Évaluation et Intégration de solutions techniques pour la création d'un assistant vocal

Manuel Labous

MINISTÈRE DE L'AGRICULTURE, DE L'AGROALIMENTAIRE ET DE LA FORÊT

ECOLE NATIONALE SUPÉRIEURE des SCIENCES AGRONOMIQUES de BORDEAUX AQUITAINE
1, cours du Général de Gaulle - CS 40201 – 33175 GRADIGNAN cedex

MEMOIRE de fin d'études

pour l'obtention du titre

d'Ingénieur de Bordeaux Sciences Agro

Évaluation et Intégration de solutions techniques pour la création d'un assistant vocal.

Labous Manuel

Spécialisation : AgroTIC

Etude réalisée à : Ekylibre, 4 rue Charles Domercq, 33130 Bègles

- 2019 -

MINISTÈRE DE L'AGRICULTURE, DE L'AGROALIMENTAIRE ET DE LA FORÊT

ECOLE NATIONALE SUPÉRIEURE des SCIENCES AGRONOMIQUES de BORDEAUX AQUITAINE
1, cours du Général de Gaulle - CS 40201 – 33175 GRADIGNAN cedex

MEMOIRE de fin d'études

pour l'obtention du titre

d'Ingénieur de Bordeaux Sciences Agro

Évaluation et Intégration de solutions techniques pour la création d'un assistant vocal.

Labous Manuel

Spécialisation : AgroTIC

Tuteur de stage : Rémi de Chazelles

Etude réalisée à : Ekylibre, 4 rue Charles Domercq, 33130 Bègles

- 2019 -

MINISTÈRE DE L'AGRICULTURE, DE L'AGROALIMENTAIRE ET DE LA FORÊT
NATIONAL SCHOOL of AGRICULTURAL SCIENCES and ENGINEERING, BORDEAUX AQUITAINE
1, cours du Général de Gaulle - CS 40201 – 33175 GRADIGNAN cedex

MASTER THESIS

Submitted in fulfillment of the requirements for the degree of

Agricultural Engineer, Bordeaux Sciences Agro

Assessment and integration of technical solutions for the creation of a vocal assistant

Labous Manuel

Specialisation : AgroTIC

Study completed at : Ekylibre, 4 rue Charles Domercq, 33130 Bègles

- 2019 -

Remerciements

Je souhaiterai tout d'abord remercier Jean Pierre Da-Costa, mon professeur référent école, pour ses conseils avisés tout au long du mémoire ainsi que pour sa disponibilité.
Je remercie aussi Nathalie Toulon, qui lors d'un point à mi-stage m'a aidé à mieux comprendre et organiser l'entièreté de ce mémoire.
J'adresse aussi un grand merci à Rémi, mon maître de stage, pour sa bienveillance tout au long du stage, ainsi que pour la façon dont il m'a encadré et orienté.
Merci à Malou Hyenard pour sa formation sur Zotero, qui m'aura finalement été d'une grande utilité.
Et merci à tous les anciens fumeurs d'Ekylibre pour avoir participé à mon arrêt progressif de la cigarette.

Sommaire

Introduction 1

1: Le projet Duke 2

1.1: Ekylibre, à l'origine de ce projet 2

1.1.1: Ekylibre, un éditeur de logiciels destinés au monde agricole 2

1.1.2: Organisation interne de l'entreprise 3

1.1.3: Les solutions proposées par Ekylibre: des outils pour la gestion des exploitations 4

1.2: Un projet pour intégrer le vocal aux solutions Ekylibre 5

1.2.1: L'avancée des assistants vocaux 5

1.2.2: L'origine du projet et ses enjeux 6

1.2.3: Duke, un projet renaissant de bases existantes 7

1.3: Démarche générale du projet 9

1.3.1: Questionnement sur l'intégration du Duke aux fonctionnalités d'Ekylibre 9

1.3.1.1 : Sur quelle plateforme proposer Duke 9

1.3.1.2 : Quelles fonctionnalités développer 10

1.3.2: Questionnement sur le mode de développement de Duke 11

1.3.2.1: Quelle place laisser au prototype précédent 11

1.3.2.2: Qui pour développer cet assistant 11

1.3.3: Planification des étapes du projet 12

2: De l'analyse des besoins au choix du mode de conception de l'assistant intelligent. 13

2.1: Besoins et attentes des agriculteurs 13

2.1.1: Méthodologie du recueil des besoins 13

2.1.2: Besoins identifiés 14

2.1.2.1: Priorisation des fonctionnalités à développer 14

2.1.2.2: Attente du monde agricole pour un assistant vocal : 16

2.2: Le choix du mode de conception pour répondre aux attentes du monde agricole 17

2.2.1: Analyse de l'existant 17

2.2.1.1: Entreprises proposant des solutions clef en main 17

2.2.1.2: Etude de solutions permettant le développement d'assistants 20

2.2.2 : Les solutions retenues 22

2.2.2.1 : Sous-traiter la création de Duke à EA4T 22

2.2.2.2: Continuer la création d'une solution en interne 24
2.3: Mise en place d'un plan de test 25
2.3.1: Restriction à un petit périmètre 25
2.3.2. Création d'une base de requêtes-type pour le test 26
2.3.3: Définition des métriques 26
3: Le développement d'une solution interne d'assistant vocal agricole 27
3.1: Etude des briques technologiques utilisables 28
3.1.1: Moteur de reconnaissance vocale, Speech to text (STT) 28
3.1.2: Moteur de synthèse vocale, Text to speech (TTS) 29
3.1.3: Outils de NLP (Natural language processing) 30
3.2: Choix des technologies utilisées 33
3.2.1: Socle de base de l'assistant vocal 33
3.2.2: Moteurs STT et TTS 34
3.2.3: Natural Language Processing: 35
3.3 Méthode de développement de la solution interne 36
3.3.1: Création d'un modèle d'analyse de phrases 36
3.3.1.1: Définition des entités à reconnaître 37
3.3.1.2: Entraînement du modèle par deep learning 37
3.3.2 Automatisation de la génération de la base d'apprentissage 39
3.3.3 Développement des actions de l'assistant 41
4: La phase de test, instigatrice de la mise en production de Duke 41
4.1 Etude de notre solution sur le périmètre des interventions 41
4.1.1: Essais sur les phrases du jeu de test 42
4.1.2: Autres métriques et synthèse 43
4.2: Prérequis au développement final 44
4.2.1: Rédaction des spécifications de développement 44
4.2.2: Le mode d'intégration de Duke 45
4.3: Perspectives 46
4.3.1: Phase de test 47
4.3.2: Le marché, et le mode de commercialisation de Duke 47
Conclusion 49
Bibliographie 50
Liste des annexes 52
Résumé - Abstract 55

Liste des illustrations :

Figure 1 : logo de l'entreprise Ekylibre…… 2
Figure 2 : principales fonctionnalités du logiciel de gestion Ekylibre…….. 4
Figure 3 : L'évolution des assistants personnels, d'IBM Simon à Siri…….. 6
Figure 4 : Les différentes catégories de fonctionnalités de Duke…… 10
Figure 5 : Niveau de pénibilité de certaines activités agricoles…… 14
Figure 6 : Logo de l'entreprise Agvoice…… 17
Figure 7 : Logo de l'entreprise EA4T…… 18
Figure 8 : Exemple d'interaction entre Duke sur Mycroft et un utilisateur….. 20
Figure 9 : Exemple d'annotation de phrases d'entraînement et légende…..21
Figure 10 : Matrice SWOT illustrant la possibilité de sous-traiter la création de Duke…… 23
Figure 11 : Matrice SWOT décrivant l'intérêt de développer la solution nous-même…… 25
Figure 12 : Architecture globale d'un assistant vocal……..27
Figure 13 : Une phrase et son analyse par l'outil de NLP spaCy……..31
Figure 14 : Arbre issu du traitement d'une requête par le parseur ANTLR…..32
Figure 15 : Comparaison de deux chaînes de caractères résultant de l'interprétation d'un algorithme phonétique……...33
Figure 16 : Ratio de ressemblance entre les codes phonétiques de ces deux mêmes phrases……..33
Figure 17 : Etude comparative de différents moteurs STT disponibles sur le marché……34
Figure 18 : exemple de requêtes annotées présentes dans la base d'apprentissage……..38
Figure 19 : Itérations de l'algorithme de spaCy et évolution de la valeur de perte……39
Figure 20 : Pipeline visant à entraîner le modèle avec les transcriptions des données réelles……..40
Figure 21 : Interface web de Duke sur Mycroft……46
Figure 22 : Les différents abonnements proposés par Ekylibre……48

Glossaire

API : Une API pour Application programming interface, est une interface de programmation permettant à un logiciel d'offrir ses services, et de transmettre des données à d'autres logiciels

Android - IOS : Ce sont des systèmes d'exploitation mobiles, respectivement développés par Google, et Apple.

Business model : C'est une représentation simplifiée des aspects majeurs de l'activité économique d'une entreprise

Deep learning : C'est un ensemble de méthodes liées à l'évolution de l'intelligence artificielle permettant une apprentissage automatique par la machine à partir de données de référence

ERP : Un ERP, pour Enterprise resource planning permet de gérer l'ensemble des processus et des fonctions d'une entreprise.

Framework : Cela désigne un ensemble de composants logiciels structurels servant à créer les fondations ou les grandes lignes d'un logiciel

NLP : Le NLP pour Natural language processing est un domaine impliquant la linguistique, l'informatique et l'intelligence artificielle dans le but de créer des outils pour le traitement du langage naturel.

Ontologie : En informatique, une ontologie est un ensemble structuré de termes et de concepts décrivant un champ d'information précis.

Open-source : La désignation 'Open-source' s'applique aux logiciels permettant une libre redistribution, un accès au code source, et la création de travaux dérivés. Ils respectent les critères établis par l'open source initiative.

R&D : Cela désigne l'ensemble des actions entreprises en vue d'accroître la somme des connaissances, et les applications découlant de ces nouvelles connaissances.

Skills : Dans les cas des assistants vocaux, un skill correspond à une fonctionnalité prise en charge.

STT : Pour Speech To Text. Cela désigne un moteur de reconnaissance vocale

TTS : Pour Text To Speech. Cela désigne un moteur de synthèse vocale

Wake-word : Cela désigne le mot déclenchant l'écoute par l'assistant vocal. Dans notre cas le wake-word est " Eh-Duke".

Introduction

Le milieu agricole est actuellement en pleine transition. De plus en plus d'acteurs du numérique proposent des outils digitaux à destination des agriculteurs. Parmi ces nouveaux instruments, nous pouvons distinguer plusieurs catégories. Ceux permettant l'automatisation et la robotisation de l'agriculture, ceux favorisant une agriculture de précision, et les systèmes permettant la gestion des données agricoles. Comme l'a récemment souligné le rapport Agriculture-Innovation 2025, l'exploitation des données numériques pour le monde agricole représente un des enjeux majeurs de cette nouvelle ère agricole et de plus en plus de solutions sont disponibles.

Ces outils sont parfois facilement adoptés au sein du monde agricole, mais de nombreux agriculteurs peinent encore à s'en approprier. Cette aversion pour la technologie peut être perçue comme un frein au développement massif d'outils technologiques dans le monde agricole, et cela pousse les innovateurs à proposer des interfaces homme-machine plus simples et plus ergonomiques.

En parallèle, et au sein des interfaces homme-machine, les solutions de reconnaissance et de synthèse vocale se sont très fortement développées. L'usage de la voix présente de nombreux intérêts. L'une des raisons pratiques évidentes est que l'humain peut énoncer 150 mots en une minute tandis qu'il n'en tape en moyenne que 40. Le système vocal permet aussi de ne pas obligatoirement faire face à un écran et peut faciliter la saisie dans des outils informatiques complexes. De plus les solutions vocales peuvent être intégrées dans tous types de supports numériques.
L'ensemble de ces avantages pourraient donc faciliter la transition vers une gestion numérique des exploitations agricoles.

En prenant conscience du manque d'accessibilité du numérique dans le milieu agricole, Ekylibre a donc décidé d'innover en tentant de développer une solution possédant une interface diamétralement opposée à ce qui se fait actuellement. Et c'est tout naturellement qu'Ekylibre a choisi le vocal comme nouvelle jonction entre leur solution et le milieu agricole. Le nom de ce projet est « Duke ».

L'objectif de ce mémoire sera d'évaluer un certain nombre de solutions techniques qui peuvent permettre la création d'un assistant vocal personnel lié aux solutions proposées par Ekylibre, et à destination du monde agricole. Nous commencerons par nous intéresser à Ekylibre en tant qu'entreprise innovante du secteur agricole. Cette présentation permettra de mieux délimiter le cadre de ce projet novateur. Nous décrirons ensuite les besoins exprimés par le monde agricole afin de justifier la création de cet assistant,

puis nous tenterons de proposer une méthode de développement de cet assistant. Nous finirons par décrire l'état et les perspectives du projet en lien avec l'avancement ayant été réalisé au cours des six derniers mois.

1: Le projet Duke

Avant de décrire « Duke » plus en détail, une rapide présentation du contexte va être effectuée pour clarifier l'environnement au sein duquel ce projet s'insère.

1.1: Ekylibre, à l'origine de ce projet

Afin d'éclaircir le contexte dans lequel ce mémoire a été rédigé, nous commencerons par décrire Ekylibre, entreprise évoluant dans l'écosystème de l'agriculture numérique, et nous décrirons son logiciel éponyme.

1.1.1: Ekylibre, un éditeur de logiciels destinés au monde agricole

Les prémices d'Ekylibre datent de 2005, lorsque Michel Antoli, alors président de l'ABUL (Association Bordelaise des Utilisateurs de logiciels Libres) et céréalier, travaille avec des étudiants de l'ENSEIRB sur un projet de gestion des très petites entreprises agricoles. Sept étudiants travaillent sur le projet, donc Brice Texier qui continuera à travailler sur le projet.

Figure 1 : Logo de l'entreprise Ekylibre (Source : Ekylibre.com)

Le projet Ekylibre, représenté par le hibou ci-dessus, a ensuite été créé en 2007 grâce au soutien du SACEA, (le service d'Aide et de Conseil aux Exploitants Agricoles). Ce projet avait pour but de créer un logiciel de gestion des exploitations agricoles dédié aux très petites entreprises. Il a été financé dès 2008 par le Conseil Régional d'Aquitaine et par l'Union Européenne.
En 2012, David Joulin, actuel président d'Ekylibre, rencontre Brice Texier, et décide de continuer le projet avec lui. Mais ce n'est qu'au début de l'année 2015 que la société Ekylibre, a été officiellement créée, avec pour mission principale de proposer un logiciel de gestion pour les petites entreprises agricoles. Cette entreprise, basée sur le modèle des *start-up* s'est depuis beaucoup développée jusqu'à avoir actuellement une quinzaine de salariés.

Le secteur dédié de cette entreprise est l'édition de logiciels. L'ensemble de ces logiciels sont open-source (libres) et sont à destination du monde agricole. Mais Ekylibre ne se limite pas à fournir des logiciels, elle propose aussi de la formation, du support pour la prise en main ainsi qu'une possibilité de créer des fonctionnalités sur mesure pour les grands groupes agricoles souscrivant à cette option.
Ekylibre gère actuellement des types d'agriculture très variés allant de la grande culture, à l'élevage, en passant par la viticulture.

Tous les agriculteurs peuvent obtenir gratuitement l'ensemble des logiciels d'Ekylibre grâce au caractère open source de ces produits, et le *business model* d'Ekylibre se base sur un abonnement proposé à l'ensemble de ces travailleurs agricoles pour leur proposer un hébergement du logiciel, l'ensemble des mises à jour, le support et les offres d'intégration.
La seconde partie du modèle économique correspond à toutes les activités annexes telles que les développements sur mesure pour les clients le réclamant, le conseil, la formation et l'audit.

1.1.2: Organisation interne de l'entreprise

Le fonctionnement d'Ekylibre s'organise autour de pôles qui interagissent entre eux de façon régulière. Ceux-ci sont composés d'un responsable et de plusieurs ekylibristes (nom donné aux employés d'Ekylibre) Étant donné la nature de *start-up* de l'entreprise, sa récente création et le faible nombre de salariés, certaines personnes peuvent être intégrées dans plusieurs pôles. Les différents pôles sont les suivants :

- Commercial et marketing
- Communication
- Administration systèmes
- Développement
- R & D (Recherche et Développement)
- Intégration & projets

Le travail effectué pour la conception de Duke s'insère dans ce cas dans le pôle R&D, qui est dirigé par David Joulin, co-fondateur et président d'Ekylibre.
Après avoir abordé l'historique et l'organisation actuelle de l'entreprise, nous décrirons brièvement les solutions qu'Ekylibre propose. Son principal logiciel est aussi nommé Ekylibre, et une rapide présentation sera faite par la suite dans ce mémoire.

1.1.3: Les solutions proposées par Ekylibre: des outils pour la gestion des exploitations

Ekylibre développe et édite un logiciel éponyme qui permet aux agriculteurs une gestion facilitée de leur exploitation agricole. Le principal avantage du logiciel Ekylibre est qu'il facilite tous les compartiments de la gestion d'exploitation, en passant notamment par la traçabilité ou la comptabilité. Les utilisateurs n'ont donc plus la contraintes d'utiliser de nombreux outils numériques non-interopérables. Toute la gestion peut désormais être groupée dans Ekylibre.
Les grandes catégories de fonctionnalités sont décrites dans la figure suivante.

Figure 2 : Principales fonctionnalités du logiciel de gestion Ekylibre (source : P.Tena. Conception d'un *serious game* sur le thème de la gestion des exploitations agricoles. 2017. p.14)

L'outil est actuellement édité sous licence libre, et l'ensemble du code source est disponible sur le *github* d'Ekylibre : https://github.ekylibre.com, ainsi que sur le *gitlab* : https://gitlab.com/ekylibre
En plus de sa solution Ekylibre, la société travaille sur la mise en place du programme Zéro saisie, ayant pour finalité de limiter la contrainte que représente la saisie pour les agriculteurs. Deux composantes majeures existent au sein de ce programme, il s'agit de :

- Zéro, une application mobile se basant sur l'interopérabilité, qui limite la saisie en récupérant directement des informations depuis des objets connectés permettant une collecte (ex : GPS du téléphone, stations météorologiques..). Ces données sont ensuite automatiquement

entrées dans le logiciel et la centralisation a donc lieu sans aucune intervention de l'agriculteur.
- Duke, le projet d'assistant vocal d'Ekylibre, qui propose à l'agriculteur de rentrer l'ensemble de ses informations dans l'ERP d'Ekylibre uniquement grâce à sa voix.

Ce mémoire sera centré sur l'assistant intelligent d'Ekylibre : "Duke". Etant donné que ce projet se base sur la technologie de reconnaissance vocale, nous expliciterons dans le suite de ce mémoire l'état d'avancement actuel des technologies vocales.

1.2: Un projet pour intégrer le vocal aux solutions Ekylibre

1.2.1: L'avancée des assistants vocaux

Le premier outil qui a permis une reconnaissance vocale numérique fut conçu par IBM en 1961. Cet ordinateur, était capable de reconnaître 16 mots ainsi que les chiffres allant de 0 à 9. Ce premier outil fut suivi dans les années 70 par un projet de l'université de Pittsburgh financé par le département de la défense américain. La finalité de leurs travaux mena à « Harpy », un outil capable de maîtriser plus de mille mots de vocabulaires. Ce même groupe de scientifiques travailla dix ans plus tard à développer un système ayant la capacité de comprendre des séquences de mots. Ces outils furent principalement utilisés pour la création de logiciels automatisés de dictée médicale.

C'est uniquement dans les années 90 que la technologie de reconnaissance vocale devient une caractéristique des ordinateurs personnels, et que le premier assistant vocal tel que nous le connaissons actuellement apparaît sur le smartphone IBM Simon.

Au XXIe siècle, Apple lance en 2011 Siri sur Iphone 4S, après avoir acquis l'entreprise SIRI qui était aussi financée par le département de la défense américaine. Siri va alors démocratiser l'utilisation de ce type de technologies et deviendra une référence parmi les assistants vocaux. Siri est alors accessible sur un téléphone à partir d'un simple bouton latéral, ce qui montre l'importance accordée par le constructeur à cette fonctionnalité.
La figure ci-dessous permet de visualiser l'évolution dans le temps entre la première technologie vocale, numérique et portative disponible sur IBM Simon, et SIRI actuellement disponible sur Apple.

Figure 3 : L'évolution des assistants personnels, d'IBM Simon (à gauche) à SIRI (à droite)

Google et Amazon lanceront aussi leurs assistants, intégrés dans plusieurs types de plateformes tels que des enceintes intelligentes ou des mobiles. La seconde décennie des années 2000 sera très marquée par des avancées majeures sur le marché du vocal pour les grandes entreprises du numérique. L'ensemble de ces progrès importants sont décrits sur l'annexe I, issu de l'article d'Ava Muchtler.
A la fin des années 2010, le vocal est en pleine expansion, et celui-ci est de plus en plus accessible, grâce à la disponibilité des technologies de certains géants du web, ou grâce à la naissance de certaines entreprises proposant des outils libres.
Cette accessibilité a été une composante principale du lancement du projet Duke par Ekylibre.

1.2.2: L'origine du projet et ses enjeux

Comme énoncé précédemment, le développement de logiciels à destination du monde agricole nécessite des interfaces très simplifiées pour

convenir à l'ensemble des agriculteurs, qui ne sont pas forcément proches des nouveautés technologiques. Pour un développement massif des logiciels édités par Ekylibre au sein de la communauté agricole, il est donc nécessaire d'offrir une alternative intuitive aux interfaces complexes d'un ERP.
De nombreux paramètres sont à l'origine du projet Duke. Nous pouvons entre autre citer le constat précédemment établi, ainsi que les fortes avancées des solutions vocales. Un autre point important est le fait que les interfaces associées à ce type de technologie sont rapidement devenues les plus simples et les plus intuitives disponibles sur le marché.
Les objectifs principaux de Duke sont alors :

- faciliter l'interface entre les agriculteurs et la solution Ekylibre
- attirer les clients réticents à l'idée d'utiliser des interfaces numériques complexes
- faire diminuer le temps alloué par les agriculteurs à la saisie de données

Duke avait été pensé pour être disponible sur des enceintes intelligentes, avec par exemple un RaspBerry, mais la question est de nouveau posée pour obtenir une accessibilité la plus importante possible.
Une ébauche du projet avait été développée en 2016, puis ce projet a été mis en pause à cause de la présence de nombreuses contraintes technologiques. Duke n'était alors plus une priorité pour Ekylibre.
Aujourd'hui, le projet Duke est relancé, car la solution Ekylibre est fonctionnelle et car beaucoup de travail a été effectué sur l'application Zéro.

1.2.3: Duke, un projet renaissant de bases existantes

Duke a d'abord été lancé en 2016, dans le cadre du projet Zéro Saisie. Rémi de Chazelles était l'ingénieur travaillant en recherche et développement sur ce projet. L'objectif au lancement était de pouvoir intégrer les technologies vocales pour réaliser toutes les activités disponibles dans l'application Ekylibre.
Après plus d'une année de travail et une année de pause, en 2019, le projet Duke a été relancé à partir de l'ébauche effectuée en 2016. Cette première ébauche avait permis le développement d'un prototype créé sur les bases d'un logiciel open source : Mycroft.
Le premier prototype était un assistant vocal développé sur le principe de Skills.
Ce principe consiste à attribuer des mots clés à certains types de fonctionnalités, puis à créer les actions associées à cette fonctionnalité. Lorsque le mot clé sera reconnu dans une requête utilisateur, le skill sera lancé et les actions seront effectuées.

De nombreux modèles d'assistants vocaux tels qu'Alexa d'Amazon travaillent sur le modèle des *skills*.
Mycroft permet donc de travailler avec des skills, ce qui peut faciliter la tâche car certains skills tels que l'obtention de données météorologiques, ou l'obtention de la date et de l'heure, étaient déjà développés.
Deux modèles avaient été tentés :

- Un modèle avec plusieurs *skills* :
  - InterventionSkill : pour la saisie des interventions
  - IssueSkill : pour la saisie des incidents
  - WeatherSkill : pour obtenir des informations météo sur la parcelle
- Un modèle avec un *skill* unique : EkylibreSkill qui comprenait toutes les fonctionnalités précédemment énumérées en un.

Le modèle comprenant plusieurs *skills* a été assez concluant, mais nécessitait un mot clé pour chacune des fonctionnalités. Il fallait par exemple prononcer « enregistre » pour saisir une intervention, et « problème » pour saisir un incident. Cela impliquait donc que l'utilisateur apprenne un nombre de mots-clés assez important, ce qui dénotait une des limites de la méthode.
Le modèle de l'EkylibreSkill a quant à lui été peu concluant. La reconnaissance entre deux fonctionnalités était hasardeuse et de très nombreuses requêtes n'étaient pas comprises.

Au début de ce stage, la finalité était de déterminer les contraintes spécifiques liées à l'utilisation du vocal dans le monde agricole, et de pouvoir fixer les technologies pouvant être utilisées pour proposer aux utilisateurs d'Ekylibre un assistant personnel fonctionnel. Le présent mémoire tentera donc de répondre à la problématique ci-dessous.

PROBLÉMATIQUE :

Comment sélectionner, intégrer et évaluer les meilleures alternatives techniques pour le développement de l'assistant vocal d'Ekylibre répondant au mieux aux besoins de ses utilisateurs ?

Étant donné que toutes les fonctionnalités ne seront pas développées, nous nous intéresserons particulièrement durant cette étude au cas de l'enregistrement des interventions.
Après avoir exprimé le contexte du projet ainsi que la problématique de ce mémoire, nous allons nous intéresser à la démarche du projet, lors de sa reprise en 2019.

1.3: Démarche générale du projet

La reprise du projet a induit de nombreuses questions afin de repartir sur de bonnes bases. Les premières questions concernent la méthode de déploiement de Duke.

1.3.1: Questionnement sur l'intégration du Duke aux fonctionnalités d'Ekylibre

Duke aura pour rôle de créer une nouvelle interface entre la solution d'Ekylibre et ses utilisateurs. Il est nécessaire de réfléchir aux options s'offrant à nous concernant le choix du support sur lequel il s'opérera.

1.3.1.1 : Sur quelle plateforme proposer Duke

Pour sélectionner la plateforme qui aura pour rôle de supporter l'assistant vocal, plusieurs choix s'offrent à nous. Une étude comparative a donc été effectuée pour déterminer les forces et les inconvénients de chacun d'entre eux.
Parmi les critères de comparaison, les suivants ont été retenus :
- la facilité d'intégration (possibilité de rajouter Duke à l'architecture existante)
- la présence d'un micro
- la couverture réseau
- la possibilité de déplacer le support (jusque dans les zones de cultures par exemple)

Les options s'offrant à nous sont les suivantes :

- Le smartphone
    - Le smartphone possède déjà un micro, bien qu'il soit rarement de bonne qualité
    - Le smartphone est un outil portatif
    - La couverture réseau n'est pas toujours assurée
- Le site web
    - Le site web offre une grande facilité d'intégration de Duke
    - Le site web offre le maximum de visibilité
    - Le site est accessible depuis un smartphone
    - Le site est créé pour un ordinateur, qui n'aura pas forcément de micro et n'est pas portatif
- L'enceinte intelligente
    - L'enceinte intelligente possède un micro de bonne qualité
    - L'enceinte serait spécifique à Duke, avec une faible consommation et un fonctionnement permanent en arrière-plan.
    - L'enceinte entraînerait des coûts supplémentaires, et n'est pas portative

Tous les supports présentent des avantages spécifiques, mais le site web semble être le meilleur compromis, notamment car le site peut être accessible depuis un smartphone. Cependant, les autres options sont aussi viables, et à l'heure actuelle, aucune décision définitive n'a été prise concernant le support pour Duke.

L'objectif est de ne pas se fermer de portes, et de tenter de développer un outil pouvant être intégré sur tous les supports énumérés ci-dessus.

Après avoir étudié les plateformes potentielles, nous tenterons de décrire rapidement l'ensemble des fonctions que Duke pourra remplir.

1.3.1.2 : Quelles fonctionnalités développer

Duke a pour rôle de faire l'interface entre l'ERP Ekylibre et ses utilisateurs, cependant l'ERP permet la gestion de l'ensemble de l'exploitation et il n'est pas envisagé de pouvoir gérer l'ensemble des paramètres avec la voix. Une analyse des besoins plus poussée nous permettra par la suite de déterminer quels sont les fonctionnalité d'Ekylibre à développer prioritairement.

Cependant, les fonctionnalités à développer peuvent d'ores et déjà être décomposées en trois parties :

- Les fonctionnalités de saisie liées à l'ERP d'Ekylibre
- Les fonctionnalités d'information générale
- Les fonctionnalités élémentaires d'un assistant vocal

Elles sont résumées dans la figure ci-dessous.

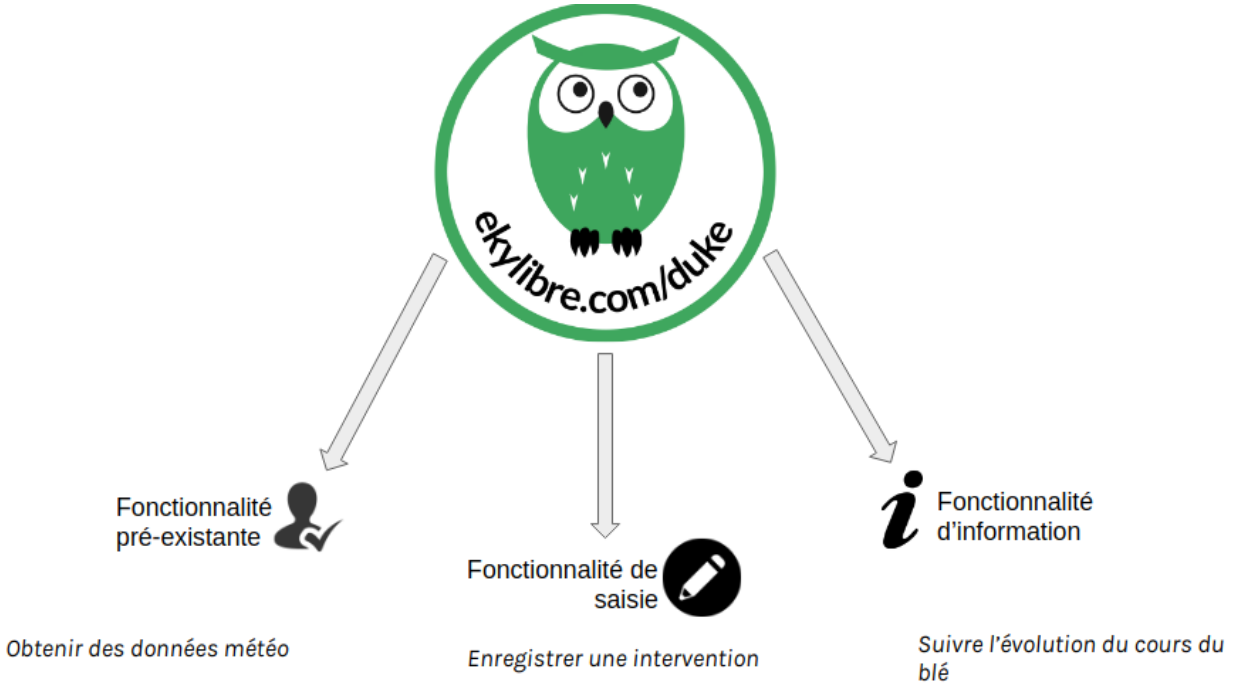


Figure 4 : Les différentes catégories de fonctionnalités de Duke

L'ensemble de ces fonctionnalités, en lien avec le support souhaité pour proposer Duke, vont conditionner son mode de développement, que nous allons détailler par la suite.

1.3.2: Questionnement sur le mode de développement de Duke

Étant donné qu'un prototype avait été développé, l'étape suivante de ce questionnement consiste à déterminer si celui-ci peut être réutilisé pour ce nouveau départ

1.3.2.1: Quelle place laisser au prototype précédent

A l'aube d'un nouvel exemplaire, il est nécessaire d'évaluer les forces et les faiblesses de l'ancien prototype pour déterminer si celui-ci peut être en partie réutilisé.
Le prototype précédent, était réalisé sur Mycroft, un outil open-source développé par une entreprise américaine. Le langage de programmation associé à Mycroft est du python, qui est compatible avec l'ensemble des supports envisagés pour supporter Duke. La première ébauche a aussi servi de proof of concept, montrant la faisabilité d'un assistant malgré quelques contraintes techniques.
Le code présent dans ce premier prototype peut être conservé. En effet, de nombreux éléments sont réutilisables comme par exemples les envois de données aux APIs d'Ekylibre.

Depuis plus d'un an, Mycroft a aussi fait beaucoup de progrès grâce au travail de leurs équipes de développeurs, ainsi qu'aux contributions de la communauté open-source, donc Mycroft semble encore être une solution d'actualité qui devra être envisagée parmi les possibilités techniques à sélectionner.
Il est donc possible que nous soyons amenés à retravailler à partir de l'ancien prototype, mais d'autre possibilités existent, et elles seront abordées dans la partie suivante.

1.3.2.2: Qui pour développer cet assistant

Actuellement, de nombreuses solutions techniques permettent à des développeurs de créer eux même leurs propres assistants vocaux, certains de ces outils sont *open source* et très respectueux de la vie privée de leurs utilisateurs. Un état des lieux plus profond sera présent dans la suite de ce mémoire.

Le point à retenir est qu'il est tout à fait possible d'internaliser la compétence du développement d'un assistant vocal au sein d'Ekylibre, avec pour exemple, les prémices ayant été réalisées sur Mycroft.

Cependant, un nombre conséquent d'entreprises ont fait de la création d'assistants vocaux leur cœur de métier. Elles ont créé leurs propres solutions qui sont dotées de spécificités très intéressantes.
Des entreprises françaises ont par exemple travaillé sur leur propre moteur de reconnaissance vocale en français, qui peut être très perfectionné et qui fonctionne généralement mieux que ceux proposés par les grosses entreprises du numérique. Les moteurs de synthèse vocale privés de ces entreprises sont eux aussi plus réalistes que ceux des géants du marché, qui ne ressemble que très peu à une voix humaine.
L'expertise de ces professionnels pourraient nous permettre d'obtenir un assistant intelligent plus perfectionné. Toutes ces options seront comparées par la suite, et après avoir introduit le projet plus en détail, nous allons nous intéresser à sa planification.

1.3.3: Planification des étapes du projet

Cette partie décrira la planification des étapes qui nous aura permis d'aboutir à une bonne gestion de projet, avec des délais respectés et une méthodologie de travail cohérente.
Les étapes du projet ont été les suivantes :

- Le cadrage du projet : Étape permettant de définir les objectifs visés par Ekylibre, et de mieux comprendre l'ensemble des enjeux pour l'entreprise.
- Le recueil et l'analyse des besoins : Recueil des besoins réalisé auprès d'agriculteurs partenaires d'Ekylibre. Les attentes et les craintes ont été recensées pour permettre la création d'un assistant le plus adapté possible à leurs besoins.
- L'état des lieux : Passage en revue de l'ensemble des entreprises proposant la création d'assistant vocaux, de toutes les solutions permettant la création interne d'assistants et de toutes les briques technologiques utilisables dans le cas de l'internalisation des compétences.
- La mise en place d'un plan de test : Phase au cours de laquelle sont déterminés un périmètre restreint pour l'utilisation de la solution ainsi que les métriques de test, afin de choisir parmi les solutions de création d'assistants vocaux disponibles.
- La phase de développement interne : Au cours de cette étape, un premier modèle a été créé pour faire état de la possibilité qu'Ekylibre crée lui même son assistant.
- La phase de comparaison : Une comparaison a été effectuée entre les différentes solutions techniques retenues précédemment, pour savoir quel serait le meilleur mode de développement de « Duke ».

- La phase de pré-production : Cette étape a permis la réalisation des spécifications de développement, nécessaire pour le bon déroulement de la phase de développement.

- La phase de développement de l'assistant : Une fois le choix du mode de développement effectué, celui-ci peut commencer selon les spécifications fonctionnelles et dans le but de respecter l'ensemble des contraintes et des besoins exprimées par les utilisateurs.

Une fois cette planification effectuée, nous pouvons nous intéresser à la première étape correspondant à l'analyse des besoins du monde agricole.

2: De l'analyse des besoins au choix du mode de conception de l'assistant intelligent.

Après avoir défini plus en détail le contexte et les enjeux de ce projet pour Ekylibre, il est nécessaire de s'intéresser aux besoin des futurs utilisateurs plus en profondeur.
La partie suivante décrit donc la méthode employée pour recueillir et déterminer les besoins des agriculteurs.

2.1: Besoins et attentes des agriculteurs

Afin de recueillir les besoins des utilisateurs, plusieurs moyens ont été employés, ils seront décrits dans la suite du mémoire

2.1.1: Méthodologie du recueil des besoins

Le recueil des besoins a été effectué grâce à une enquête proposée à des agriculteurs. L'objectif était de mieux connaître les activités contraignantes auxquelles les agriculteurs sont exposés, et la façon dont ils appréhendent les outils de reconnaissance vocale.
L'enquête était composée de plusieurs types de questions. Ce sondage contenait des questions fermées, demandant par exemple de prioriser certaines tâches selon la contrainte ressentie, et des questions ouvertes enquêtant sur les différents problèmes que les usagers pouvaient s'attendre à rencontrer avec l'utilisation de technologies vocales. Un extrait de cette enquête est disponible en annexe II.
Ce sondage visait plus précisément à connaître les réponses aux questions suivantes :

-Connaître le secteur d'activité de l'agriculteur sondé

-Connaître les types de travaux les plus pénibles à effectuer parmi une liste de tâches agricoles.

-Déterminer un classement de certaines tâches de saisie selon leur pénibilité

-Obtenir des exemples de sous-tâches pesantes
-Obtenir des exemples de tâches qu'ils souhaiteraient voir facilitées
-Entrevoir les premiers freins que les utilisateurs percevaient à l'utilisation du vocal.
-Entrevoir les premiers freins que les utilisateurs percevaient à l'utilisation d'objets connectés.

Ce sondage a aussi été proposé sur le forum d'Ekylibre.com, en étant épinglé pour obtenir une bonne visibilité. Il fut aussi envoyé à une petite liste d'agriculteurs nommés les « ekytesteurs ». Ces ekytesteurs sont des agriculteurs enthousiasmés par le numérique, et qui s'intéressent aux innovations qu'Ekylibre peut apporter.

A la suite de ce sondage, nous disposions d'une petite dizaine de réponses que nous avons pu tenter d'interpréter. Le panel était composé de personnes représentant différents secteurs de l'agriculture.

Les besoins que nous avons pu identifier sont présentés dans la partie suivante.

2.1.2: Besoins identifiés

A partir du sondage réalisé au préalable, nous avons pu obtenir une première approximation de plusieurs paramètres sur lesquels les agriculteurs avaient été questionnés.

2.1.2.1: Priorisation des fonctionnalités à développer

La pénibilité de certains types de travaux a pu être évaluée et le graphique suivant permet de comprendre quelles activités sont jugées les plus contraignantes par les agriculteurs.

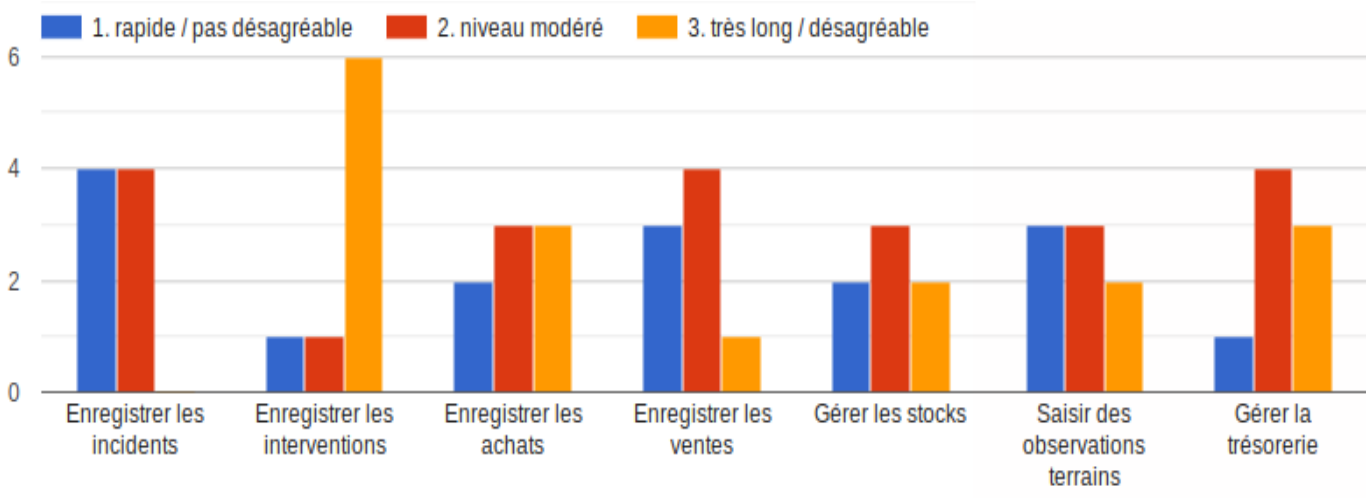


Figure 5 : Niveau de pénibilité de certaines activités agricoles

Ce résultat nous a permis de pouvoir envisager une première priorisation des fonctionnalités.
Cette première priorisation est la suivante :

1. Enregistrement des interventions
2. Gestion de la trésorerie/comptabilité
3. Gestion des achats
4. Gestion des stocks
5. Saisie des interventions terrains
6. Gestion des ventes
7. Saisie des incidents

La saisie des interventions, est un tâche commune à tous les types d'agriculture. Celle-ci semble très contraignante, et la gestion de cette saisie sera donc la première fonctionnalité qu'Ekylibre développera. Comme annoncé à la suite de la problématique, ce mémoire se concentrera en particulier sur la mise en place de cette fonctionnalité.

Une seconde analyse a été réalisée à partir de ces résultats. Il s'agit d'une note de cadrage par périmètre, permettant à nouveau de prioriser les tâches, mais cette fois-ci selon le secteur d'activité des agriculteurs sondés.
Les principaux retours ont été les suivants (avec pour chaque secteur agricole, les tâches considérées les plus pénibles):

Élevage Bovin allaitant :

1. Enregistrer interventions
2. Enregistrer Achats
3. Gestion de la comptabilité

Elevage Moutons :

1. Gestion des stocks
2. Saisie des observations terrain
3. Saisie des documents réglementaires

Grandes cultures :

1. Enregistrer les interventions
2. Gérer la comptabilité
3. Enregistrer la traçabilité

Maraîchage :

1. Enregistrer les interventions
2. Enregistrer les achats

Ces informations peuvent nous servir si nous remarquons que Duke s'oriente vers un secteur agricole en particulier et non à une utilisation généraliste.
Une seconde partie de l'analyse des besoins consistait à obtenir des réponses quant à des attentes spécifiques du monde agricole.

2.1.2.2: Attente du monde agricole pour un assistant vocal :

Le sondage réalisé auprès des agriculteurs nous a donc permis de prioriser certaines des futures fonctionnalités de Duke. Cependant, un autre objectif de cette enquête était d'appréhender les problèmes auxquels les utilisateurs pouvaient déjà penser.

Nous avons donc pu extraire certaines préoccupations de ces questionnaires, qui servent de lignes directrices pour connaître les attentes des utilisateurs.

Le premier point important que les agriculteurs identifient comme un frein à l'utilisation du vocal en agriculture concerne la problématique de compréhension du langage par la machine.
En effet, les utilisateurs semblent inquiets concernant la faculté qu'aura une machine à comprendre des phrases complètes et à pouvoir en extraire les bonnes informations. De plus, les différents accents présents dans les milieux ruraux français peuvent complexifier la tâche pour l'assistant intelligent.

Une autre préoccupation concerne la grande exigence du monde agricole. Un outil vocal sera très vite considéré comme un gadget inutile si celui-ci ne fonctionne pas parfaitement dès les premiers essais de l'agriculteur. Cette exigence pourrait être un frein à sa mise en place globale.
Les agriculteurs sondés considèrent donc qu'il est important que l'assistant comprenne puis reconnaisse le langage de façon très précise, et que Duke puisse comprendre les intentions de ses utilisateurs sans qu'il n'y ait de contraintes syntaxiques dans les requêtes.

L'idée principale ressortant de ces informations est donc qu'il est nécessaire de construire un assistant vocal ayant une compréhension approfondie du langage naturel.

Un autre point souligné correspond à l'absence assez régulière de réseau mobile ou internet dans l'entièreté de l'exploitation. Pour les agriculteurs sondés, il serait nécessaire d'avoir un outil fonctionnant hors connexion.

Le dernier point souligné est le bruit ambiant présent au sein des exploitations agricoles. Il semblerait en effet compliqué de pouvoir utiliser Duke directement en étant dans son tracteur, ou au milieu du poulailler. Ce bruit ambiant est donc vu comme un contrainte, mais celui-ci est porteur d'information et pourrait à l'inverse nous faciliter la tâche dans certaines situations. Ces contraintes seront fondamentales pour le choix de la méthode

de développement de Duke, que nous allons aborder au cours de la section suivante.

2.2: Le choix du mode de conception pour répondre aux attentes du monde agricole

Afin de choisir le mode de conception optimal, la première étape consiste à faire un état de l'art approfondi concernant les assistants vocaux. Une analyse de l'existant est donc de rigueur.

2.2.1: Analyse de l'existant

Cette analyse de l'existant va être divisée en deux étapes distinctes. Nous étudierons dans un premier temps les entreprises dont l'activité principale est la création d'assistants vocaux avancés.

2.2.1.1: Entreprises proposant des solutions clef en main

Plusieurs entreprises sur le marché proposent de construire des assistants vocaux afin que ceux-ci soient ensuite utilisés à des fins commerciales. Parmi l'ensemble des acteurs du marché, deux ont attiré notre attention. Le premier est très intéressant de par sa spécialisation vers les assistants pour un usage dans le domaine agricole.

- Agvoice

Figure 6 : Logo de l'entreprise Agvoice

Agvoice est une société américaine basée à Atlanta, dont la mission principale est de fournir une technologie mobile d'interaction vocale destinée aux professionnels du milieu agro-alimentaire et agricole pour faciliter leurs enregistrements de données.
AgVoice est conçu pour une utilisation dans tous les domaines de l'écosystème agricole, incluant la gestion des cultures et des élevages, la santé et le bien-être animal, ainsi que la gestion des outils.
Leur équipe est composée de personnes ayant des profils très techniques, et ils ont développé un moteur de reconnaissance vocale spécifique au milieu

agricole permettant donc de pouvoir comprendre du langage naturel dans ce domaine de parole.
Leur technologie pourrait donc beaucoup nous intéresser afin de mieux comprendre le contexte de chacune de nos requêtes utilisateurs.
Agvoice fournit la solution à des intermédiaires et non directement au monde agricole, ainsi Ekylibre, en tant qu'éditeur de logiciel, pourrait être un très bon intermédiaire entre les agriculteurs français et leur technologie.
L'un des problèmes majeurs est le fait que leur moteur de reconnaissance vocal n'est actuellement disponible qu'en anglais. Ce problème pourrait être résolu, mais un grosse partie de leur expertise et de leur travail serait perdue. De plus, il s'agit uniquement d'une solution mobile développée pour IOS.

- EA4T

Figure 7 : Logo de l'entreprise EA4T

EA4T tente de réinventer l'interface homme-machine, afin faire surgir de nouveaux usages pour répondre à des problématiques universelles et aux besoins de tous. L'entreprise est basée à Talence (33) et regroupe de nombreuses compétences. Au sein d'EA4T, nous pouvons distinguer de nombreux profils techniques s'étant spécialisés dans les technologies vocales, ainsi que des docteurs en langage, qui jouent un rôle dans le traitement de ces informations vocales. Leur solution présente de nombreuses spécificités présentées par la suite.

- Toutes les technologies ont été réalisées par leurs soins, ils ne passent par aucune API externe donc les données utilisateur ne transitent pas par des entreprises parallèles.
- Ils possèdent un moteur de reconnaissance vocale créé sur la base de Kaldi open source (un moteur libre), et ont un taux de reconnaissance légèrement supérieur à celui des moteurs des grandes entreprises du numérique.
- Ils ne s'intéressent pas réellement au STT(Speech to Text) mais au STC (Speech to Context). L'objectif n'est plus uniquement de bien recevoir la

phrase de l'utilisateur mais de réellement la comprendre et l'interpréter en lien avec la discussion en cours.

- Une ontologie est utilisée pour bien comprendre le modèle des données reçues. L'ontologie est couplée au STT et au NLP (Natural language processing) pour obtenir une meilleure compréhension de la requête client.
- Du vocabulaire spécifique peut être ajouté au modèle, et certains mots de vocabulaire mal prononcés, inventés ou raccourcis par un utilisateur spécifique peuvent être compris avec un peu d'entraînement.
- Une adaptation en fonction du régionalisme peut être effectuée dans le but de comprendre différents accents. Avec certains corpus de texte retranscrit par des personnes possédant un accent, le modèle peut être entraîné.
- Au sein d'un domaine, la reconnaissance vocale peut être spécifique à un sous-domaine (Le domaine de la viticulture n'aura pas exactement le même assistant que la culture de céréales.)
- Quelques phrases peuvent être prononcées hors ligne par l'assistant pour indiquer qu'il ne peut actuellement pas répondre aux requêtes envoyées par l'utilisateur, mais il est envisageable d'enregistrer les requêtes en mode hors ligne, pour les traiter plus tard.
- Toutes leurs technologies sont disponibles grâce à des APIs. Ces assistants ne nécessitent pas énormément de stockage de données, mais sont en conséquence peu efficaces sans réseau disponible. Il est cependant important de noter qu'un faible fonctionnement hors ligne est envisageable, comme énoncé précédemment

Ces deux entreprises pourraient donc nous permettre de déployer un assistant vocal intelligent sans avoir la nécessité d'internaliser l'ensemble des compétences. Le maintien à jour et les nouveaux développements à partir d'une version de base pourraient ensuite être assurés par Ekylibre.

L'autre option consistant à l'internalisation des compétences nécessite un état des lieux de l'ensemble des solutions et des briques technologiques utilisables pour la mise en place de cet assistant. Ce *benchmark* sera présenté dans la suite de ce mémoire.

2.2.1.2: Etude de solutions permettant le développement d'assistants

Pour juger de la possibilité de créer un assistant vocal directement chez Ekylibre, nous comparerons deux solutions disponibles sur le marché.

- Mycroft

Mycroft est la solution initialement envisagée pour réaliser Duke. Le début de prototype avait été effectué pour Mycroft. Cet outil permet de créer des *skills* basés sur des reconnaisseurs d'intents. Les intents correspondent aux intentions de l'utilisateur lorsque celui-ci émet une requête. Ces intentions lorsqu'elles sont reconnues, permettent de lancer un *skill* un particulier, correspondant à une fonctionnalité. L'ensemble du code est *open-source*, et le projet est conçu pour que n'importe qui puisse créer des *skills* qui lui soient adaptés.

Mycroft n'est pas lié à des moteurs STT et TTS particuliers. Il est possible d'importer la grande majorité des moteurs disponibles via API sur un assistant créé à l'aide de Mycroft.
Mycroft peut permettre des conversations assez rudimentaires, mais son socle en Python permettant d'utiliser un très grand nombre d'outils pour le complexifier. Le Python permet aussi de répondre aux exigences du support pour Duke que nous avons pu déterminer précédemment. En effet, nous ne fermons aucune porte pour le choix du support en développant en Python.
De plus, certains *skills* sont déjà existants sur Mycroft comme par exemple celui permettant d'obtenir des informations météorologiques selon la position, ainsi que des alarmes et des minuteurs. Les fonctionnalités élémentaires sont donc déjà disponibles, et certaines fonctionnalités d'information peuvent facilement être développées à partir d'API externes.
La capture d'écran ci-dessous est la retranscription d'une discussion entre un utilisateur et Duke, son assistant. On remarque que Duke peut facilement faire des liens entre des phrases naturelles (en bleu) et leur retranscription théorique pour pouvoir déclencher des actions, illustrées ici par des réponses vocales et écrites (en jaune).

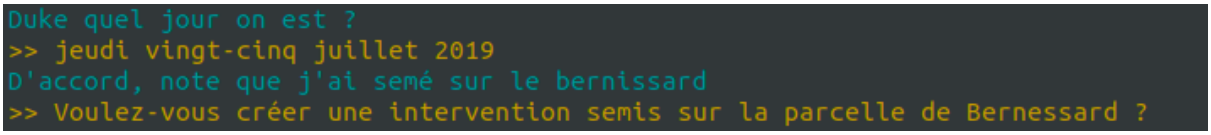


Figure 8 : Exemple d'interaction entre Duke sur Mycroft et un utilisateur.

Mycroft, de par son caractère *open source*, possède aussi une grande communauté travaillant régulièrement à la création de *skills*, ou à la mise en

place de nouvelles interfaces. Tout ce contenu est utilisable et commercialisable.

- Snips

Snips est un outil de service permettant initialement l'aide à la création d'assistant vocaux. C'est un outil comprenant un moteur STT très développé et un interpréteur d'intentions. Il est possible de créer des *skills*, en donnant à Snips des phrases-type pour que celui-ci s'entraîne.
La figure ci-dessous montre des exemples de phrases permettant l'entraînement du modèle sur Snips. Ces phrases doivent être annotées avec les différentes entités pour que le modèle puisse reconnaître facilement les éléments importants au sein d'une requête.

Figure 9 : Exemple d'annotation de phrases d'entraînement et légende.

Snips peut écouter l'utilisateur, proposer une transcription associée à un taux de confiance, essayer d'en extraire l'intention et lancer l'action adéquate. Snips peut parfois reconnaître des phrases très pointues, si le modèle a été correctement entraîné, mais son entraînement semble être prévu pour de la reconnaissance de commandes courtes. La compréhension du langage naturel est moins évidente car lorsqu'on s'éloigne légèrement des données d'entraînement du modèle, le taux de reconnaissance diminue très fortement. Ainsi une phrase exacte du modèle sera parfaitement reconnue, mais d'autres phrases formulant la même idée de façon légèrement différente ne le seront pas.
L'atout majeur de cette technologie est qu'elle est disponible hors ligne. La grande majorité des moteurs STT sont disponibles à partir du *cloud* et nécessitent donc une connexion Internet pour fonctionner, tandis que Snips

travaille avec un moteur en local et permet à notre assistant vocal de fonctionner complètement hors connexion.
Les briques technologiques de Snips semblent très développées mais celles-ci ne peuvent pas être utilisées sans que l'ensemble de la solution de Snips ne soit déployée. Un autre désavantage de Snips semble être sa difficulté de déploiement et le coût associé à une utilisation de leurs technologies dans un projet commercialisé.

D'autres solutions telles que Linto.ai ont été testées, mais une description exhaustive ne sera pas effectuée au cours de ce mémoire.
Après avoir effectué un *benchmark* des différentes solutions disponibles, deux solutions ont été retenues, celles-ci seront présentées dans la partie suivante.

2.2.2 : Les solutions retenues

Au moment de retenir des solutions, nous avons dû estimer la qualité des assistants pouvant être développés par les différentes méthodes. Pour choisir entre les deux entreprises spécialisées dans le vocal, les critères d'évaluation ont été les suivants :

- La qualité de compréhension des intentions d'utilisateurs agricoles français
- La disponibilité de l'entreprise, et son aptitude à comprendre nos problématiques
- Le mode de déploiement de leur solution technique

2.2.2.1 : Sous-traiter la création de Duke à EA4T

Après avoir mis en place ces critères, EA4T et AgVoice ont été comparés. Concernant la premier critère, EA4T semble être une meilleure option car ils sont habitués à travailler en français pour de la compréhension de langage naturel. Leur manque de connaissances agronomiques pourra éventuellement poser problème pour la compréhension de certains termes spécifiques, mais il paraît concevable de pouvoir partager notre expertise dans ce domaine. Leur mode de déploiement est aussi beaucoup plus diversifié que AgVoice qui ne propose qu'une solution mobile.

La première solution retenue serait donc de sous-traiter la création de l'assistant intelligent à EA4T. Cette entreprise spécialisée pourrait nous permettre d'obtenir rapidement un assistant fonctionnel et durable, pouvant être continuellement amélioré. Cependant, faire appel à une autre entreprise présente quelques inconvénients d'autonomie et de difficulté de déploiement.
Cette option est à confronter avec la création d'une solution en interne.
La matrice SWOT suivante permet d'illustrer les différents points spécifiques au sous-traitement de la tâche à EA4T.

Forces	Faiblesses
• Très bonne compréhension du langage naturel et possibilité de s'adapter aux bruits et accents communs du monde agricole • Fonctionnement (très limité) hors connexion • Solution durable dans le temps, pas de nécessité de changer le modèle une fois que celui-ci est en place	• Nécessité de faire appel à une autre société • Coût important de la solution • Temps important nécessaire pour l'implémentation des ontologies • Besoin de transmettre nos compétences agricoles
Opportunités	**Menaces**
• Permettre la mise à jour des ontologies d'Ekylibre • L'entreprise partenaire prévoit de l'amélioration dans leur solution dont Ekylibre pourrait profiter • Pouvoir utiliser la solution en différents langages et sur différentes plateformes.	• Ekylibre sera dépendant d'une entreprise partenaire, et ce pour les années à venir.

Figure 10 : Matrice SWOT illustrant la possibilité de sous traiter la création de Duke

Travailler avec EA4T implique de fournir un grand nombre de données à cette entreprise. Il serait nécessaire de leur proposer l'ensemble des ontologies d'Ekylibre, avec un modèle de données complet et une explication sur l'ensemble de l'architecture de nos données. Nous devrions aussi leur faire parvenir un ensemble de requêtes-type pour qu'ils puissent entraîner leur modèle sur du lexique agricole réel. Ces requêtes permettent à EA4T d'obtenir plus de précisions sur le domaine métier.
L'autre alternative envisagée consisterait à continuer le développement en interne.

2.2.2.2: Continuer la création d'une solution en interne

Concernant un développement interne, et afin de choisir entre Mycroft et Snips, les différents critères ont été :

- la qualité espérée de la solution finale
- la dispense de contraintes de développement
- la facilité de déploiement

Selon ces critères établis, Mycroft s'est avéré comme le meilleur choix notamment grâce à sa facilité de déploiement et grâce à la grande liberté qu'il offre aux développeurs dans la réalisation de leur assistant. Snips aurait été une solution payante difficilement déployable et nous imposant de nombreux outils, bien que cette entreprise possède certains outils techniques très avancés.

Il a donc été envisagé comme une autre issue viable, de continuer la solution à partir du prototype précédent en internalisant toutes les compétences. Les avantages sont très différents de ceux proposés par EA4T, et les faiblesses de Mycroft correspondent souvent à leurs points forts.

La matrice SWOT ci-dessous décrit en détail cette option :

Forces	**Faiblesses**
• Technologies *open-source*, faible coût • Développement rapide, et grande liberté dans les outils choisis • Possibilité de modifier facilement certaines fonctionnalités • Pas de besoin d'externaliser des compétences	• Mauvaise compréhension du langage naturel • Impossible d'adapter le vocal à la régionalité ou à des modèles acoustiques spécifiques du milieu agricole. • Nécessite d'être connecté à un réseau
Opportunités	**Menaces**
• *Skills* développés en continu par la communauté.	• Utilisateurs finaux très exigeants concernant la qualité du produit.

• Amélioration perpétuelle des moteurs de reconnaissance vocale. • Possibilité d'implémenter facilement cet assistant vers un navigateur Web	• Développement d'un système moins durable, qui sera probablement dépassé durant les prochaines années

Figure 11 : Matrice SWOT décrivant l'intérêt de développer la solution nous même.

Le choix doit donc être effectué entre deux méthodes de développement dont les finalités semblent très différentes. Etant donné que nous ne connaissons pas encore l'étendue du potentiel d'un assistant qui serait développé en interne , et que nous ne pouvons pas non plus connaître la qualité de l'assistant qu'EA4T pourrait créer sur un thème agricole dont ils n'ont aucune expertise, la meilleure alternative semble de mettre en place un plan de test pour pouvoir comparer les deux solutions retenues et faire le choix en amont du développement.

2.3: Mise en place d'un plan de test

La mise en place d'un plan de test va permettre de pouvoir comparer les deux solutions retenues. Il est nécessaire de pouvoir déterminer un environnement strict pour effectuer cette comparaison, et de déterminer les critères de test.
Nous allons tout d'abord nous intéresser au périmètre de test.

2.3.1: Restriction à un petit périmètre

Afin de pouvoir réaliser ce test, nous avons commencé par définir le périmètre qui servira à la comparaison des deux modèles d'assistant.
Afin que deux modèles puissent être réalisés rapidement, les fonctionnalités de l'assistant modèle se devaient d'être limitées. Le premier point fût donc de sélectionner une petite partie représentative d'un grand nombre de contraintes que l'assistant final pourrait rencontrer.
Le choix a été porté sur la saisie des interventions liées aux grandes cultures, avec des données de démonstration déjà existantes et provenant d'une réelle exploitation agricole.

La saisie des interventions consiste donc à pouvoir enregistrer une intervention culturale (semis, labour, pulvérisation, récolte, transport de récolte..) sur plusieurs parcelles existantes avec des noms variés, et en choisissant ou non des outils et un travailleur.

Les liste des travailleurs, outils, parcelles et interventions sont incluses dans les données de démonstration.

2.3.2. Création d'une base de requêtes-type pour le test

De plus, pour pouvoir tester les deux modèles, il est nécessaire de posséder une grosse base de requêtes-type d'utilisateurs. Pour éviter que la base de test et la base d'apprentissage d'EA4T ne soient trop proches l'une de l'autre, et puisque nous leur avions déjà fourni une grande liste de requêtes-type, il était nécessaire d'obtenir ces requêtes "test" d'une autre manière.

Nous avons donc contacté des agriculteurs via les comptes Twitter et Facebook d'Ekylibre pour leur demander la façon la plus naturelle par laquelle ils enregistreraient une intervention à la voix, en précisant la parcelle, l'outil et le travailleur.

Les réponses nous ont été envoyées via les réseaux sociaux, ou directement par mail, et sur le forum d'Ekylibre, et nous avons ainsi pu entamer la création d'une base de test.

2.3.3: Définition des métriques

Après avoir restreint le périmètre d'étude à un fragment de l'ensemble des données d'Ekylibre, il était nécessaire de définir comment nous pourrions juger du bon fonctionnement des solutions testées.

La première métrique de test correspond à l'efficacité de la compréhension de requêtes par les prototypes. La base de test contient une vingtaine de phrases qui sont enregistrées avec ou sans bruit ambiant, et par plusieurs voix ayant des tonalités variées. Le pourcentage de compréhension de ces requêtes est un premier élément de comparaison.

Le temps de réponse est aussi un élément de comparaison. Il est nécessaire de comparer le temps de réponse pour les solutions afin de s'assurer que le traitement du langage ne s'oppose pas à un dialogue fluide entre l'homme et la machine. Cette mesure du temps de réponse devrait être effectuée avec plusieurs type de connexion internet, de façon à mesurer l'efficacité pour des connexions faibles.

Le troisième élément mesuré est le taux de fonctionnement hors connexion.

Pour finir, la dernière métrique correspond à la facilité du déploiement de l'assistant vocal par Ekylibre ainsi qu'à la facilité d'intégration au sein de l'architecture existante chez Ekylibre.

Après avoir mis en place un plan de test permettant de comparer la solution développée par EA4T et celle qu'Ekylibre peut produire, un nouveau prototype doit donc être mis en place, dont l'efficacité doit pouvoir être mesurée au sein du périmètre défini précédemment.

3: Le développement d'une solution interne d'assistant vocal agricole

Un assistant vocal est composé de plusieurs briques technologiques permettant d'assurer l'ensemble des fonctions allant de la compréhension du langage, à la synthèse vocale, en passant par l'extraction de l'intention et le choix des actions à mener.
Le *wake-word* permet de réveiller l'assistant, les moteurs STT et TTS permettent respectivement la reconnaissance et la synthèse vocale, et le socle de l'assistant va permettre de réaliser les actions souhaitées.
L'ensemble de ces briques sont décrites dans le schéma ci-dessous retraçant l'architecture globale d'un assistant vocal

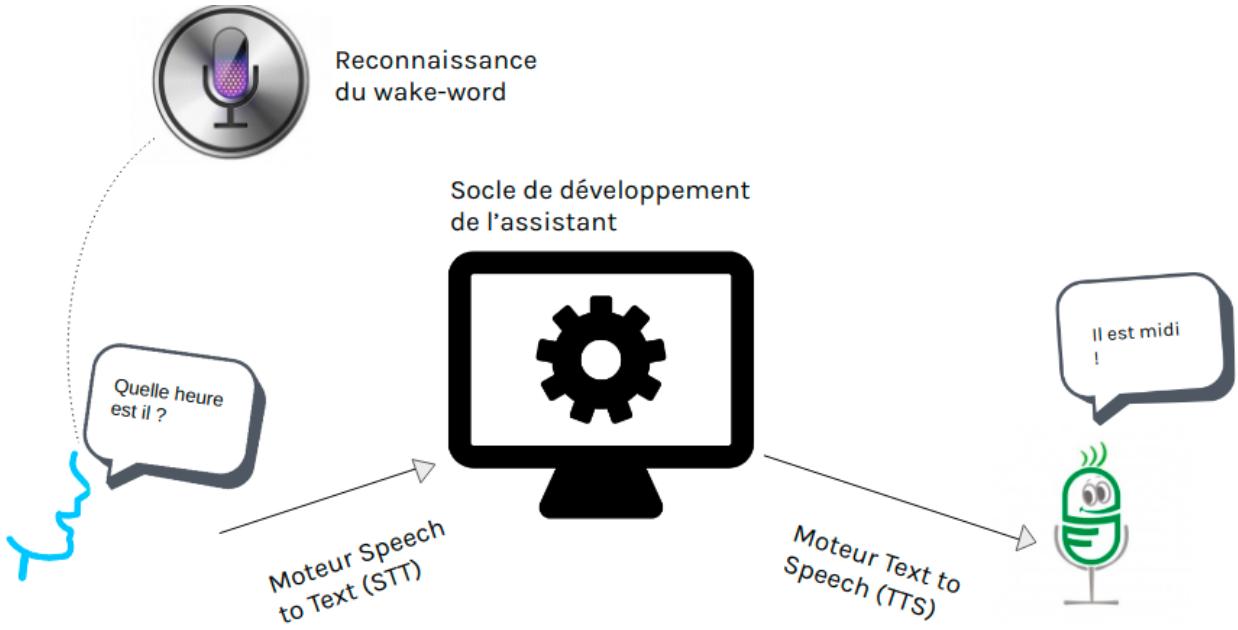


Figure 12 : Architecture globale d'un assistant vocal

L'étape de développement d'une ébauche de solution va permettre de prouver les qualités que pourraient présenter un assistant développé directement chez Ekylibre.

Afin de pouvoir commencer à créer notre modèle d'assistant intelligent, la première partie consiste à sélectionner les technologies à intégrer.
L'étape suivante consiste donc à approfondir nos connaissances sur l'état actuel des briques participant à la création d'un assistant.

3.1: Etude des briques technologiques utilisables

Les outils numériques utilisables peuvent être classés en deux catégories : les librairies permettant un traitement de données informatisées relatives à des séquences vocales humaines, et les technologies vocales, permettant d'établir un dialogue entre un utilisateur humain et la machine. Au sein des technologies vocales, nous commencerons par nous intéresser aux technologies de reconnaissances vocales disponibles.

3.1.1: Moteur de reconnaissance vocale, Speech to text (STT)

Les moteurs de *Speech to Text* (STT) permettent la transcription d'un message vocal en un texte. Ces systèmes de reconnaissance vocale sont basés sur des modèles de Markov cachés, car un signal sonore peut être assimilé à un signal stationnaire de courte durée et car ces modèles peuvent être entraînés automatiquement.
La majorité des grandes entreprises du numérique proposent d'accéder à leur moteur STT, disponible généralement via API. Nous avons donc mené une étude comparative.
Cette étude a été réalisée à partir d'un grand corpus de texte, mais uniquement une phrase de test représentative sera présentée ici pour plus de clarté.
La phrase d'essai choisie est la suivante :

"La repousse du couvert végétal et l'oïdium de la vigne, et l'helminthosporiose du blé"

Cette phrase a été enregistrée, puis proposée à tous les moteurs STT pour obtenir par la suite leur transcription et l'enregistrer.

- ***IBM watson :***

Retranscription :
"*ans de refus du couvert végétal et l'odieux de la vigne est nettement les deux yeux du p*"

- ***Google Cloud :***

Retranscription :
"la repousse du couvert végétal et l'oïdium de la vigne et l'île mystérieuse du blé"

- **Amazon AWS :**

Retranscription :
"La Repousses-de-couvert-végétal et l'eau Oïdium-de-la-vigne éléments tout sport, yeux du blé"

- **Microsoft Azure :**

Retranscription :

”La repousse du couvert végétal et l’oïdium de la vigne et l’élément au sport yeux du blé”

Sur la précision de l’interprétation, Microsoft Azure STT et Google Cloud STT semblent être les meilleurs choix disponibles en français.
Mais pour le choix du moteur STT, la précision n’est pas le seul critère.
L’ensemble des critères d’évaluation sont les suivants :

- ➔ La qualité du vocabulaire reconnu
- ➔ La possibilité de rajouter du lexique personnalisé (dans le cas de certains mots spécifiques au domaine agricole n’étant pas reconnus)
- ➔ Le prix
- ➔ La rapidité d’exécution
- ➔ L’accessibilité hors ligne

Après avoir évalué les moteurs STT, une rapide présentation des moteurs TTS sera effectuée.

3.1.2: Moteur de synthèse vocale, *Text to speech* (TTS)

Le moteur de *Text To Speech* (TTS) permet de synthétiser un message vocal à partir d’un texte. Ces outils sont très répandus en ligne, utilisés dans beaucoup de domaines et très accessibles. Quatre des outils les plus précis ont été comparés.
Les critères de sélection sont cette fois-ci beaucoup plus subjectifs, puisqu’il s’agit de sélectionner la voix se rapprochant le plus d’une voix humaine.
Contrairement au *Speech to text,* le *Text to speech* n’aura pas d’incidence sur le développement des fonctionnalités, et le moteur TTS sélectionné sera uniquement celui possédant la voix et l’accent le plus adapté à la situation, ainsi que la plus grande rapidité d'exécution, pour toujours maintenir un dialogue continu avec l’utilisateur.

- ***Google Cloud TTS :***

Google propose sa solution de *Speech to text*. La voix féminine disponible en français est très utilisée dans différents domaines mais n’est pas du tout la plus proche du réel. Certaines accentuations n’ont pas lieu d’être, les liaisons entre les mots sont mal prononcées et les ponctuations sont parfois mal prises en compte. Le temps de traitement d’une phrase est équivalent à 1/1, soit équivalent au temps que mettrait un utilisateur pour lire une phrase. Il est cependant possible de mettre un nombre illimité de phrases en cache, pour faire diminuer d’un facteur 100 le temps de traitement, et donc le temps d'exécution.

- ***IBM Watson TTS :***

IBM propose une solution très adéquate à nos besoins. Il s'agit d'une voix féminine peu accentuée, très neutre mais faisant très peu d'erreurs de prononciation ou de ponctuation. Le temps de traitement de ce moteur TTS est environ de 0.5/1, donc satisfaisant.

- ***BING TTS :***

Microsoft Bing propose un moteur relativement fluide dans différents langages, mais la voix française n'est pas convaincante. Le temps de traitement non plus. Ce service disponible par API sur Azure ne semble donc pas être une solution envisageable en français.

- ***Mimic :***

Mimic est le moteur TTS proposé d'office par Mycroft, leur solution est très lourde, mais la voix est convaincante en anglais. Le français n'est malheureusement pas encore disponible, ce qui est regrettable puisque leur solution est disponible hors ligne, et car il est aussi possible de mettre en cache un grand nombre de phrases pour un traitement quasi instantané.

3.1.3: Outils de NLP (Natural language processing)

Le *natural language processing* (NLP) vise à créer des outils de traitement de la langue naturelle pour diverses applications. Le NLP a été progressivement mis en œuvre dans des applications informatiques nécessitant l'intégration du langage humain à la machine. Plusieurs outils ont été évalués en prévision d'une future utilisation au cours de ce projet.

- **SpaCy**

SpaCy est une librairie Python permettant de faire du *natural language processing* . Cette librairie prend une phrase en entrée et nous permet d'effectuer plusieurs opérations de traitement. Par exemple, nous pouvons effectuer :

1. Tokenisation : Segmentation d'un texte en mots, ponctuations …
2. Part of speech Tagging : Associer le type de mot à chacun des *tokens* (ex : verbe, nom ..)
3. Dependency Parsing : Décrire les relations entre les *tokens* (ex : sujets, objets..)

4. Lemmatization : Donner la forme basique des mots (ex : lu → lire, réglées → règle)
5. Named Entity Recognition : Libeller des objets du monde réel (ex : Google → Entreprise)

SpaCy nous permet donc de mieux analyser les requêtes de l'utilisateur. Cette librairie peut ainsi nous faciliter la tâche pour reconnaître certains éléments clés dans les phrases reconnues par le moteur STT.
L'exemple ci-dessous présente un échantillon des types d'analyses qui peuvent être obtenues à partir de spaCy pour une requête type.

```
>>> J'ai labouré la parcelle Barnessard avec ma charrue reversible
```

```
J'          0    je          False   False   X'      PRON    PRON__Number=Sing|Person=1
ai          2    avoir       False   False   xx      AUX     AUX__Mood=Ind|Number=Sing|Person=1|Tense=Pres|VerbForm=Fin
labouré     5    labourer            False   False   xxxx    VERB    VERB__Gender=Masc|Number=Sing|Tense=Past|VerbForm=Part
la          13   le          False   False   xx      DET     DET__Definite=Def|Gender=Fem|Number=Sing|PronType=Art
parcelle         16   parcelle            False   False   xxxx    NOUN    NOUN__Gender=Fem|Number=Sing
Barnessard  25   Barnessard          False   False   Xxxxx   PROPN   PROPN___
avec        36   avec        False   False   xxxx    ADP     ADP___
ma          41   mon         False   False   xx      DET     DET__Gender=Fem|Number=Sing|Poss=Yes
charrue     44   charrue     False   False   xxxx    NOUN    NOUN__Gender=Fem|Number=Sing
reversible       52   reversibl           False   False   xxxx    ADJ     ADJ__Number=Sing
```

Figure 13 : Une phrase et son analyse par l'outil de NLP spaCy

Il s'agit ici d'une analyse syntaxique qui permet de lier tous les mots entre eux, et de pouvoir connaître des caractéristiques spécifiques à chacun d'entre eux.

Au delà de ces possibilités syntaxiques, spaCy permet aussi de reconnaître certaines entités telles que des entreprises, des pays ou des groupes politiques lorsqu'ils sont présents dans une phrase. Cette possibilité peut s'avérer intéressante si nous l'appliquons à notre problématique agricole, puisqu'il serait intéressant de pouvoir reconnaître les parcelles ou les procédures dans nos phrases. Et spaCy permet justement de faire cela. Cette librairie nous donne la possibilité de créer notre propre modèle de reconnaissance d'entités. Celui-ci fonctionne grâce à un algorithme de *deep learning*. Nous pouvons ainsi fournir de nombreuses requêtes types à spaCy, en les annotant pour indiquer les différentes entités que celui-ci doit pouvoir retrouver (caractère 5 à 16 -> 'PARCEL' ..), et après avoir parcouru les données d'entraînement, si le jeux de données est suffisamment important, le modèle de reconnaissance d'entités est en mesure de distinguer avec une grande précision les nouvelles entités fournies. A partir de plusieurs centaines de requêtes fournies, le modèle commence à être fiable. Ce modèle peut ensuite être incorporé à Mycroft pour reconnaître les éléments caractéristiques des requêtes de l'utilisateur. Ces entités permettent ensuite le déclenchement des actions adéquates.

Cette solution permettrait de proposer une technologie en constante amélioration. En effet, en commercialisant un produit basé sur ce principe, il

serait possible de récupérer l'ensemble des requêtes n'ayant pas abouti pour pouvoir les ajouter aux phrases-type servant d'exemple à l'algorithme de *deep learning*. Ces phrases seraient alors comprises lorsque le modèle serait remplacé.
Cette méthode permettrait une meilleure compréhension du langage naturel, puisque l'ensemble des formes syntaxiques correspondant à une intention précise pourraient être ajoutées au modèle.

- **ANTLR**

ANTLR pour *ANother Tool for Language Recognition* est un *framework* qui permet la construction de compilateurs pour effectuer des analyse de texte.
Il est nécessaire de fournir à ANTLR une grammaire ainsi que des règles syntaxiques et ANTLR va produire un algorithme permettant de reconnaître les différents éléments définis par les règles venant d'être créées. ANTLR peut générer du code pour plusieurs langages dont Python. Un code généré par ANTLR peut donc être inclus dans un développement effectué pour Mycroft.
ANTLR permet de générer des analyseurs lexicaux ou syntaxiques. Un analyseur syntaxique permet la création d'un arbre syntaxique qui peut à son tour être traité par un analyseur d'arbre.
Ce *parseur* peut trouver de nombreuses utilités. L'exemple suivant a été obtenu après avoir décrit les règles d'un analyseur syntaxique. L'arbre obtenu peut donc ensuite être traité pour sélectionner les entités souhaitées.

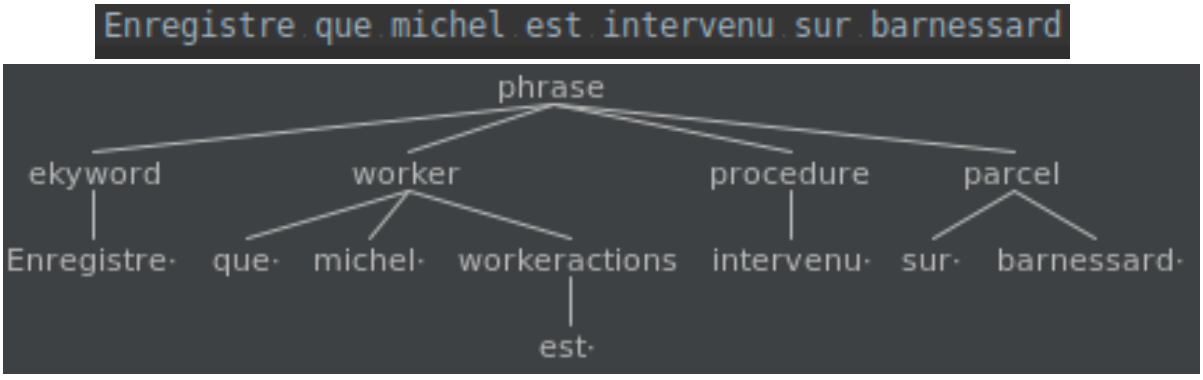


Figure 14 : Arbre issu du traitement d'une requête par le parseur ANTLR

Nous remarquons ici que les entités composant une phrase sont bien reconnues mais étant donné que la majorité des contraintes sont basées sur les expressions régulières (ReGex), il est difficile d'obtenir un modèle fixe valable pour l'ensemble des formulations syntaxiques d'une même phrase.

Phonex

Phonex est un algorithme permettant la conversion d'une chaîne de caractères en un code phonétique. C'est un algorithme spécifique au français, et créé dans le but d'améliorer des algorithmes préalablement existants

(Soundex et Soundex 2), qui sont les précurseurs des algorithmes de conversion du texte en langage phonétique. Les détails algorithmiques de Phonex ont donc été ajustés à certaines spécificités du langage français.
Dans l'exemple ci-dessous, nous obtenons le code phonétique d'une phrase prononcée et de son interprétation par le moteur STT de Bing.

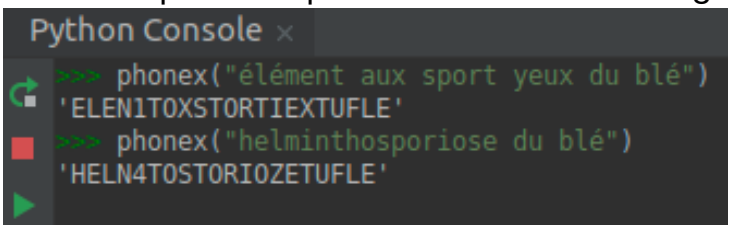


Figure 15 : Comparaison de deux chaînes de caractères résultant de l'interprétation d'un algorithme phonétique.

Nous pouvons par la suite envisager de comparer les codes phonétiques pour ainsi contrer le léger taux d'erreur du moteur STT. La bibliothèque Fuzzywuzzy peut alors être utilisée pour comparer deux chaînes de caractères, comme montré ci-dessous :

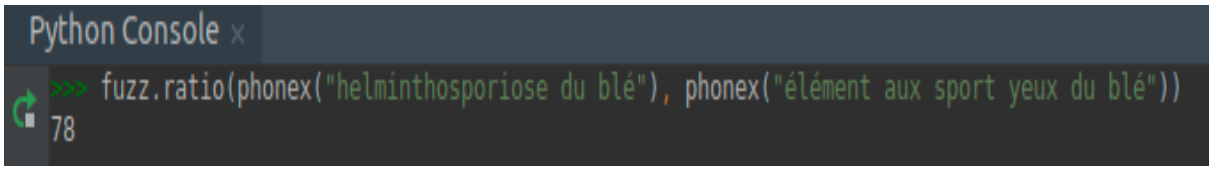


Figure 16: Ratio de ressemblance entre les codes phonétiques de ces deux mêmes phrases

Ici, nous affichons le ratio de ressemblance entre les codes phonétiques de ces deux phrases. Ce ratio est de 78%, ce qui est très important, surtout lorsqu'on le compare au ratio des chaînes de caractères qui est uniquement égal à 43%.
Phonex peut donc nous permettre de mieux reconnaître certains éléments syntaxiques.

3.2: Choix des technologies utilisées

Le choix des technologies peut être décomposé en plusieurs parties : le socle ainsi que les briques à y intégrer (les moteurs vocaux), et les outils de développement s'assimilant à du NLP. Nous commencerons par présenter le socle.

3.2.1: Socle de base de l'assistant vocal

Mycroft a été sélectionné pour être le socle de notre assistant vocal, car comme énoncé précédemment, il s'agit d'une solution libre, offrant une

grande liberté aux développeurs, et compatible avec un très grand nombre d'outils notamment grâce au langage utilisé, à savoir du Python.
Nous pouvons donc commencer à retravailler à partir du premier prototype qui avait été réalisé en 2016.

Cependant, le mode de développement va bien varier, puisque la contrainte principale étant celle de la compréhension du langage naturel, il est nécessaire de ne pas utiliser de mots-clés. Le système de *skills* utilisé précédemment ne pourra donc pas être réutilisé, ou alors, celui-ci se devra d'être remanié.
La capacité de notre assistant à comprendre le langage naturel sera en lien direct avec la qualité de son moteur de reconnaissance vocale. Ceci sera détaillé par la suite.

3.2.2: Moteurs STT et TTS

La rapide étude effectuée a permis d'aboutir à ce tableau comparatif :

Moteur STT / Facteur	Vocabulaire initial	Possibilité de rajouter un lexique	Prix (pour une minute)	Rapidité d'exécution
Google_Cloud	Très bon, peu de mots agricoles inconnus ✔	Impossible ✖	0.024 $ »	Bonne ✔
AWS_Transcribe	Bon ✔	Possible mais partiellement efficace. ✔	0.024 $ »	Mauvaise ✖
IBM_Watson	Mauvais et peu performant pour le lexique agricole ✖	Possible, actuellement peu efficace. »	0.020 $ »	Bonne ✔
DeepSpeechSTT	Mauvais ✖	Ajout de vocabulaire, et entraînement du modèle ✔	Gratuit ✔	Mauvaise ✖
Microsoft Bing	Excellent ✔	Non, création de modèle acoustique »	0.016 $ »	Mauvaise ✖

Figure 17 : Etude comparative de différents moteurs STT disponibles sur le marché

Le moteur de reconnaissance STT ayant le mieux réagi à nos tests est le moteur de Google Cloud. Les principaux défauts qu'il possède sont son accessibilité par API et l'impossibilité d'y ajouter du vocabulaire spécifique. Cependant, la bonne compréhension de leur modèle couplée à la rapidité d'exécution font de lui le meilleur candidat. Le fait que sa disponibilité soit uniquement par API implique de devoir posséder une connexion réseau pour pouvoir l'utiliser.
Le premier prototype sera donc développé à l'aide du moteur STT de Google Cloud.
L'autre alternative aurait été d'utiliser Mozilla Deepspeech, car ce moteur est gratuit et accessible hors ligne. Ces caractéristiques sont des points très importants pour Ekylibre, qui souhaite au possible utiliser des technologies *open-source*. Actuellement la qualité de reconnaissance vocale de Deepspeech est vraiment trop faible pour pouvoir être exploitée, mais grâce à un appel à contribution en continu, leur moteur devrait s'améliorer très prochainement et il est envisageable de pouvoir utiliser cet outil dans la version finale de Duke, ou de pouvoir le changer dans les années à venir.
Le moteur TTS sélectionné est quand à lui Watson d'IBM qui est celui se rapprochant le plus d'une voix humaine et ayant la meilleure vitesse d'exécution.
Après avoir choisi les outils constituant la structure de notre ébauche d'assistant, il était nécessaire de sélectionner les outils de développement, qui seront donc abordés au point suivant.

3.2.3: Natural Language Processing:

Les différents outils permettant de développer des solutions informatisées faisant appel à des fichiers vocaux sont relatifs au *natural language processing*.
Comme présenté précédemment, nous avions plusieurs options qui se sont offertes à nous, et spaCy est l'élément principal qui a été sélectionné.
Il a été décidé d'utiliser spaCy afin de créer un modèle analysant les requêtes des utilisateurs. Ne travaillant pas avec des mots-clés, le principe est donc de pouvoir comprendre l'intention en associant des actions à certains types d'entités pouvant être trouvées dans une requête.
La façon dont nous travaillerons avec spaCy sera plus détaillée par la suite.
Phonex sera aussi utilisé dans un autre cadre. En effet, pour contrer les erreurs du moteur de reconnaissance vocale, comparer l'interprétation de ce moteur à nos valeurs de test apporterait une sécurité supplémentaire.
Par exemple si l'on demande à un utilisateur le type d'intervention qu'il souhaite enregistrer, il est commun que le moteur STT comprenne 'la bourre' au lieu de 'labour'. Dans ce cas de figure, une correspondance classique ne

pourrait pas être trouvée, mais une correspondance phonétique serait immédiatement effectuée.
Ainsi l'algorithme Phonex sera utilisé, en second recours après spaCy et uniquement dans le cas ou l'on demande quelque chose de précis à l'utilisateur pour savoir à quoi les retours contenus dans sa requête doivent être comparés. L'un des mécanismes-type décrivant l'utilisation de Phonex est présenté dans l'exemple suivant.

(ex :
- Duke : Sur quelle parcelle enregistrez-vous l'intervention ?
- *Utilisateur* : Sur l'**enclos des chats et niais**
- Duke : Confirmez vous l'enregistrement d'une intervention sur l'**enclouse des châtaigniers** ?

"L'enclos des chats et niais" n'est pas reconnu comme une parcelle, mais une comparaison phonétique permet de faire le lien avec l'enclouse des châtaigniers.

- *Utilisateur* : Oui
- Duke : C'est noté
)

Phonex et spaCy seront tous deux utilisés pour le développement de la solution interne, et nous rentrerons plus en détail dans leur utilisation au cours de la partie suivante.

3.3 Méthode de développement de la solution interne

La contrainte principale du développement de "Duke" est la compréhension du langage naturel. Pour aboutir à un prototype se rapprochant d'une compréhension parfaite des intentions de l'utilisateur, une analyse des phrases a été mise en place. Elle sera détaillée par la suite.

3.3.1: Création d'un modèle d'analyse de phrases

Ce modèle d'analyse est entraîné avec du *deep learning* et à partir d'une base de phrases-type pour que l'intelligence artificielle puisse en tirer les enseignements nécessaire à une bonne compréhension d'une réelle requête utilisateur. Nous commencerons par décrire le fonctionnement du modèle, puis nous détaillerons la façon dont il a été entraîné.

3.3.1.1: Définition des entités à reconnaître

De façon traditionnelle, spaCy permet une analyse de phrase. Une des fonctionnalités associées permet notamment de reconnaître les entités présentes dans la phrase. Il existe originellement 18 entités que l'algorithme peut reconnaître par lui même.
Un exemple de cette reconnaissance d'entité peut être compris avec la phrase suivante, issue de leur documentation officielle:

" Apple cherche à racheter une startup anglaise pour un milliard de dollars"

Au sein de cette phrase, spaCy va reconnaître que :

- "Apple" est de type "Organisation"
- "anglaise" est en lien avec un "groupe - pays"
- "un milliard de dollars" sont relatifs à l'entité "Money - Argent"

Ces entités ont pu être distinguées du reste de la phrase car le modèle a été entraîné pour les reconnaître. A partir de *corpus* disponibles sur Wikipedia, le modèle a appris à observer ces différents groupes, avec de très nombreux exemples pour chacun d'entre eux.
Actuellement, cette reconnaissance est très aboutie, et spaCy a mis en place un outil permettant aux utilisateurs de créer leurs propres types d'entités et d'entraîner leur modèle personnel pour que ces entités soient elles aussi bien reconnues.
Dans le cadre de Duke, il est donc apparu que créer des entités pour distinguer toutes les fonctionnalités et ainsi comprendre les intentions des requêtes permettrait la compréhension du langage naturel.
Pour commencer, et à partir de la fonctionnalité correspondant à la saisie des interventions, il a donc été décidé de définir quatre entités:

- La procédure -> PROC (correspondant au type d'intervention)
- Le travailleur -> WORKER (celui qui a effectué la tâche)
- La parcelle -> PARCEL
- L'outil -> TOOL

Ces quatre entités nous permettent de repérer l'ensemble des mots nous intéressant dans une requête, soit les mots porteurs d'informations.
Après avoir défini ces entités, il était cependant nécessaire de faire en sorte que notre programme les reconnaisse dans les requêtes des utilisateurs. L'entraînement de notre modèle sera expliqué dans la partie suivante.

3.3.1.2: Entraînement du modèle par deep learning

Pour que le modèle soit capable de reconnaître les entités concernées par les fonctionnalités que nous développons, spaCy propose un algorithme

de *deep learning* qui nécessite un grand nombre de phrases-référence, avec des annotations permettant de situer les mots correspondant à certaines entités.
Ces annotations indiquent le caractère de début d'entité et celui de fin d'entité, ainsi que l'entité attribuée au(x) mot(s).
Les requêtes fournies à EA4T ont pu être utilisées pour créer la base de requêtes destinées à l'apprentissage du modèle.
Plusieurs paramètres sont ensuite choisis, à savoir le nombre d'itérations et le pourcentage de phrases prises en compte pour le modèle à chaque itération.
Un exemple de phrases annotées est observable dans l'extrait de script suivant.

```
("Enregistre que j'ai pressé de la paille de blé sur Bousseau", {
'entities': [(15, 20, 'WORKER'), (20, 26, 'PROC'),(51,59,'PARCEL')]
}),
("Planifie la moisson du grand champ", {
'entities': [(0, 8, 'PLAN'), (12, 19, 'PROC'),(23,34,'PARCEL')]
}),
("J'ai pulvérisé 30 % de la parcelle du poteau edf", {
'entities': [(5, 14, 'PROC'),(26,47,'PARCEL')]
}),
("Yvan a pulvérise la parcelle des pommiers", {
'entities': [(0, 4, 'WORKER'), (7, 16, 'PROC'),(20,41,'PARCEL')]
}),
("J'ai labouré cinquante pour cent du carrefour", {
'entities': [(15, 20, 'WORKER'), (5, 12, 'PROC'),(36,45,'PARCEL')]
}),
```

Figure 18 : exemple de requêtes annotées présentes dans la base d'apprentissage

En lançant les itérations, nous pouvons suivre l'évolution de la valeur de perte.
La **perte** est un nombre qui indique la médiocrité de la prévision du modèle pour un exemple donné. Si la prédiction du modèle est parfaite, la perte est nulle.
Nous pouvons ainsi savoir quand la qualité du modèle commence à stagner pour l'arrêter et éviter un surapprentissage, synonyme de modèle trop ajusté sur les données de référence. La phase de *learning* est observable sur la figure ci-dessous.

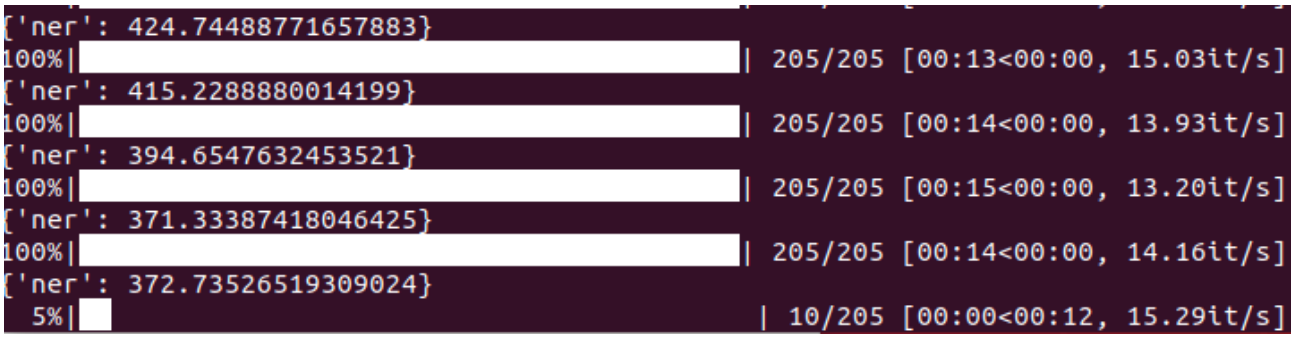


Figure 19 : Itérations de l'algorithme de spaCy et évolution de la valeur de perte.

Ici, la valeur figurant à droite de 'ner' correspond à la perte.
Les avantages de travailler avec du *deep learning* sont multiples. Comme énoncé précédemment, l'utilisation d'entités va permettre de se rapprocher du langage naturel, et même si les premiers modèles ne seront probablement pas parfaitement adaptés aux utilisateurs, il est ensuite très facile de modifier un modèle spaCy avec du learning. Ces modèles pourraient donc être améliorés continuellement, au fur et à mesure que nous remarquons de nouvelles formulations des requêtes utilisateurs.
La base d'apprentissage utilisée correspondait dans un premier temps aux requêtes fournies à EA4T, mais il a été décidé d'automatiser la création de cette base pour que celle-ci puisse être adaptable à tous les utilisateurs, et donc aux données de toutes les fermes.

3.3.2 Automatisation de la génération de la base d'apprentissage

Nous disposions dans un premier temps d'un soixantaine de phrases pour entraîner notre modèle, ce qui est très peu. Il est admis que quelques centaines de phrases forment un bon début de base d'apprentissage. Un autre inconvénient était que ces phrases n'étaient applicables qu'aux données de démo puisque tous les noms de parcelles et d'outils utilisés correspondaient à ceux de la ferme de test. Pour généraliser la création de ces phrases annotées à toutes les fermes d'Ekylibre et pour avoir une base plus conséquente, une automatisation a été suggérée.

Dans un premier temps, toutes les données d'une ferme peuvent être récupérées par l'API d'Ekylibre (les *travailleurs*, parcelles et outils), puis les requêtes initiales peuvent être modifiées pour que les noms de ces données écrites en dur ne le soient plus.
Ainsi une requête sous la forme de : "David a labouré sur Bernessard" deviendra : "Worker a PROC sur PARCEL ".
Nous pouvons alors itérer sur les nouvelles requêtes en y incrustant aléatoirement les données adéquates. Ainsi à partir de soixante phrases-type,

nous pouvons générer plusieurs milliers de phrases différentes permettant d'entraîner notre modèle et étant spécifiques à la ferme en question.

Comme exprimé précédemment, un des verrous technologiques majeurs de la création d'un assistant vocal est la mauvaise compréhension du langage par les outils de reconnaissance vocale. Pour contrer ce problème, il a en plus été décidé d'orienter notre modèle en l'entrainant à la fois avec les mots exacts, et avec leurs transcriptions parfois faussées par les moteurs de reconnaissance vocale.

L'idée suivante a été appliquée :

- Pour chaque donnée faisant partie des entités, sa forme écrite est envoyée par API au moteur de synthèse vocale le plus réaliste (à savoir IBM Watson), puis l'enregistrement vocal est sauvegardé
- Cet enregistrement vocal est ensuite envoyé à notre moteur de reconnaissance vocale, qui nous retourne donc la forme écrite selon sa compréhension
- Cette forme écrite est ensuite ajoutée à la liste des données de la ferme étudiée et sera donc ajoutée aux requêtes-type au même titre que la forme écrite originale.

L'ensemble de ces actions peuvent être résumées dans la figure suivante :

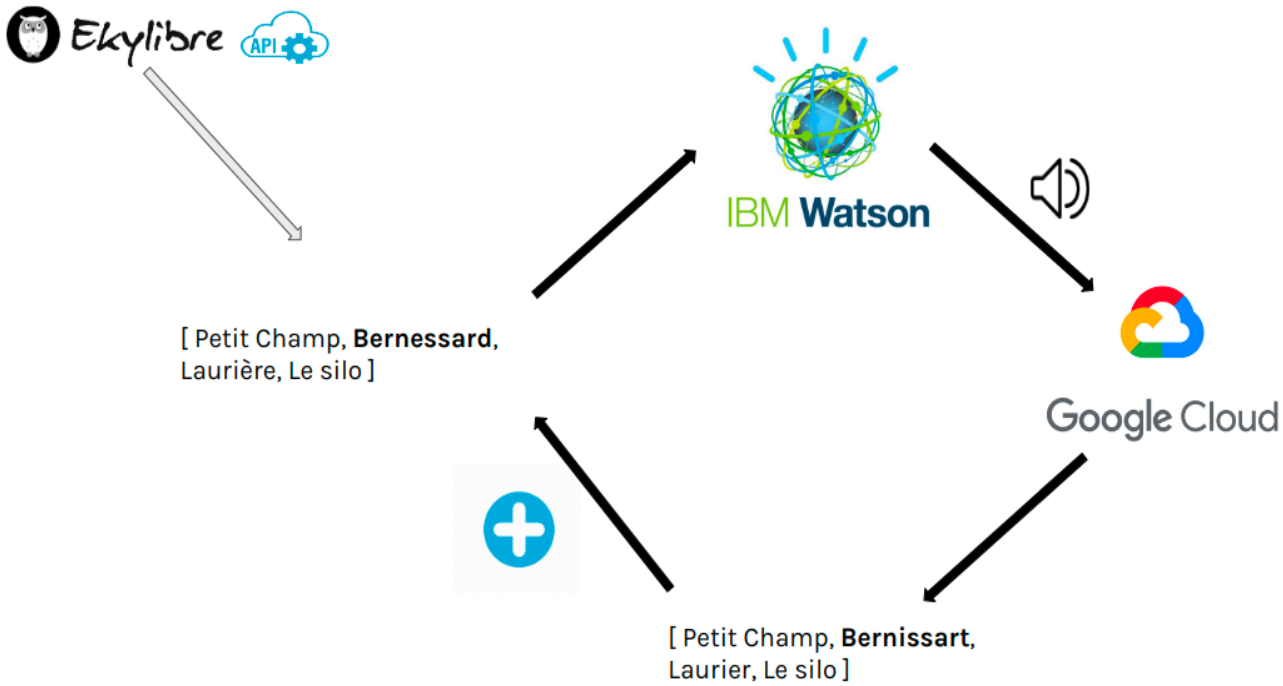


Figure 20 : Pipeline visant à entraîner le modèle avec les transcriptions des données réelles

Dans cet exemple, les mots Bernessard et Laurière ne sont pas compris par le système de reconnaissance vocale, et les mots compris Bernissart et Laurier vont donc être ajoutés à la liste des parcelles. Il est important de noter que si le mot est bien compris par le STT, celui-ci va aussi être ajouté à la

liste. Ainsi dans le cas où le moteur STT comprend parfaitement l'ensemble des données, nous aurons une liste de doublons ne présentant aucun problème pour la traitement qu'il en sera fait par la suite, et si des erreurs sont présentes, nous pouvons alors préparer notre assistant à ce qu'il les rencontre. De plus, nous testons le moteur STT à chaque appel de ce script, donc si le moteur de reconnaissance vocal change ou évolue, notre base d'entraînement suivra son évolution. Une fois ce modèle en place, il est possible de développer les actions de l'assistant.

3.3.3 Développement des actions de l'assistant

Le déclenchement des fonctionnalités sera directement lié à la reconnaissance d'entités au sein d'une requête. Une entité ne correspond pas à un mot en particulier, mais à ce que l'algorithme va reconnaître à partir de son entraînement : de la tournure de la phrase, de la syntaxe et de l'orthographe. Dès la création d'entités, nous ne travaillons plus avec une multitude de *skills*, mais avec un unique *skill*. Ce skill gère donc l'ensemble des fonctionnalités, et une priorisation est effectuée selon les entités reconnues dans la requête.
Les actions basiques correspondent toujours à des *skills* déjà existants créés par la communauté, et le *skill* comprenant l'ensemble des fonctionnalités de saisie peut être assimilé à un "EkylibreSkill" qui est lui considéré comme un *fallback* de Duke. Un *fallback* correspond à la fonctionnalité qui va être appelée dans le cas ou aucun autre *skill* n'est reconnu.

Pour la solution interne réalisée, l'objectif principal est de pouvoir prouver la bonne compréhension des requêtes par notre assistant. Il n'est donc pas nécessaire de développer toutes les subtilités ni de réaliser un programme complet effectuant la totalité des actions énoncées. Ce prototype agit comme une Proof of Concept, permettant surtout de montrer la capacité qu'aurait une solution de ce type à fonctionner. Le réel développement sera présenté dans la dernière partie, mais il est d'abord nécessaire de pouvoir juger de la qualité de notre prototype.

4: La phase de test, instigatrice de la mise en production de Duke

4.1 Etude de notre solution sur le périmètre des interventions

Il aurait été apprécié de pouvoir comparer le prototype réalisé de façon interne avec une ébauche d'assistant réalisé par EA4T, mais l'entreprise n'a pas pu réaliser son prototype assez rapidement pour que cette comparaison

puisse être réalisée. Seul notre prototype sera testé et les résultats pourront par la suite être comparés avec leur solution, lorsque celle-ci sera disponible. Le principal test correspond au taux de compréhension de notre assistant sur les phrases du jeu de test. Nous allons donc commencer par présenter ces résultats.

4.1.1: Essais sur les phrases du jeu de test

Nous avons pu juger de la qualité de compréhension de notre modèle grâce aux phrases du jeu de test. A partir d'une vingtaine de phrases obtenues directement auprès d'agriculteurs, nous pouvions avoir une idée du taux de réussite de notre solution.
Sur vingt-deux phrases prononcées à notre prototype, celui-ci a été en mesure d'en comprendre douze parfaitement. Parfois, la parcelle n'était pas reconnue, et notre prototype redemandait à l'utilisateur de la déclarer. Ce cas de figure était donc anticipé, et un fonctionnement de la sorte est acceptable. De même, certains outils n'étaient parfois pas bien reconnus, mais dans le cadre de cette étude nous ne considérerons pas cela comme un échec, car les outils sont des éléments secondaires des requêtes utilisateur. Les deux types d'occurrences détaillées ci-dessus ont donc été considérées comme des succès.
Nous pouvons alors arriver au nombre de dix-sept phrases bien reconnues par notre moteur de reconnaissance vocale, puisque dans trois requêtes, seul l'outil n'avait pas été compris, et dans deux requêtes, la parcelle ne l'avait pas été.

Le taux de réussite apparent serait alors de 78%, ce qui est tout à fait honorable. D'autant plus que parmi les phrases non comprises, certaines sont basées sur des éléments implicites, et aucun travail n'avait été fait pour la compréhension implicite d'éléments. Notre solution ne pouvait pas tenter de bien les comprendre.
Un exemple de phrases non reconnues par notre prototype est la suivante :

"Eh Duke, je suis passé avec le semoir à grains sur le champ guillon petit"

Ici, nous observons qu'aucun type d'intervention ne peut être reconnu, et notre assistant ne peut donc pas déterminer que l'utilisateur souhaite enregistrer une procédure. Cependant, la présence d'un outil "semoir à grains" implique forcément qu'il s'agit d'une procédure de type semis.

Ces éléments nécessitent un gros travail sur la structure des données, et la transmission de cette structure sous la forme d'ontologie à notre assistant. EA4T travaille justement à partir d'ontologies, et grâce à toutes les données fournies, il devrait pouvoir faire les liens entre un élément implicite présent dans une requête, et l'interprétation à en faire.

Cette procédure pourra éventuellement être comprise par le prototype d'EA4T, et nous jugerons de cette éventualité lorsque celui-ci sera disponible.
Cependant, la qualité de compréhension du modèle n'est pas le seul critère permettant de juger de la qualité d'une solution, et d'autres métriques avaient été mises en place. Nous continuerons donc en détaillant les autres résultats provenant de ce test.

4.1.2: Autres métriques et synthèse

Parmi les autres métriques de test ayant été définies précédemment, nous avons pu mesurer la vitesse de traitement moyen de l'information par notre assistant. Avec une connexion wi-fi acceptable, possédant un débit d'envoi moyen de 0.632 Mb/s et un débit de réception moyen de 6.217 Mb/s, le temps moyen de réponse suite à une requête utilisateur est de 2.7 secondes.
Lorsque la connexion internet diminue, le résultat est sensiblement similaire. Il semble en effet que les étapes les plus longues soient la reconnaissance vocale et la synthèse du vocal, or ces étapes soient indépendantes de notre connexion internet, puisque faites directement sur les serveurs d'IBM et de Google.
Ce temps de réponse est acceptable, même si on peut considérer qu'au-delà d'une seconde, cela peut être légèrement dérangeant pour maintenir le fil d'une discussion. Ce temps de réponse a été obtenu avec le moteur TTS d'IBM Watson, et il est intéressant de noter qu'en réalisant la même expérience avec le moteur de Google ayant enregistré l'ensemble des phrases en cache, cette valeur diminuerait sous la seconde. Cette valeur serait très satisfaisante, et bien que la qualité de la voix ait été favorisée jusqu'ici, nous pourrions changer de TTS pour une exécution plus rapide.
La seconde métrique correspond aux taux de fonctionnement hors ligne. Dans le cas de ce prototype, celui-ci se doit d'être connecté à internet pour fonctionner, donc son taux de fonctionnement hors ligne est de 0%.
Concernant la facilité de déploiement d'un assistant créé selon la méthode explicitée précédemment, celui-ci serait très simple à réaliser. L'ensemble des missions pourrait être effectué par Ekylibre, et nous pourrions le déployer sur tous les supports envisagés. Bien que nous ne puissions pas comparer notre prototype à celui d'EA4T, nous pouvons d'ores et déjà annoncer que le déploiement sera forcément bien plus compliqué à mettre en place dans le cas où nous sous-traiterions la création d'un assistant, car l'ensemble des travaux seront effectués à partir des infrastructures en place chez ces tiers.

A partir de toutes les métriques déclarées précédemment, nous avons pu juger de la qualité de ce prototype. Nous pouvons annoncer que la qualité de sa compréhension semble actuellement acceptable sur le petit périmètre des interventions. La vitesse de réaction et la facilité du déploiement sont aussi

des points encourageants de ce prototype. Le désavantage majeur provient du fait que celui-ci ne fonctionne pas du tout hors ligne.
Après avoir pu juger de la qualité du prototype, il est important de s'intéresser à la suite du projet, une fois que toutes les solutions techniques utilisées auront été fixées.

4.2: Prérequis au développement final

Après avoir sélectionné les acteurs ainsi que les technologies pour la création de notre assistant vocal, la phase de développement peut être envisagée.
Dans le cas de Duke, le développement des fonctionnalités passe tout d'abord par deux étapes préalables :

- La rédaction des spécifications fonctionnelles.
- Le choix du mode d'intégration.

La rédaction des spécifications fonctionnelles va permettre aux personnes chargées de développer le projet de mieux comprendre les besoins et les détails liés à chacune des fonctionnalités. Ce livrable sera indispensable dans tous les cas, que Duke soit conçu au sein d'Ekylibre, ou par une autre entreprise n'ayant pas d'expertise agronomique. Ce document sera détaillé dans la suite de ce mémoire.

4.2.1: Rédaction des spécifications de développement

Les spécifications de développement ont été rédigées pour l'ensemble des fonctionnalités. Comme précédemment, nous traiterons uniquement l'exemple de la saisie d'intervention.
L'extrait ci-dessous permet donc de décrire la fonctionnalité de saisie d'intervention, avec les différentes actions possibles et les cas d'erreur qu'un utilisateur doit rencontrer.

Description

Cette fonctionnalité permet à un agriculteur d'enregistrer l'ensemble des interventions qu'il a pu réaliser sur son exploitation. Ces interventions correspondent à des opérations sur des cultures, sur des animaux d'élevage ou à certaines actions liées à la traçabilité de la production de l'utilisateur.

Actions possibles

Un utilisateur doit pouvoir enregistrer une intervention, directement avec un travailleur, un outil, une parcelle et le type de procédure.

Si aucun travailleur n'est reconnu, l'intervention est pré-sauvegardée et le travailleur associé à cette intervention est l'utilisateur.

Si aucun outil n'est reconnu, l'intervention est pré-sauvegardée sans outils.

Une intervention pré-sauvegardée doit ensuite être validée par l'utilisateur. Un récapitulatif est donné par Duke et l'utilisateur doit pouvoir confirmer ou annuler l'intervention. Des modifications ne peuvent plus être effectuées.

- *Cas d'erreur*

Si la procédure n'est pas reconnue, l'ensemble des autres paramètres reconnus doivent être pré-enregistrés et la procédure doit être redemandée. Si à nouveau, aucune procédure n'est reconnue, l'intervention est annulée.

Si aucune parcelle n'est reconnue, l'intervention ne peut pas être enregistrée et il est nécessaire de demander à l'utilisateur le nom de la parcelle où l'intervention a lieu.

Si, à nouveau, aucune parcelle n'est reconnue, l'intervention est annulée.

A partir de ces spécifications, un développeur peut facilement ajouter une fonctionnalité à l'assistant vocal. La question se posant alors est le mode d'intégration de Duke.

4.2.2: Le mode d'intégration de Duke

Comme décrite précédemment, l'intégration de Duke doit pouvoir être réalisée sur différents types de plateformes. L'intégration sur un site web, semblait être une bonne option, et dans le cas de notre prototype créé sur Mycroft, la communauté a déjà travaillé à la réalisation d'une interface permettant d'intégrer un assistant à un site web sous la forme d'un *bot* conversationnel. La capture d'écran ci-dessous montre l'apparence d'une telle interface.

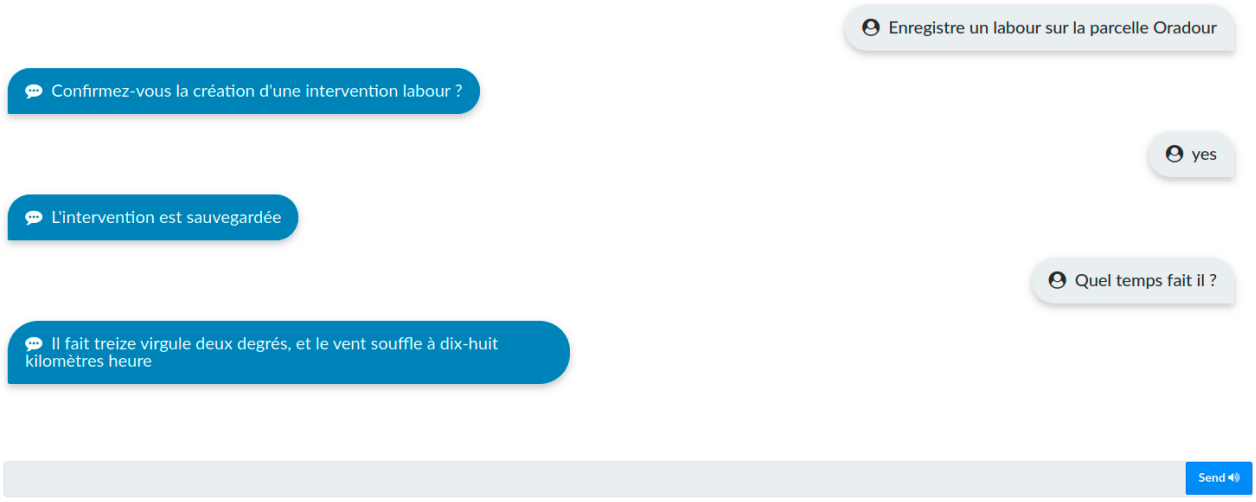


Figure 21 : Interface web de Duke sur Mycroft

Cette interface est très rudimentaire, mais intégrer un agent conversationnel de ce type au site web d'Ekylibre semble possible. Une telle interface pourrait être activée sur le site d'Ekylibre pour un utilisateur, lorsqu'une fonctionnalité de saisie est disponible. Dans le cas ou notre prototype serait mis en production, la mise en service des fonctionnalités peut être progressive pour obtenir les premiers retours sur la fonctionnalité mise en place, ainsi que toutes les phrases n'ayant pas été reconnues et ainsi pouvoir améliorer le modèle général pour cette fonctionnalité, ou pour l'ensemble de la méthode.

Dans le cas ou EA4T finirait par réaliser l'assistant, ils pourraient nous proposer des interfaces déjà développées pour les intégrer aux architectures d'Ekylibre, mais la responsabilité de l'intégration reviendrait à Ekylibre.

En parallèle de la phase préliminaire au développement, il est tout de même nécessaire d'envisager la suite du projet, en se penchant notamment sur les actions à mener une fois qu'Ekylibre possédera un prototype fonctionnel.

4.3: Perspectives

Une fois la première solution fonctionnelle créée, plusieurs autres problématiques vont apparaître. L'objectif final étant la commercialisation de Duke, deux nouvelles étapes interviennent. Celles-ci sont relatives au choix du mode de commercialisation de Duke, qui sera évoqué par la suite, et à une phase de test en condition réelle dont l'objectif sera d'évaluer notre solution, et de pouvoir y déceler quelques erreurs non observables dans des bureaux. Nous allons donc d'abord aborder cette phase de test.

4.3.1: Phase de test

Afin de tester le prototype réalisé, il est important de réaliser des tests en conditions réelles. Il faudra donc se déplacer dans des fermes pour déceler d'éventuels problèmes liés à l'environnement d'une exploitation agricole. Ekylibre pourra à nouveau faire appel aux ekytesteurs ayant déjà répondu aux premiers sondages pour qu'ils soient les premiers à avoir accès à cette innovation.

En parallèle, des étudiants de l'ENSAT vont participer à un projet tutoré pour tester Duke dès septembre si un prototype fonctionnel peut leur être fourni. Ce projet est réalisé dans le cadre de leurs études, et il sont intéressés par les enjeux relatifs au projet Duke. Leur objectif serait donc d'avoir un grand nombre de retours sur l'utilisation par les agriculteurs d'un outil vocal, et de déterminer l'ensemble des problèmes qu'ils pourraient rencontrer. Tous ces éléments pourraient ensuite nous être remontés pour que nous puissions ajuster notre prototype aux besoins du monde agricole. Une autre partie du travail que ce groupe d'étudiants pourrait réaliser concerne le mode de commercialisation de Duke, et un début de traitement de cet aspect sera effectué dans la partie suivante.

4.3.2: Le marché, et le mode de commercialisation de Duke.

Dès qu'un prototype suffisamment fonctionnel sera disponible, Duke sera proposé à l'ensemble des clients d'Ekylibre. Duke ne sera pas proposé gratuitement, bien que l'ensemble du projet sera proposé à la communauté *open-source* de Mycroft, et différentes solutions sont envisageables pour le commercialiser. Les voici :

- Proposer Duke à la vente.

La vente de Duke est une alternative qui peut faire du sens, notamment si nous proposons Duke sur une enceinte intelligente. Nous pouvons alors vendre l'ensemble et nous assurer que les utilisateurs possèdent un micro de qualité suffisante. Cette alternative pourrait cependant poser problème car si nous sous-traitons la création de l'assistant vocal, nous paierons probablement EA4T à la requête. Le même problème a lieu avec l'utilisation de Mycroft car les moteurs STT et TTS utilisés sont actuellement propriétaires et payants. Ainsi cela sera très rentable pour Ekylibre mais uniquement à court terme.

- Proposer une facturation à la requête.

Pour parer au problème évoqué précédemment, il est possible de facturer les utilisateurs selon le nombre de requêtes qu'ils font à partir de

Duke. Cette solution nous permettrait d'être rentables dans tous les cas de figure, mais risque d'être très mal accueillie par le monde agricole

- Proposer Duke au sein d'un abonnement

La dernière alternative consiste à proposer Duke au sein d'un abonnement. Le business model d'Ekylibre est déjà basé sur des abonnements, et un abonnement premium comprenant de nouvelles fonctionnalités sera bientôt disponible. Les différents abonnements proposés par Ekylibre sont décrits dans la figure ci-dessous.

Figure 22 : Les différents abonnements proposés par Ekylibre

Il serait donc possible d'intégrer Duke dans Ekylibre +, pour justifier de sa valeur ajoutée en termes de gain de temps.
Après avoir déterminé les perspectives du projet, pour le bon suivi du projet après la phase de développement, nous pouvons désormais aborder la conclusion de ce mémoire.

Conclusion

Le travail effectué au cours de ce stage avait pour finalité de faire progresser un projet de recherche et développement : Duke.
L'objectif principal était de pouvoir choisir le mode de développement d'un assistant vocal destiné à l'agriculture. Deux méthodes de développement avaient été retenues en vue d'une comparaison pour déterminer la manière la plus adaptée aux besoins du monde agricole. Il s'agissait ainsi de choisir entre le développement interne d'une solution d'assistant vocal à l'aide d'outils majoritairement libres, et le sous-traitement de cette création à une entreprise spécialisée dans ce secteur du numérique.

Comme cela avait été suggéré par l'analyse des besoins réalisée auprès d'agriculteurs partenaires, la contrainte principale qu'il a fallu respecter était celle de la compréhension du langage naturel. Après six mois de travail, une solution technique a été édifiée avec des résultats sur un petit périmètre d'étude pouvant être jugés convenables.
Cette solution, faisant intervenir des librairies libres de traitements syntaxiques, a permis d'aboutir à un modèle respectant la contrainte principale citée précédemment. Cette solution a été développée pour Mycroft, un socle *open-source* crée pour la démocratisation des assistants vocaux en lien avec le respect de la vie privée de l'utilisateur. Mycroft permet à des développeurs de créer des fonctionnalités en Python, un langage très généralisé et adaptables à tous les types de plateformes sur lesquelles il est encore envisagé de déployer Duke. Cette solution présente de nombreux avantages comme la facilité de développement, de déploiement, le prix très faible voire nul ou bien la possibilité d'une évolution constante au fur et à mesure que les utilisateurs l'utiliseront.

La comparaison avec l'autre solution retenue n'a cependant pas pu être effectuée car l'entreprise responsable de la production de l'autre prototype n'a pas pu répondre dans les temps. L'ensemble des éléments et livrables permettant d'effectuer une comparaison rigoureuse sont disponibles et cela sera donc la prochaine étape pour permettre l'avancement du projet Duke.
Il sera par la suite nécessaire d'effectuer des tests en conditions réelles dans des exploitations agricoles avant d'envisager une mise en production. Ces tests auront la responsabilité de déceler d'autres types d'erreurs que celles détectées durant la phase de préproduction. Les dernières questions à se poser concernent alors la commercialisation de Duke, et le choix de la formule de vente devra être décidé selon les indications des utilisateurs.

Bibliographie

L. Anavi. «Comparison of Voice Assistant SDKs for Embedded Linux», 2018, *p. 33-43*.

R. Jiang, R. E. Banchs, et H. Li, «Evaluating and Combining Name Entity Recognition Systems», 2016, *p. 21-27*.

G. Lample, M. Ballesteros, S. Subramanian et al. «Neural Architectures for Named Entity Recognition», 2016, *p. 7-8*

B. Lauser, M. Sini, G. Salokhe et al. «Agrovoc Web Services-Improved, real-time access to an agricultural thesaurus», 2006, *IAALD Quarterly Bulletin, p. 79-81*.

C. Mcdonald. «A short introduction to NLP in python with spaCy», 2017, <https://towardsdatascience.com/a-short-introduction-to-nlp-in-python-with-spacy-d0aa819af3ad>, [Consulté le 28/05/2019].

A. Muchtler. «A short history of the voice revolution», 2017, <https://voicebot.ai/2017/07/14/timeline-voice-assistants-short-history-voice-revolution/>, [Consulté le 14/06/2019].

P. Pestanes et B. Gautier, «Essor des assistants vocaux intelligents : nouveau gadget pour votre salon ou fenêtre d'opportunité pour rebattre les cartes de l'économie du web ?», 2017, *p. 2-5*.

P. Pringuet, J.-M. Bournigal, F. Houllier, et al. «Pour une agriculture compétitive et respectueuse de l'environnement», 2015, *p. 20-24*.

F. Seide, G. Li, et D. Yu, «Conversational Speech Transcription Using Context-Dependent Deep Neural Networks», 2011, *p. 2-4*.

J. Tejedor,R. Garcia , M. Fernández *et al.*, «Ontology-Based Retrieval of Human Speech», 2007.

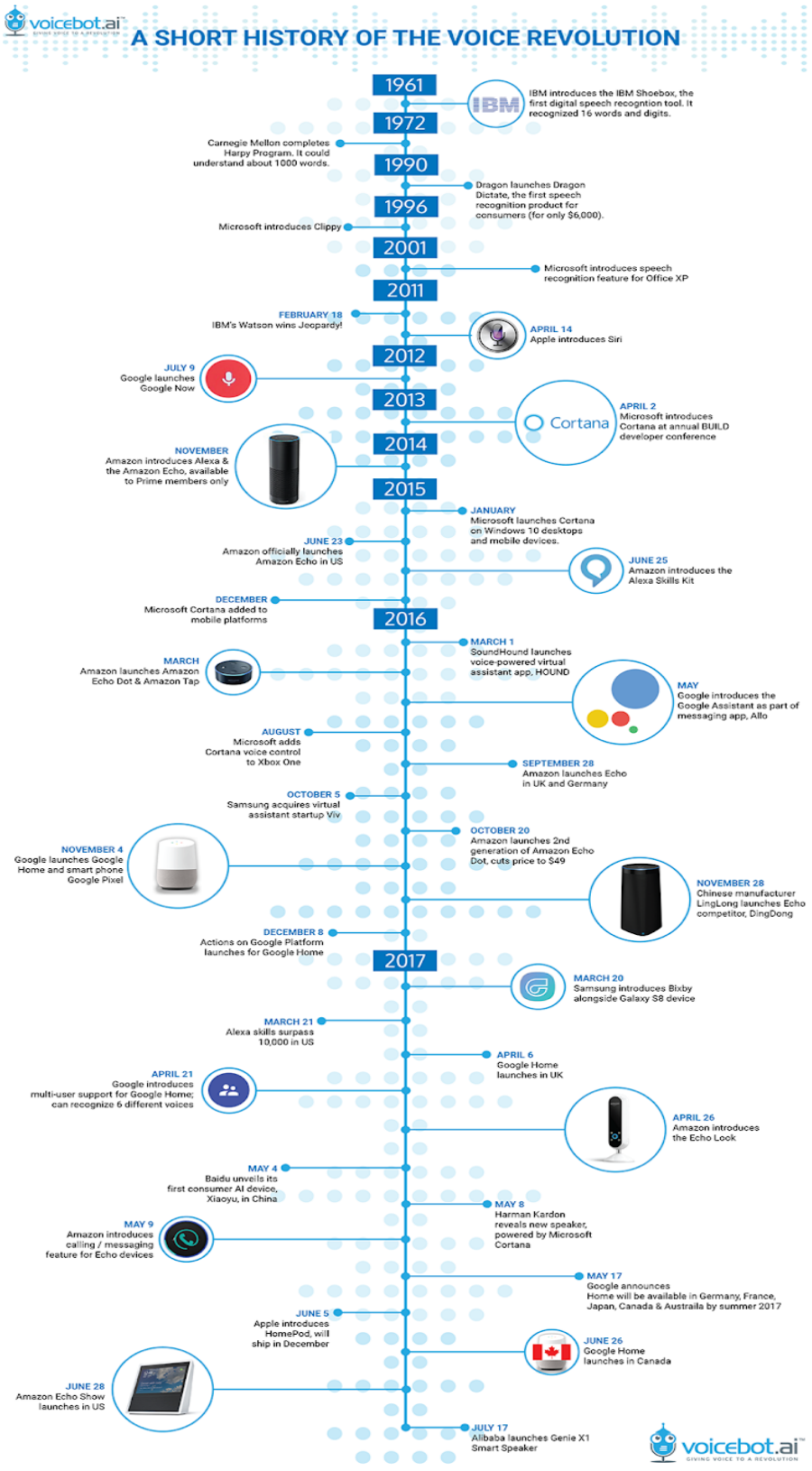
P. Tena. « Conception d'un serious game sur le thème de la gestion des exploitations agricoles », 2017, p.12-14. Mémoire de fin d'études : Bordeaux : Bordeaux Sciences Agro ; Ekylibre : 2017

Liste des annexes :

Annexe i : Historique récent de l'avancée des technologies vocales (source : A. Muchtler. «A short history of the voice revolution»

Annexe ii : Extrait du questionnaire proposé aux ekytesteurs et sur le forum d'ekylibre

Annexe i : Historique récent de l'avancée des technologies vocales (source : A. Muchtler. «A short history of the voice revolution»)



Annexe ii : Extrait du questionnaire proposé aux ekytesteurs et sur le forum d'ekylibre.com

---

Classez les éléments de saisie suivants selon leur pénibilité/durée

	1. rapide / pas désagréable	2. niveau modéré	3. très long / désagréable
Enregistrer les incidents	○	○	○
Enregistrer les interventions	○	○	○
Enregistrer les achats	○	○	○
Enregistrer les ventes	○	○	○
Gérer les stocks	○	○	○
Saisir des observations terra...	○	○	○
Gérer la trésorerie	○	○	○

Au sein des catégories ci-dessus, citez un/des exemple(s) de tâche(s) pesante(s) ?

Réponse longue

Quels (autres) éléments de saisie souhaiteriez vous voir facilités ?

Réponse longue

Quels freins voyez vous à l'utilisation d'un assistant vocal ?

Réponse longue

Résumé :

Évaluation et Intégration de solutions techniques pour la création d'un assistant vocal.

Ekylibre est un éditeur de logiciel à destination du monde agricole. Cette entreprise innovante tente actuellement de faciliter l'interface entre l'homme et la machine pour permettre une meilleure accessibilité à ses logiciels, et notamment à son logiciel ERP éponyme. Afin de réaliser cet objectif, la voix est perçue comme l'interface d'avenir, et l'objectif est de permettre à l'utilisateur d'utiliser la voix pour réaliser une grande partie des actions disponibles dans l'ERP Ekylibre.

Dans le cadre de ce projet, la méthodologie de travail conduisant à l'évaluation et l'intégration de solutions techniques en vue de la création d'un assistant vocal sera détaillée. Les besoins ont d'abord été recueillis, avant de réaliser un état des lieux des solutions disponibles pour la création d'un assistant vocal répondant aux contraintes déterminées. Un premier prototype a été réalisé par Ekylibre, à l'aide d'outils *open-source*. Ce prototype permet une compréhension acceptable du langage naturel dans le domaine agricole.
La solution créée devra être confrontée avec une solution proposée par une entreprise spécialisée dans la création de ce type d'outils afin de définir les technologies utilisées, pour permettre la mise en production de "Duke", l'assistant vocal lié à la solution d'Ekylibre. Cette mise en production entraînera une phase de développement, et un questionnement sur le mode de commercialisation de "Duke".

Mots clés : Duke, assistant vocal, Natural Language Processing, gestion agricole, reconnaissance vocale

Abstract:

Assessment and integration of technical solutions to create a vocal assistant

Ekylibre is a software editor dedicated to the agricultural world. This innovant company is actually trying to facilitate the interface between the man and the machine to allow a better accessibility to it's software, and especially it's eponym ERP. In order to achieve this goal, the voice has been perceived as the interface of the future, and the purpose will be to enable the voice as a way to perform most of the functionalities available in the Ekylibre ERP.

In this project, the work methodology that lead to the assessment and the integration of technical solutions to create a vocal assistant will be discussed. Firsty, the needs have been collected, and a benchmark of all available solutions that can help create a vocal assistant, which can work within the defined constraints, has been done. A first prototype was created by Ekylibre, mostly using open-source tools. This prototype allows a correct comprehension of the natural language in the agricultural field.

This prototype has to be compared with a solution which will be suggested by a company which is specialized in creating vocal assistants. This comparison will help fix the technologies to use, to help put "Duke", Ekylibre's vocal assistant in production. This production will demand a development phase, and a questionment on Duke's commercialisation.

Keywords ***:*** Duke, Vocal assistant, Natural Language Processing,farm management, voice recognition