs = decoder(
    vocab_size=vocab_size,
    num_layers=num_layers,
    units=units,
    d_model=d_model,
    num_heads=num_heads,
    dropout=dropout,
)(inputs=[dec_inputs, enc_outputs, look_ahead_mask,
dec_padding_mask])

outputs = tf.keras.layers.Dense(units=vocab_size,
name="outputs")(dec_outputs)

return tf.keras.Model(inputs=[inputs, dec_inputs],
outputs=outputs, name=name)

```

#### **Algorithm 4.9** Implémentation du transformateur

Nous définissons la fonction *Loss* et la fonction du taux d'apprentissage personnalisé évoluant avec le pas d'apprentissage (Figure 4.8).

```

def loss_function(y_true, y_pred):
    y_true = tf.reshape(y_true, shape=(-1, MAX_LENGTH - 1))

    loss = tf.keras.losses.SparseCategoricalCrossentropy(
        from_logits=True, reduction='none')(y_true, y_pred)

    mask = tf.cast(tf.not_equal(y_true, 0), tf.float32)
    loss = tf.multiply(loss, mask)

    return tf.reduce_mean(loss)

```

**Algorithme 4.10** Implémentation de la fonction Loss

```

class CustomSchedule(tf.keras.optimizers.schedules.LearningRateSchedule):

    def __init__(self, d_model, warmup_steps=4000):
        super(CustomSchedule, self).__init__()

        self.d_model = d_model
        self.d_model = tf.cast(self.d_model, tf.float32)

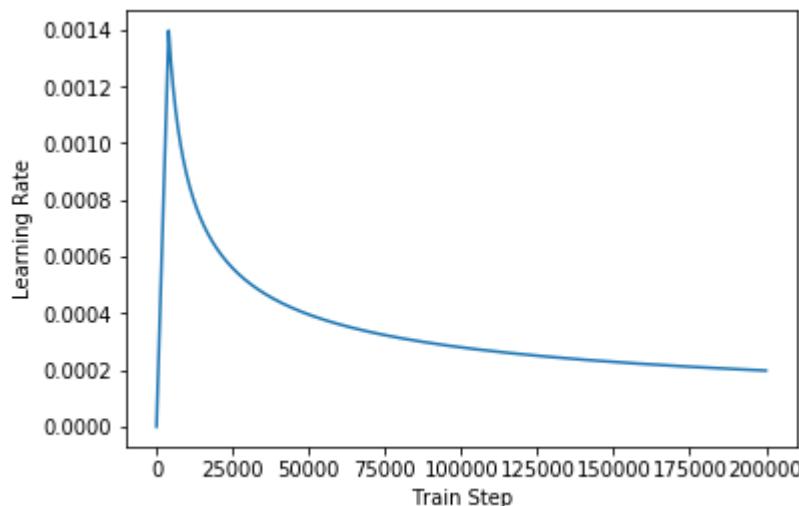
        self.warmup_steps = warmup_steps

    def __call__(self, step):
        arg1 = tf.math.rsqrt(step)
        arg2 = step * (self.warmup_steps**-1.5)

        return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)

```

**Algorithme 4.11** Implémentation de la fonction du taux d'apprentissage personnalisé



**Figure 4.8** variation du taux d'apprentissage en fonction du pas d'apprentissage

Nous lançons maintenant l'apprentissage après avoir compilé le modèle.

```
tf.keras.backend.clear_session()

# Hyper-parameters
NUM_LAYERS = 2
D_MODEL = 256
NUM_HEADS = 8
UNITS = 512
DROPOUT = 0.1

model = transformer(
    vocab_size=VOCAB_SIZE,
    num_layers=NUM_LAYERS,
    units=UNITS,
    d_model=D_MODEL,
    num_heads=NUM_HEADS,
    dropout=DROPOUT)

EPOCHS = 20

model.fit(dataset, epochs=EPOCHS)
```

#### Algorithm 4.12 Lancement de l'apprentissage

Puisque nous avons utilisé Colab avec GPU, le temps d'apprentissage est réduit. En effet, la moyenne du temps d'exécution d'une itération (epoch) est d'une minute, ce qui nous a permis d'analyser et de tester plusieurs architectures des transformateurs.

##### 4.5.3.1 Expérimentations et évaluation du modèle d'apprentissage

Nous avons décrit précédemment la conception, l'architecture et l'implémentation de notre modèle d'apprentissage. Bien que nous ayons utilisé un transformateur pour la génération de réponses qui impose quelques paramètres (comme les couches d'encodeur et de décodeur), mais il reste encore quelques paramètres à ajuster et c'est au développeur de trouver comment bien les régler afin d'obtenir les meilleurs résultats pour son modèle.

Nous avons testé différentes architectures en variant à chaque fois un paramètre puis nous évaluons les résultats. Les paramètres que nous avons variés sont les suivants.

- **Le nombre de couches (Layers)** : compte tenu de la complexité de notre modèle, nous nous sommes limités à deux couches. En effet, chaque ajout d'une couche duplique les

3 composants de notre modèle (l'encodeur, le décodeur et le transformateur), ce qui augmente d'une façon exponentielle le temps d'apprentissage.

- **Le nombre de têtes (Heads)** : c'est le nombre de têtes encodeurs et décodeurs dans chaque couche.
- **Les unités (Unités)** : il s'agit du nombre de neurones du transformateur.
- **Le dropout** : le dropout est la probabilité d'ignorance de certains neurones aléatoires dans une couche à une itération donnée (epoch) de l'apprentissage. Cette ignorance prévient le développement d'une forte dépendance entre les neurones, ce qui entraîne un sur-apprentissage. Notons que ce paramètre doit être modifié avec précaution, en dépassant un certain seuil. En revanche, il peut prévenir le modèle à converger vers le minimum.

### 1) Données expérimentales

Nous avons varié à chaque fois un paramètre (cités plus haut), puis lancé l'apprentissage. En effet, nous avons divisé les données du corpus TALAA-AFAQ en trois, 70% des données sont consacrées à l'apprentissage, 20% pour le test et 10% pour la validation. Nous avons ainsi fixé le nombre d'itérations (epochs) à 20.

### 2) Métriques d'évaluation

Les métriques utilisées pour évaluer notre modèle d'apprentissage sont les suivantes.

- **La Précision (Accuracy)** : c'est le nombre des mots prédits correctement (mot correct et position correcte) par rapport au nombre total des mots prédits.
- **La perte (Loss)** : c'est une mesure d'erreur, elle représente le rapport entre les mots prédits avec erreur (faux mots ou fausse position) avec les mots qui doivent être prédits (la séquence but).

### 3) Résultats et discussion

Dans ce qui suit, nous présenterons les résultats de l'apprentissage, qui ont été obtenus grâce à l'outil *Tensorboard*, qui permet la visualisation des différents paramètres d'apprentissage avec Tensorflow.

Nombre de couches	Nombre de têtes	Nombre d'unités	Dropout	Précision	Perte

1	2	128	0	0.269	0.956
1	2	256	0	0.285	0.924
1	2	512	0	0.346	0.895
1	4	128	0	0.456	0.811
1	4	256	0	0.562	0.922
1	4	512	0	0.582	0.823
1	8	128	0	0.438	0.836
1	8	256	0	0.429	0.755
1	8	512	0	0.411	0.752

**Tableau 4.2** Résultats d'apprentissage avec une seule couche de transformateur

Les résultats obtenus avec une seule couche de transformateur ne sont pas assez satisfaisants, car comme nous pouvons l'observer, nous n'avons pas pu dépasser les 58% de précision. Maintenant, gardons la même architecture précédente et modifions uniquement le dropout et observons à nouveau les résultats.

Nombre de couches	Nombre de têtes	Nombre d'unités	Dropout	Précision	Perte
1	2	128	0.1	0.322	0.927
1	2	256	0.1	0.371	0.919
1	2	512	0.1	0.344	0.885
1	4	128	0.1	0.382	0.872
1	4	256	0.1	0.408	0.892
1	4	512	0.1	0.428	0.812
1	8	128	0.1	0.519	0.732
1	8	256	0.1	0.559	0.751
1	8	512	0.1	0.571	0.728

**Tableau 4.3** Résultats d'apprentissage avec une seule couche de transformateur et un taux de dropout de 0.1

Avec l'utilisation du dropout, les résultats sont légèrement améliorés, mais restent toujours pas assez satisfaisants. Nous pouvons clairement remarquer que la précision reste petite avec un

taux d'erreur considérable. Maintenant, augmentons le nombre de couches du transformateur et observerons une fois de plus les résultats.

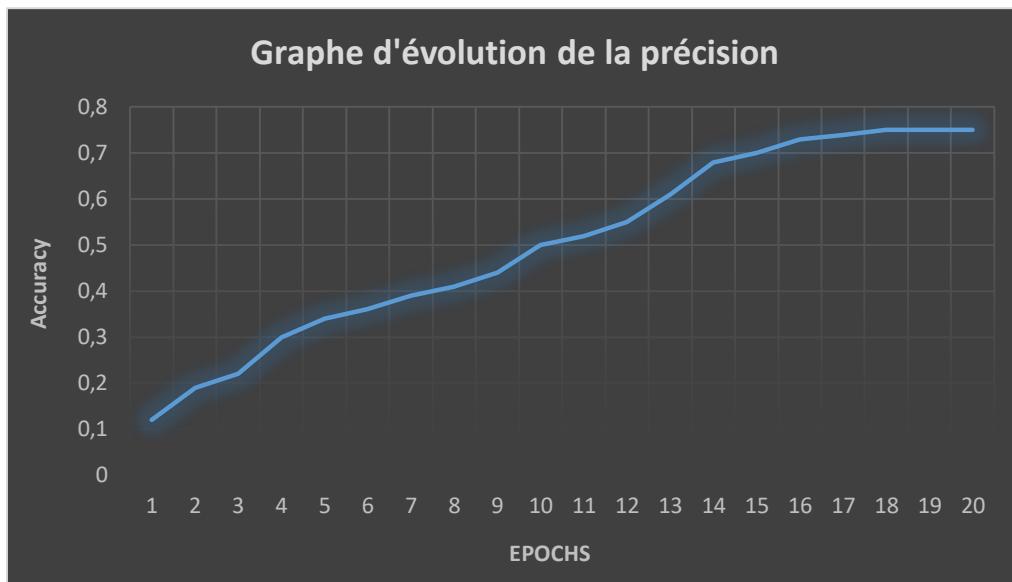
Nombre de couches	Nombre de têtes	Nombre d'unités	Dropout	Précision	Perte
2	2	128	0.1	0.513	0.515
2	2	256	0.1	0.584	0.476
2	2	512	0.1	0.652	0.414
2	4	128	0.1	0.68	0.403
2	4	256	0.1	0.713	0.39
2	4	512	0.1	0.722	0.361
2	8	128	0.1	0.736	0.355
2	8	256	0.1	0.739	0.348
2	8	512	0.1	0.751	0.342

**Tableau 4.4** Résultats d'apprentissage avec deux couches de transformateur et un taux de dropout de 0.1

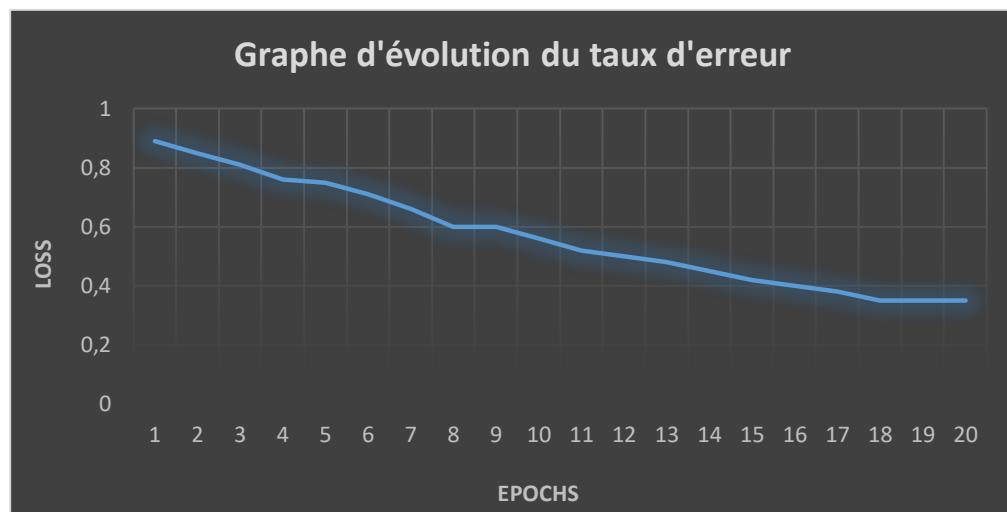
Les meilleurs résultats ont été obtenus grâce à la dernière architecture ayant 2 couches d'apprentissage, 8 têtes d'encodeur et de décodeur, 512 unités et un taux de dropout de 0.1. La précision a atteint 75% avec un taux d'erreur de 34%.

En effet, le taux d'erreur reste élevé étant donné que le modèle n'arrive pas à se généraliser parfaitement sur de nouvelles instances de tests. Ceci est dû au manque de données d'apprentissage (la taille du corpus est petite). Généralement, les transformateurs nécessitent beaucoup plus de données pour donner de meilleurs résultats, mais malheureusement, le manque de tels corpus reste un obstacle à l'obtention des résultats souhaités dans les travaux informatiques portants sur la langue arabe.

Les courbes d'apprentissages (graphes d'évolution de la précision et du taux d'erreur) du meilleur modèle obtenu sont représentées dans les figures ci-dessous.



**Figure 4.9** Graphe de variation de la précision du modèle



**Figure 4.10** Graphe de variation du taux d'erreur du modèle

Les réponses générées par le module de génération de réponses, vont être envoyées au prochain module pour les communiquer enfin à l'utilisateur.

## 4.6 Module de synthèse vocale

Ce module représente la fin du cycle du traitement de la requête de l'utilisateur, il est évident que la réponse générée pendant l'étape précédente doit être retournée à l'utilisateur. En revanche, la meilleure façon pour retourner la réponse à l'utilisateur est la même façon dont il a envoyé sa requête, i.e. via une voix, mais une voix synthétique (simulée par le système). Cette communication vocale entre l'utilisateur et l'application familiarise l'utilisateur avec l'assistant, ce qui rapproche l'expérience d'un processus réel entre humains.

Le processus de synthèse vocale commence d'abord par convertir la réponse textuelle en audio. Pour ce faire, le module de synthèse vocale prend comme entrée un texte (réponse) et génère la transcription vocale de ce dernier.

Le service *Google Cloud* offre une API sous Android pour faire la synthèse vocale, nous avons choisi donc d'utiliser cette API puisque elle est parfaitement intégrable avec Android, ainsi que pour garder le même service (*Google Cloud*) utilisé dans la reconnaissance vocale.

En effet, cette conversion en audio est faite en initialisant un objet **TextToSpeech** et redéfinir la fonction *onInit*.

```
mTTS = new TextToSpeech(this, new TextToSpeech.OnInitListner(){
    @Override
    public void onInit(int status){
        if(status == TextToSpeech.SUCCESS){
            int result = mTTS.setLanguage(Locale.Arabic);
            if (result == TextToSpeech.LANG_MISSING_DATA || result
== TextToSpeech.LANG_NOT_SUPPORTED){
                Log.e("TTS", "Language Not Supported")
            }
        }
    }
});
```

**Algorithme 4.13** Lancement du service *Text To Speech* de *Google Cloud*

Cette fonction doit ensuite être associée à un événement (dans notre cas l'évènement est l'arrivée d'une réponse). Donc elle va attendre l'arrivée d'une réponse à partir du serveur pour la transmettre en voix synthétique afin de la communiquer à l'utilisateur.

Notons que ce service doit avoir un accès à Internet pour qu'il puisse s'exécuter correctement.

## 4.7 Application Fassiha

Nous avons décrit dans le chapitre précédent l'architecture, les modules et les niveaux de traitement de notre application, mais tous ces derniers doivent être liés pour assurer une meilleure coordination dans le but d'atteindre les résultats attendus.

L'application Fassiha est conçue d'une façon très simple afin que l'utilisateur comprenne tout de suite comment l'exploiter. En effet, elle est composée de.

- Back-end : englobant tous les modules de traitement de la requête.
- Front-end : représentant l'interface graphique à travers laquelle l'utilisateur interagit avec notre système (envoi de requêtes et réception de réponses).

Notre application fonctionne en mode « client-serveur », où le client est représenté par l'appareil et le serveur est représenté par le back-end qui gère les relations entre les appareils. Dans ce qui suit, nous allons décrire l'implémentation du back-end et du front-end.

#### 4.7.1 Le système intelligent d'arrière-plan (Back-end)

Le back-end est la partie interne du système, elle est abstraite pour l'utilisateur. En effet, notre architecture permet de séparer parfaitement ce fonctionnement interne de l'interface.

Pour implémenter le back-end, nous avons utilisé le framework python « Django REST Framework », qui nous a permis de construire une API REST pour notre application.

**API REST** (*Representational State Transfer Application Programming Interface*) est une architecture pour la création des services web. Notre application comporte donc un service web permettant au consommateur du service (dans notre cas l'application mobile) d'implémenter des méthodes standards pour l'envoi et la réception des requêtes (méthodes comme POST et GET).

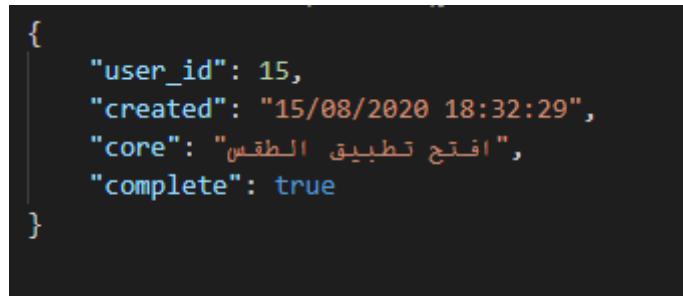
En effet, notre choix a été penché sur « Django REST framework », parce que tous les modules du back-end sont codés en python, de plus, ce framework offre un ensemble de fonctionnalités par défaut, qui facilite l'implémentation des APIs, dont la majeure fonctionnalité est la gestion automatique de la sécurité de l'API. Ce dernier aspect est très important car notre application fonctionne sous Internet et a besoin d'être sécurisée.

Pour la communication entre front-end et back-end, notre API utilise le format JSON, qui permet le bon passage des requêtes et réponses formalisées et structurées. En effet, les requêtes et les réponses sont les deux éléments majeurs qui vont être fournis par l'API, c'est pour cela que nous avons implémenté deux modèles Django spécifiques pour ces deux derniers avec des méthodes permettent leur consommation.

Quant à la communication entre les modules internes du back-end, cela se fait indépendamment via des lectures et écritures d'une mini base de données interne gérée par SQLite. Chaque module au début de son traitement, lit son entrée de la base et réécrit à la fin le résultat dans la même base, pour que le prochain module puisse prendre la main.

Résumons les flux majeurs de communication entre le front-end et le back-end de notre application :

- l'utilisateur communique sa requête vocalement via le front-end ;
- le module de reconnaissance vocale reçoit cette requête, la convertit en texte et la renvoie au front-end ;
- le front-end structure cette requête sous un format JSON et l'envoie au back-end en utilisant la méthode HTTP POST ;



```
{  
    "user_id": 15,  
    "created": "15/08/2020 18:32:29",  
    "core": "افتح تطبيق الملايين",  
    "complete": true  
}
```

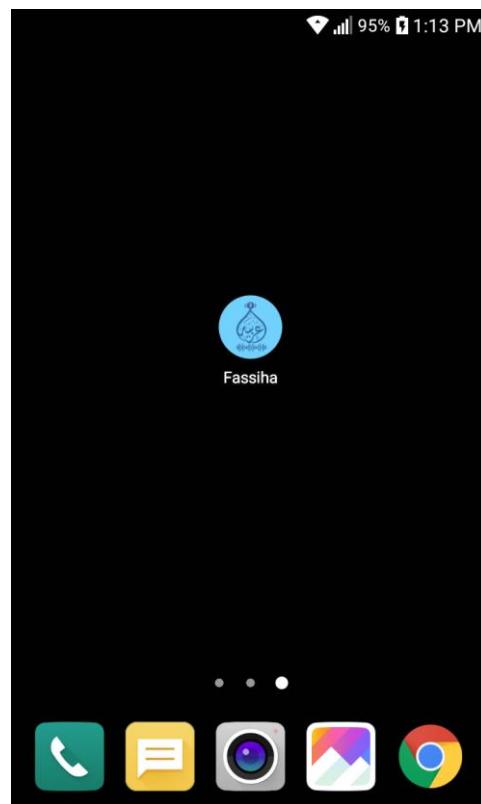
**Figure 4.11** Requête sous format JSON

- l'API convertit la requête sous format JSON en un objet python prêt pour le traitement ;
- le traitement se lance, tels que chaque module lit et écrit les résultats dans la base de données interne ;
- une fois la réponse est prête, elle sera reconvertie en format JSON et communiquée au front-end via la méthode GET ;
- le front-end reçoit alors la réponse et la retourne à l'utilisateur vocalement et graphiquement (affichage des résultats).

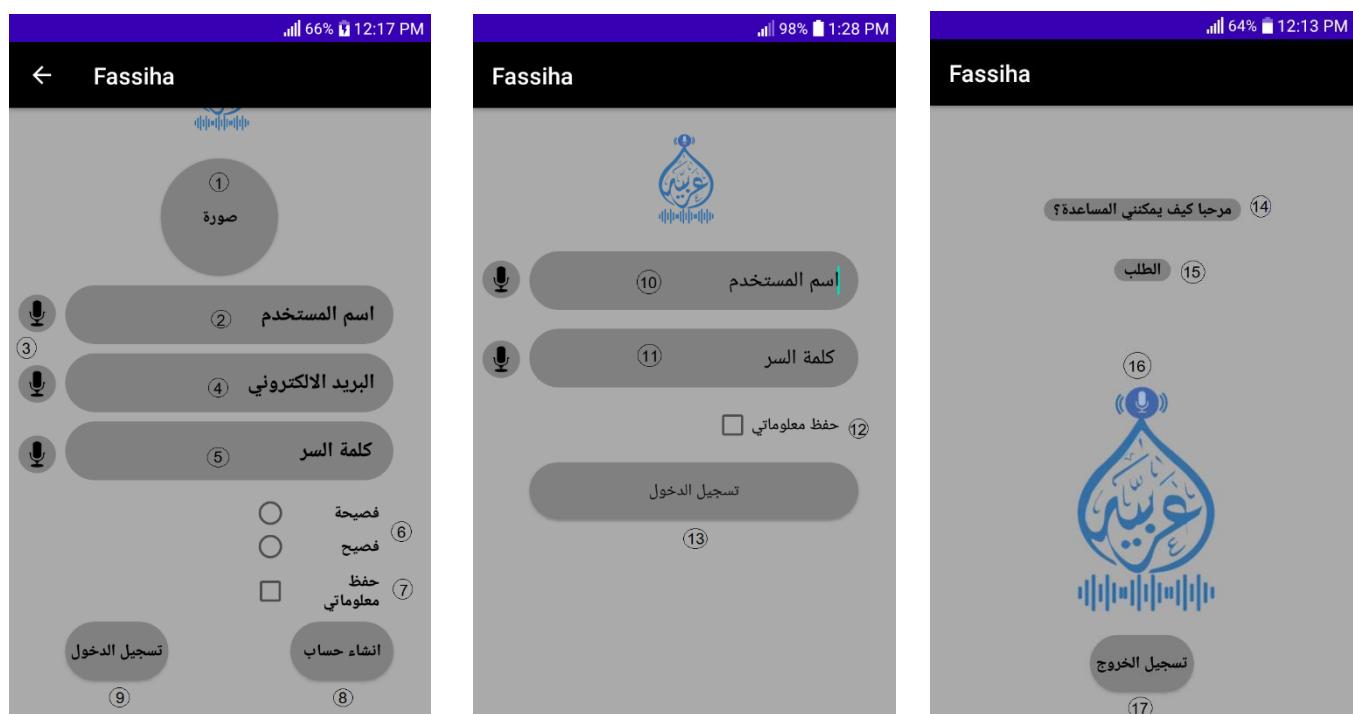
#### 4.7.2 L'interface utilisateur (Front-end)

Le front-end est l'interface de notre application. En effet, un assistant vocal est conçu pour faciliter la vie quotidienne des utilisateurs. Donc il doit se présenter dans la plus simple des formes. De ce fait, nous nous sommes basés sur la simplicité dans la conception et la réalisation de l'interface de notre application.

Notre interface est dédiée au système Android. Ce dernier utilise la notion "d'Activités" pour la construction de l'interface. Chaque activité présente une page contenant des éléments (bouton, champ de saisie, etc.). Dans ce qui suit, nous allons présenter les activités de notre interface en décrivant leurs différents éléments.



**Figure 4.12** Lanceur de l'application Fassiha



**Figure 4.13** Activités de l'application Fassiha

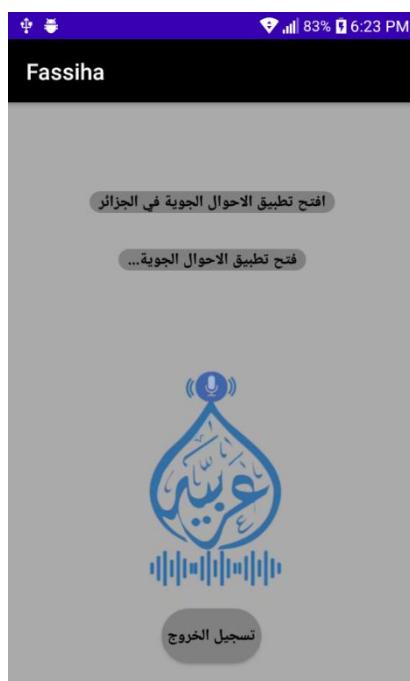
<b>Page (Activité)</b>	<b>Fonction de la page</b>	<b>Élément</b>	<b>Fonction de l'élément</b>
<b>1.</b> <b>Page d'inscription</b>	La première activité est lancée lors de la toute première utilisation de Fassiha, c'est une activité pour s'inscrire (créer un compte).	1	Bouton pour sélectionner une image pour le profil de l'utilisateur (champs facultatif).
		2	Champ pour saisir le nom de l'utilisateur (l'identifiant).
		3	Bouton microphone, pour remplir les différents champs vocalement.
		4	Champ pour saisir l'adresse électronique de l'utilisateur.
		5	Champ pour saisir le mot de passe de l'utilisateur.
		6	Bouton radio pour choisir une voix pour l'assistant (féminin/masculin)
		7	Case à cocher pour sauvegarder les informations de l'utilisateur (pour éviter la resaisie à chaque connexion).
		8	Bouton pour créer le compte de l'utilisateur avec les informations précédentes.
		9	Bouton d'accès à la page d'accueil de l'assistant vocal Fassiha
<b>2.</b> <b>Page d'authentification</b>	Une fois inscrit, cette activité permet à l'utilisateur de s'authentifier (prouver son identité) pour accéder à son compte Fassiha.	10	Champ pour saisir le nom de l'utilisateur
		11	Champ pour saisir le mot de passe de l'utilisateur.
		12	Case à cocher pour sauvegarder les informations de l'utilisateur (pour éviter la resaisie à chaque connexion).
		13	Bouton d'accès à la page d'accueil de l'assistant vocal Fassiha.

<b>3. Page d'accueil</b>	Une fois authentifié, cette activité permet à l'utilisateur de lancer des requêtes et de recevoir des réponses.	14	Champ d'affichage du message d'accueil, qui sera remplacé par le texte correspondant à la requête vocale de l'utilisateur.
		15	Champ d'affichage de la réponse correspondante à la requête de l'utilisateur.
		16	Logo interactif de Fassiha, en appuyant dessus, le système commence à capturer la requête vocale de l'utilisateur (capture audio). La capture audio s'arrête automatiquement lorsque l'utilisateur cesse de parler.
		17	Bouton pour se déconnecter du compte (sortir de l'application)

**Tableau 4.5** Guide d'utilisation de l'application Fassiha

Nous allons maintenant lancer un exemple de chaque niveau de requête, puis nous affichons la réponse générée par notre système.

#### Requête de niveau 0 (avec intention explicite)



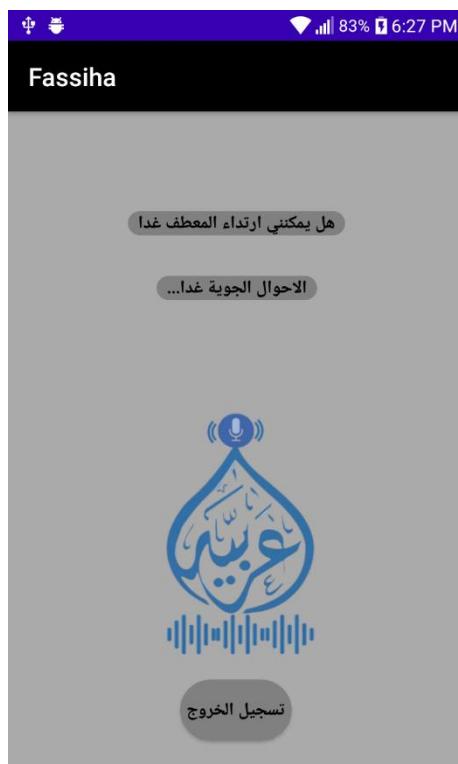
**Figure 4.14** Requête de niveau 0 et sa réponse



**Figure 4.15** Lancement de l'application Météo (à Alger)

L'application Fassiha lance l'application météo installée par défaut sur le smartphone, en tenant compte de l'argument « Alger ».

### Requête de niveau 1 (avec intention implicite)



**Figure 4.16** Requête de niveau 1 et sa réponse

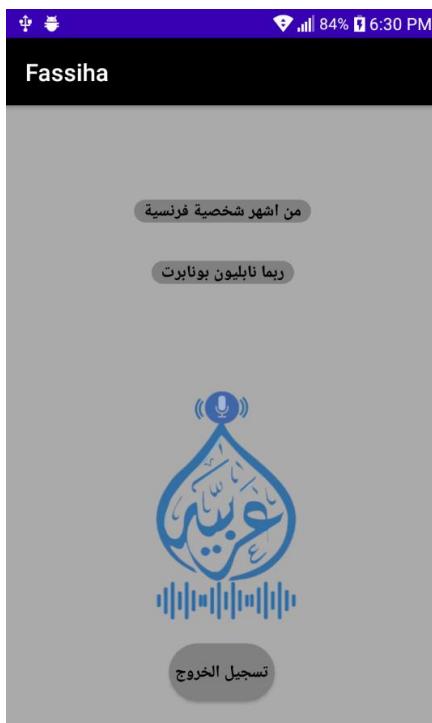


**Figure 4.17** Lancement de l'application Météo (météo de demain)

L'application Fassiha lance l'application météo installée par défaut sur le smartphone, en tenant compte de l'argument « demain ».

### Requête de niveau 2

Il faut noter que le niveau 2 n'exécute aucune application, comme expliqué auparavant. Ce niveau est conçu pour les commandes qui simulent une discussions humaine.



**Figure 4.18** Requête de niveau 2 et sa réponse

### 4.7.3 Expérimentations et évaluation générale de Fassiha

Nous avons précédemment évalué deux modules de notre application (le module de reconnaissance vocale et le module de génération de réponses pour les requêtes du niveau 2). En effet, l'évaluation de ces deux modules était évidente étant donné qu'ils se basent sur l'apprentissage automatique. Mais le reste des modules (le module de TALN, les classifieurs et les autres générateurs de réponses) est fortement attaché au fonctionnement général de l'application. De ce fait, nous estimons qu'une évaluation générale pour Fassiha est nécessaire pour déterminer sa performance.

#### 4.7.3.1 Données expérimentales (ensemble de test)

Nous avons construit un ensemble de test contenant 200 requêtes, 35% des requêtes (70 requêtes) sont des requêtes de **niveau 0**, 35% sont de **niveau 1** (70 requêtes) et 30% sont de **niveau 2** (60 requêtes). Chaque requête est stockée dans un fichier texte ayant un identifiant unique. Pour faciliter l'évaluation, les requêtes du même niveau sont ainsi regroupées dans un dossier ayant les noms 0, 1 ou 2 selon les requêtes qu'il contient.

#### 4.7.3.2 Métriques d'évaluation

Un script python est lancé pour l'évaluation. Il lit les requêtes à partir de l'ensemble de test

construit. En effet, nous nous intéressons à la classe/niveau (0, 1 ou 2) affectée aux requêtes par le système. Dans cette section, nous définissons les bases des critères que nous avons utilisés pour évaluer Fassiha.

1 - Nous faisons une sorte de classification en calculant le nombre de requêtes pour 4 classes distinctes (VP, VN, FP, FN).

- **VP (Vrai Positif)** : requête affectée à la classe  $i$  et elle appartient à la classe  $i$ .
- **VN (Vrai Négatif)** : requête n'est pas affectée à la classe  $i$  et elle n'appartient pas à la classe  $i$ .
- **FP (Faux Positif)** : requête n'est pas affectée à la classe  $i$  et elle appartient à la classe  $i$ .
- **FN (Faux Négatif)** : requête affectée à la classe  $i$  et elle n'appartient pas à la classe  $i$ .

2 - Nous calculons ensuite les mesures d'évaluation suivantes.

- **Précision (P)** : il s'agit d'une métrique classique qui évalue à quel point le modèle est bon et fiable pour prédire les classes.

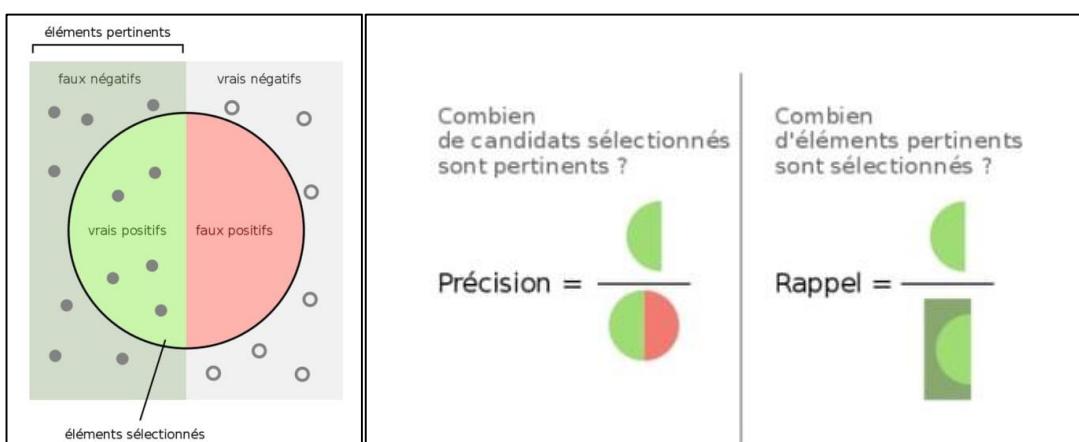
$$P = \frac{VP}{VP+FP}$$

- **Rappel (R)** : il s'agit d'une métrique qui évalue la capacité du modèle à classifier les requêtes correctement par rapport à tout l'ensemble de test.

$$R = \frac{VP}{VP+FN}$$

- **F-Mesure** : c'est une mesure qui combine la précision et le rappel. Elle essaye de donner un score plus global au modèle de classification, en prenant compte des résultats du rappel et de la précision.

$$F - \text{Mesure} = \frac{2*R*P}{R+P}$$



**Figure 4.19** VP, VN, FP, FN, rappel et précision

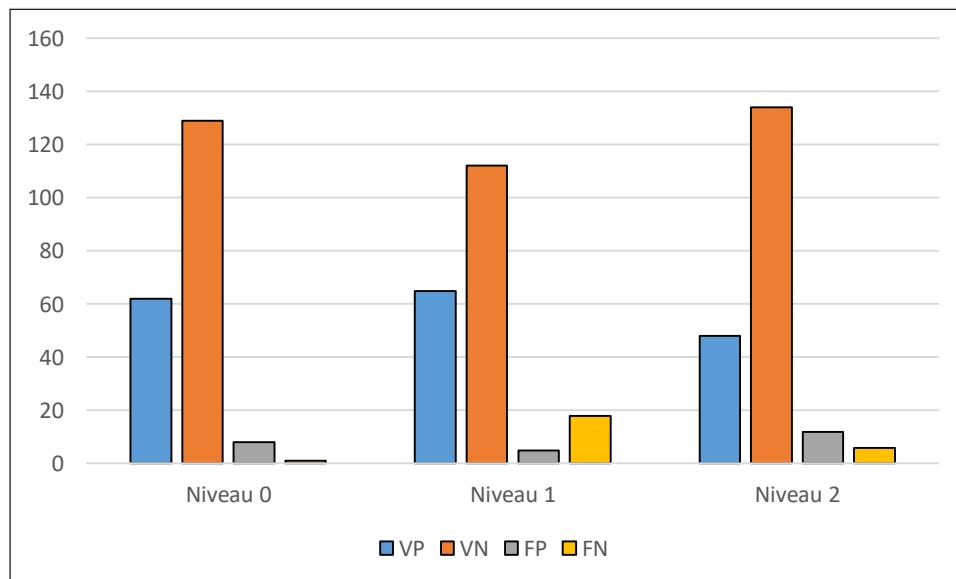
#### 4.7.3.3 Résultats et discussion

Nous exposons dans ce qui suit, les résultats obtenus sur notre ensemble de test (200 requêtes). Il faut noter que notre ensemble de test contient exclusivement les requêtes relatives aux applications Météo et Alarme. Il faut noter aussi que les résultats obtenus ne sont pas considérés comme des références).

Calculons les VP, VN, FP et FN pour chaque niveau de requête.

	<b>Niveau 0</b>	<b>Niveau 1</b>	<b>Niveau 2</b>	
<b>VP</b>	62	65	48	<b>175</b>
<b>VN</b>	129	112	134	375
<b>FP</b>	8	5	12	25
<b>FN</b>	1	18	6	25
	200	200	200	

**Tableau 4.6** Tableau montrant la classification des requêtes (VP, VN, FP, FN) en fonction des niveaux (0, 1, 2)

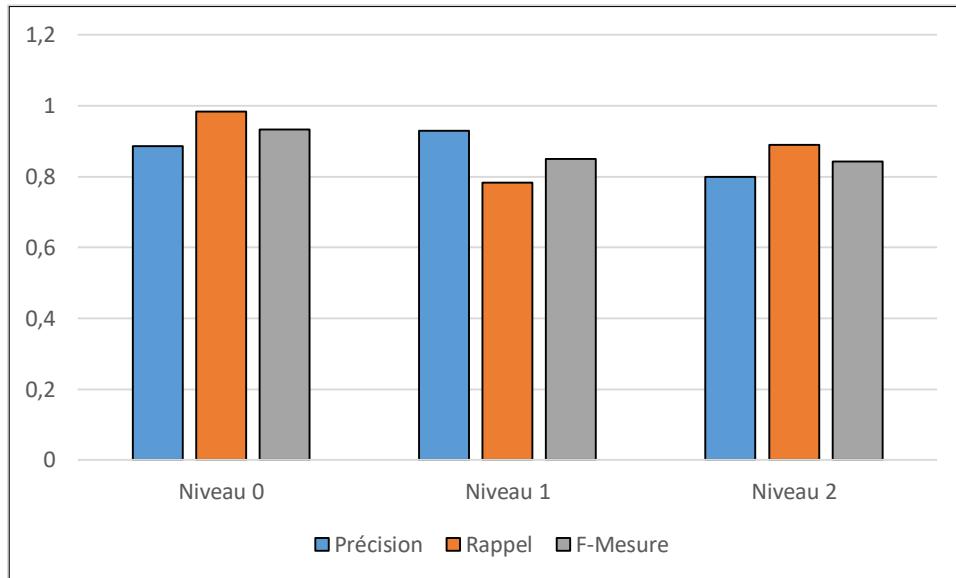


**Figure 4.20** Histogramme montrant la classification des requêtes (VP, VN, FP, FN) en fonction des niveaux (0, 1, 2)

Calculons maintenant la précision, le rappel et la F-mesure pour chaque niveau de requête en se basant sur les résultats présentés dans le tableau précédent.

	Niveau 0	Niveau 1	Niveau 2
Précision	0.886	0.929	0.8
Rappel	0.984	0.783	0.889
F-Mesure	0.932	0.849	0.842

**Tableau 4.7** Tableau montrant les valeurs de la précision, du rappel et de la F-Mesure en fonction des niveaux (0, 1, 2)



**Figure 4.21** Histogramme montrant les valeurs de la précision, du rappel et de la F-Mesure en fonction des niveaux (0, 1, 2)

Pour évaluer les performances d'un modèle de façon complète, nous devons analyser **à la fois** la précision et le rappel, y compris la F-Mesure. En effet, la précision et le rappel sont fréquemment en tension. Ceci est dû au fait que l'amélioration de la précision se fait généralement au détriment du rappel et réciproquement. Il faut noter ainsi que, plus les trois métriques citées se rapprochent de 1, plus le modèle est bon et performant.

Maintenant interprétons les résultats que nous avons obtenus.

- Pour le niveau 0, le rappel est très satisfaisant, la précision est un peu plus faible que le rappel, mais elle reste quand même très satisfaisante. En effet, uniquement 8 requêtes sur un total de 200 requêtes ont été mal classées pour ce niveau. Ceci est dû à la sensibilité du classifieur à base de règles (toute requête légèrement hors règles est

rejetée). Ainsi, la F-Mesure assure la fiabilité du classifieur de ce niveau étant donné que c'est la moyenne harmonique<sup>29</sup> de la précision et du rappel.

- Contrairement au niveau 0, le niveau 1 a plus de précision que de rappel. Le rappel est affecté par les faux négatifs (18 requêtes de niveau 0 et 2 ont été affectées au niveau 1). Néanmoins, la précision a atteint une valeur considérable et remarquable pour ce niveau. La valeur de la F-Mesure montre clairement que ce niveau est performant mais un peu moins que le niveau précédent. Ceci est dû à la tolérance de ce niveau (comme dit précédemment, toute requête ayant un score de similarité supérieur à 0.25 est acceptée comme requête de niveau 1).
- Bien que les résultats du niveau 2 soient satisfaisants (d'après la précision et le rappel), ils restent quand même plus faibles que les deux niveaux précédents. Nous pouvons le déduire en comparant par exemple la valeur de la F-Mesure aux précédentes (étant donné que cette dernière est la moyenne harmonique du rappel et de la précision). Cette diminution de valeurs est due à la complexité des requêtes de ce niveau.

La classification est bonne à 88%, c'est un bon score, mais qui laisse place à de futures améliorations, notamment pour le niveau 2.

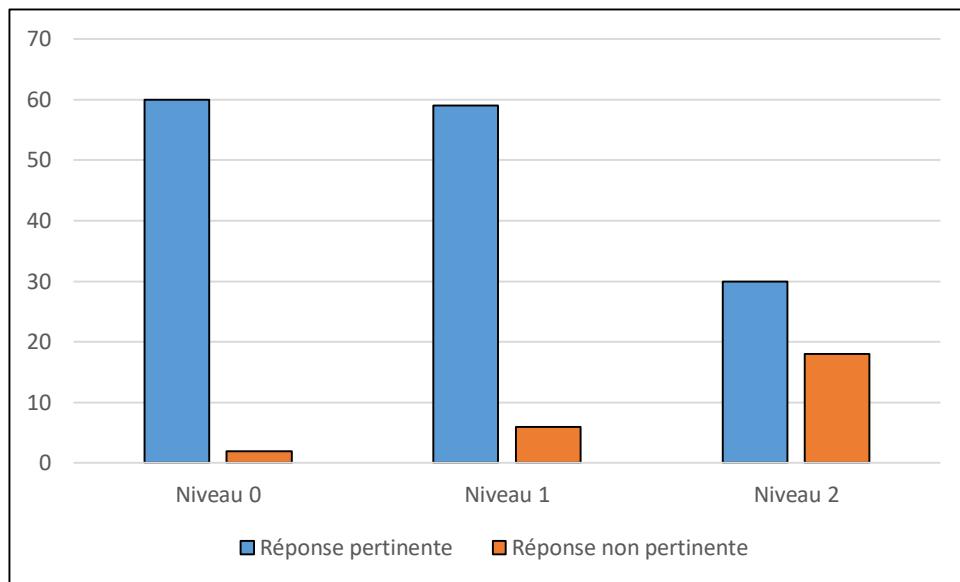
En effet, une requête bien classée, n'aura pas forcément une réponse appropriée. Pour cela, une évaluation des réponses générées par le système a été nécessaire pour savoir à quel point ce dernier est bon.

Maintenant, évaluons la génération de réponses en prenant les 175 requêtes bien classées (les vrais positifs) et calculer pour chaque niveau de requêtes, le nombre de réponses pertinentes générées.

	Niveau 0	Niveau 1	Niveau 2	
Réponse pertinente	60	59	30	149
Réponse non pertinente	2	6	18	26
	62	65	48	175

**Tableau 4.8** Tableau montrant le nombre de réponses pertinentes et non pertinentes en fonction des niveaux (0, 1, 2)

<sup>29</sup> En mathématiques, la moyenne harmonique (ou moyenne subcontraire) est l'un des nombreux types de moyenne et en particulier l'un des moyens de Pythagore. En règle générale, il convient aux situations où la moyenne des taux est souhaitée. Elle peut être exprimée comme l'inverse de la moyenne arithmétique des inverses de l'ensemble donné d'observations.



**Figure 4.22** Histogramme montrant le nombre de réponses pertinentes et non pertinentes en fonction des niveaux (0, 1, 2)

Calculons maintenant pour chaque générateur de réponse (niveau 0, 1 et 2), la précision avec la formule suivante :

$$\frac{Nb_{réponses\_pertinentes}}{Nb_{réponses\_pertinentes} + Nb_{réponses\_non\_pertinentes}}$$

**Équation 4.2** Formule de calcul de la précision

- Pour le **niveau 0** ; 60 réponses parmi 62 étaient correctement générées avec une précision considérable de 0.97.
- Pour le **niveau 1** ; 59 réponses parmi 65 étaient correctement générées avec une précision de 0.91 (inférieure par rapport au niveau précédent), bien que les requêtes aient bien été classées. Cela est dû à la difficulté d'extraction des arguments parfois pour ce niveau.
- Par dépendance totale du modèle de deep learning dont nous avons parlé précédemment, le générateur de réponses de **niveau 2** a généré seulement 30 réponses correctes parmi 48, avec une précision de 0.63. La génération de réponses pour ce niveau est vachement moins performante par rapport aux autres niveaux, mais une légère amélioration du corpus arabe va certainement améliorer la performance de ce niveau.

La génération de réponses est bonne à 85%, c'est un bon score, mais qui laisse place à de futures améliorations, notamment pour le niveau 2.

Avant de conclure ce chapitre, nous rappelons que nous étions assez satisfaits des résultats obtenus. Cependant, notre application est perfectible.

## 4.8 Conclusion

Ce chapitre a représenté le dernier volet de notre mémoire, il a validé l'étape de conception et a donné une vision globale de notre travail réalisé, en présentant les différents domaines que nous avons explorés, ainsi que les résultats que nous avons obtenus.

# CONCLUSION GÉNÉRALE ET PERSPECTIVES

Nous avons présenté dans ce mémoire la solution apportée pour la construction d'un assistant vocal virtuel personnalisé qui supporte la langue arabe, qui consiste en un système composé d'un sous-système interactif (front-end), à travers lequel l'utilisateur interagit avec son smartphone (interaction homme-machine) par le biais des directives vocales et d'un sous-système intelligent (front-back), représentant le cerveau du système. Ce dernier traite d'une manière intelligente les directives vocales provenant des utilisateurs, et ce, en utilisant les techniques du TALN, pour générer des réponses appropriées à ces directives vocales.

Par ailleurs, le système effectue des conversations avec les utilisateurs en sauvegardant l'historique de conversation (traçabilité) à chaque utilisation de l'assistant, afin d'améliorer au fur et à mesure les réponses générées.

En conclusion, nous pouvons dire que nous avons tiré une idée claire sur les difficultés et les obstacles affrontés lors de l'implémentation de chacun des modules de notre assistant vocal, ainsi que les apports et contributions pour faire face à ces difficultés.

Lors de l'implémentation du module de reconnaissance vocale, nous avons remarqué que le corpus coranique que nous avons construit et intégré dans notre système a beaucoup amélioré les résultats. Ainsi, grâce à ce module, nous avons pu constater la puissance des différents modèles de deep learning, ainsi que l'importance des modèles de langue dans les systèmes de reconnaissance vocale.

Au cours d'implémentation du module de TALN et structuration des requêtes, nous avons constaté malheureusement qu'il y a un manque aigu d'outils dédiés au TALN arabe. Cependant, nous avons fait face à ce problème en adaptant les outils dédiés au TALN des autres langues à la langue arabe et en développant ainsi nos propres outils. En revanche, les recherches et les études sont toujours en cours dans le but d'améliorer le TALN arabe.

Concernant le module de classification des requêtes, notre solution hybride nous a permis d'apercevoir l'importance des méthodes classiques symboliques, bien que les systèmes

modernes tendent à favoriser les approches par apprentissage. En effet, notre système a pris avantage des deux approches (symboliques et par apprentissage) afin d'implémenter une solution simple et efficace donnant de meilleurs résultats.

Pour la génération des réponses, nous avons analysé les dernières recherches effectuées dans ce domaine. Nous avons relevé que l'approche utilisant des transformateurs est considérée comme l'état de l'art dans la génération de réponses. Cette méthode basée sur le deep learning nous a permis d'approfondir nos connaissances en IA et d'implémenter ainsi notre propre transformateur en l'adaptant à la langue arabe. En effet, les résultats étaient assez satisfaisants, cependant, ils sont améliorables si nous fournissons des corpus ayant plus de données.

Durant l'implémentation du dernier module, qui est le module de synthèse vocale, nous avons ré-observé la puissance des différents modèles de deep learning, ainsi que l'importance des modèle de langue dans les systèmes de synthèse vocale.

Enfin, en réalisant notre application, nous avons pu voir l'importance de la simplicité lors de la conception d'un système complexe. En effet, toutes les approches et les architectures que nous avons adoptées tout au long de l'implémentation de notre solution, nous ont permis d'avoir une application efficace, ergonomique, modifiable et perfectible d'une façon simple et fluide.

Nous avons organisé notre travail en 4 phases correspondant aux 4 chapitres de ce manuscrit. Une première phase de documentation et de familiarisation avec le domaine d'assistance vocale, ainsi que les concepts fondamentaux de la langue arabe. Une deuxième phase d'analyse des articles analogues et d'étude de l'existant répondant à la même problématique que nous traitons.

Nous avons adopté une démarche de développement logiciel, avec une phase de conception à l'issue de laquelle nous avons détaillé l'architecture complète des modules du sous-système interactif et ceux du sous-système intelligent.

Enfin, vient la dernière phase de réalisation et résultats, dans laquelle nous avons mis en pratique nos compétences techniques acquises durant nos 5 années d'études en informatique, notamment en TALN, machine learning, datamining, modélisation et évaluation des performances du système, recherche d'information, services web, représentation des connaissances, réseaux etc. Ainsi qu'en deep learning et en développement Android que nous

avons appris durant nos recherches afin de développer le système dans sa globalité. Nous avons eu à manipuler des modules hétérogènes, dans des langages différents et des environnements différents, ce qui nous a permis de découvrir les différentes facettes de l'informatique en général et de l'IA en particulier. Nous avons ainsi eu à évaluer notre système en utilisant quelques techniques d'évaluation que nous avons apprises au cours de notre parcours.

En effet, la plus grande satisfaction de ce travail, est qu'il nous a permis de maîtriser le travail en équipe ainsi qu'individuel pour la réalisation d'un projet assez conséquent, mais aussi, de traiter un sujet récent et ambitieux dans un délai restreint et de y apporter notre propre contribution. Nous espérons avoir pu poser une pierre dans le domaine d'assistance vocale en langue arabe, qui pourrait aider et encourager les intéressés par ce domaine à s'impulser dans les recherches liées à cet axe.

Néanmoins, notre application est perfectible. Les résultats de notre modeste travail constituent les bases d'un travail à poursuivre et à améliorer pour une étude beaucoup plus approfondie qui pourra faire l'objet d'une thèse de doctorat.

En effet dans un tel projet, il serait essentiel d'étendre et d'améliorer les corpus arabes, étant donné que ces derniers sont considérés comme la pierre angulaire de l'apprentissage automatique. En effet, l'apprentissage automatique à partir de corpus est un procédé important pour les développeurs d'un système similaire. Ce procédé leur permet en effet de tirer profit des données linguistiques, qui se cachent derrière ces corpus et de les explorer d'une manière à atteindre la performance idéale pour leur système.

Ainsi, dans un système analogue, nous ne pouvons pas négliger l'aspect de l'humeur et de la fantaisie de l'utilisateur. Une suite logique et très intéressante de notre travail serait de se pencher sur les sautes d'humeur subites et passagères des utilisateurs. Par exemple, lorsque l'utilisateur lance une commande vocale et il est stressé, il serait intéressant que l'assistant vocal détecte son humeur à travers sa voix et accompagne ainsi la réponse à sa requête avec des programmes antistress (Aliments boosteurs d'énergie, exercices de respiration, postures de yoga, etc.).

Un autre aspect très intéressant dans un système tel que le nôtre est la compatibilité. En effet, choisir une plateforme pour une application revient à restreindre celle-ci. De cet fait, plus

l'assistant vocal est multiplateformes (iOS, Mac OS, Windows, Linux, etc.), plus il cible une grande catégorie d'utilisateurs.

Avant de conclure ce mémoire, nous citons les sources majeures qui ont nourri notre travail, et ce, pour à la fois, apercevoir la qualité du travail et permettre à ceux qui le lisent d'aller plus loin.

# BIBLIOGRAPHIE

- [1] Poushter J. (2016). “Smartphone ownership and internet usage continues to climb in emerging economies,” Pew research center, vol. 22, no. 1, pp. 1–44.
- [2] Kumar N., Gupta N., Naval C., Adhikari T., Tiwari M. N. N., Jani K., Sharma R., Rao V. V. and Katoch V. (2018). “Prevalence of tools of digital communications among population: A ground study in tribal, rural and urban slum,” J Int Med Sci Acad, vol. 31, no. 2.
- [3] Klevans R. L. and Rodman R. D. (1997). Voice recognition. Artech House, Inc.
- [4] L'opez G., Quesada L. and Guerrero L. A. (2017). “Alexa vs. siri vs. cortana vs. google assistant: a comparison of speech-based natural user interfaces,” in International Conference on Applied Human Factors and Ergonomics. Springer, pp. 241–250.
- [5] Farghaly A. and Shaalan K. (2009). “Arabic natural language processing: Challenges and solutions,” ACM Transactions on Asian Language Information Processing (TALIP), vol. 8, no. 4, pp. 1–22.
- [6] Milhorat P., Schl'ogl S., Chollet G., Boudy J., Esposito A. and Pelosi G. (2014). “Building the next generation of personal digital assistants,” in 2014 1st International Conference on Advanced Technologies for Signal and Image Processing (ATSIP). IEEE, pp. 458–463.
- [7] Miller G. A. (2003). “The cognitive revolution: a historical perspective,” Trends in cognitive sciences, vol. 7, no. 3, pp. 141–144.
- [8] Porter E. W. (1989). “Voice recognition system,” May 9 1989, uS Patent 4,829,576.
- [9] Yvon F. (2010). “Une petite introduction au traitement automatique des langues naturelles,” in Conference on Knowledge discovery and data mining, pp. 27–36.
- [10] Badaro G., Baly R., Hajj H., El-Hajj W., Shaban K. B., Habash N., Al-Sallab A. and Hamdi A. (2019). “A survey of opinion mining in arabic: a comprehensive system perspective covering challenges and advances in tools, resources, models, applications, and visualizations,” ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP), vol. 18, no. 3, pp. 1–52.
- [11] Habash N. Y. (2010). “Introduction to arabic natural language processing,” Synthesis Lectures on Human Language Technologies, vol. 3, no. 1, pp. 1–187.
- [12] Holes C. (2004). Modern Arabic: Structures, functions, and varieties. Georgetown University Press.
- [13] Versteegh K. (2014). Arabic language. Edinburgh University Press.

- [14] Aoun J. E., Benmamoun E. and Choueiri L. (2009). *The syntax of Arabic*. Cambridge University Press.
- [15] Shaalan K., Siddiqui S., Alkhatib M. and Monem A. A. (2019). “Challenges in arabic natural language processing,” *Computational Linguistics*.
- [16] Schultz T. and Waibel A. (2001). “Language-independent and languageadaptive acoustic modeling for speech recognition,” *Speech Communication*, vol. 35, no. 1-2, pp. 31–51.
- [17] Waibel A., Jain A. N., McNair A. E., Saito H., Hauptmann A. G. and Tebelskis J. (1991). “Janus: a speech-to-speech translation system using connectionist and symbolic processing strategies,” in *Acoustics, Speech, and Signal Processing, IEEE International Conference on*. IEEE Computer Society, pp. 793–796.
- [18] Agostaro F., Pilato G., Vassallo G. and Gaglio S. (2005). “A sub-symbolic approach to word modelling for domain specific speech recognition,” in *Seventh International Workshop on Computer Architecture for Machine Perception (CAMP’05)*. IEEE, pp. 321–326.
- [19] Deng L., Hinton G. and Kingsbury B. (2013). “New types of deep neural network learning for speech recognition and related applications: An overview,” in *2013 IEEE international conference on acoustics, speech and signal processing*. IEEE, pp. 8599–8603.
- [20] Hinton G., Deng L., Yu D., Dahl G. E., Mohamed A.-r., Jaitly N., Senior A., Vanhoucke V., Nguyen P., Sainath T. N. et al. (2012). “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97.
- [21] Hannun A., Case C., Casper J., Catanzaro B., Diamos G., Elsen E., Prenger R., Satheesh S., Sengupta S., Coates A. et al. (2014). “Deep speech: Scaling up end-to-end speech recognition,” *arXiv preprint arXiv:1412.5567*.
- [22] Amodei D., Ananthanarayanan S., Anubhai R., Bai J., Battenberg E., Case C., Casper J., Catanzaro B., Cheng Q., Chen G. et al. (2016). “Deep speech 2: End-to-end speech recognition in english and mandarin,” in *International conference on machine learning*, pp. 173–182.
- [23] Chiu C.-C., Sainath T. N., Wu Y., Prabhavalkar R., Nguyen P., Chen Z., Kannan A., Weiss R. J., Rao K., Gonina E. et al. (2018). “State-of-the-art speech recognition with sequence-to-sequence models,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 4774–4778.
- [24] McTear M., Callejas Z. and Griol D. (2016). “Conversational interfaces: devices, wearables, virtual agents, and robots,” in *The Conversational Interface*. Springer, pp. 283–308.
- [25] Reithinger N. and Klesen M. (1997). “Dialogue act classification using language models,” in *Fifth European Conference on Speech Communication and Technology*.

- [26] Ries K. (1999). “Hmm and neural network based speech act detection,” in 1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258), vol. 1. IEEE, pp. 497–500.
- [27] Kr’al P., Pavelka T. and Cerisara C. (2008). “Evaluation of dialogue act recognition approaches,” in 2008 IEEE Workshop on Machine Learning for Signal Processing. IEEE, pp. 492–497.
- [28] Pham B. T., Nguyen M. D., Bui K.-T. T., Prakash I., Chapi K. and Bui D. T. (2019). “A novel artificial intelligence approach based on multilayer perceptron neural network and biogeography-based optimization for predicting coefficient of consolidation of soil,” *Catena*, vol. 173, pp. 302–311.
- [29] Crook N., Granell R. and Pulman S. (2009). “Unsupervised classification of dialogue acts using a dirichlet process mixture model,” in Proceedings of the SIGDIAL 2009 Conference, pp. 341–348.
- [30] Jeong M. and Lee G. G. (2006) “Jointly predicting dialog act and named entity for spoken language understanding,” in 2006 IEEE Spoken Language Technology Workshop. IEEE, pp. 66–69.
- [31] He Y. and Young S. (2005). “Semantic processing using the hidden vector state model,” *Computer speech & language*, vol. 19, no. 1, pp. 85–106.
- [32] Noble W. S. (2006). “What is a support vector machine?” *Nature biotechnology*, vol. 24, no. 12, pp. 1565–1567.
- [33] Janetzko D. (2006). “Dialogue-based authoring of units of learning,” in Sixth IEEE International Conference on Advanced Learning Technologies (ICALT’06). IEEE, pp. 436–440.
- [34] Lokman A. S. and Zain J. M. (2010). “One-match and all-match categories for keywords matching in chatbot,” *American Journal of Applied Sciences*, vol. 7, no. 10, p. 1406.
- [35] AbuShawar B. and Atwell E. (2015). “Alice chatbot: trials and outputs,” *Computaci’on y Sistemas*, vol. 19, no. 4, pp. 625–632.
- [36] AbuShawar B. and Atwell E. (2002). A comparison between Alice and Elizabeth chatbot systems. University of Leeds, School of Computing research report 2002.19.
- [37] AbuShawar B. and Atwell E. (2009). “Arabic question-answering via instance based learning from an FAQ corpus,” Proceedings of the CL 2009 International Conference on Corpus Linguistics. UCREL, vol. 386.
- [38] Yan Z., Duan N., Bao J., Chen P., Zhou M., Li Z. and Zhou J. (2016). “Docchat: An information retrieval approach for chatbot engines using unstructured documents,” in

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 516–525.

[39] Wu Y., Wu W., Xing C., Zhou M. and Li Z. (2016). “Sequential matching network: A new architecture for multi-turn response selection in retrievalbased chatbots,” arXiv preprint arXiv:1612.01627.

[40] Koehn P. (2009). Statistical machine translation. Cambridge University Press.

[41] Cho K., Van Merriënboer B., Gulcehre C., Bahdanau D., Bougares F., Schwenk H. and Bengio Y. (2014). “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” arXiv preprint arXiv:1406.1078.

[42] Li J., Monroe W., Ritter A., Galley M., Gao J. and Jurafsky D. (2016). “Deep reinforcement learning for dialogue generation,” arXiv preprint arXiv:1606.01541.

[43] Rieser V. and Lemon O. (2011). Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation. Springer Science & Business Media.

[44] AbuShawar B. and Atwell E. (2003). “Machine learning from dialogue corpora to generate chatbots,” Expert Update journal, vol. 6, no. 3, pp. 25–29.

[45] Huang J., Zhou M. and Yang D. (2007). “Extracting chatbot knowledge from online discussion forums.” in IJCAI, vol. 7, pp. 423–428.

[46] Shrestha L. and McKeown K. (2004). “Detection of question-answer pairs in email conversations,” in COLING 2004: Proceedings of the 20<sup>th</sup> International Conference on Computational Linguistics, pp. 889–895.

[47] Rashad M., El-Bakry H. M., Isma’il I. R. and Mastorakis N. (2010). “An overview of text-to-speech synthesis techniques,” Latest trends on communications and information technology, pp. 84–89.

[48] Pagel V., Lenzo K. and Black A. (1998). “Letter to sound rules for accented lexicon compression,” arXiv preprint cmp-lg/9808010.

[49] Sejnowski T. J. and Rosenberg C. R. (1988). “Nettalk: A parallel network that learns to read aloud,” in Neurocomputing: foundations of research, pp. 661–672.

[50] Li L. (2012). “Analysis and application of mvvm pattern,” Microcomputer Applications, vol. 12, p. 019.

[51] Cohen M. H., Baluja S. and Moreno P. J. (2010). “Automatic language model update,” Jul. 13 2010, uS Patent 7,756,708.

- [52] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L. and Polosukhin I. (2017). "Attention is all you need," in Advances in neural information processing systems, pp. 5998–6008.
- [53] Selab E. and Guessoum A. (2015). "Building talaa, a free general and categorized arabic corpus." in ICAART (1), pp. 284–291.
- [54] Rotaru, Mihai. (2002). "Dialog act tagging using memory-based learning." Term project, University of Pittsburgh, 255-276.