

< TITRE DE VOTRE MÉMOIRE DE PROJET DE SEMESTRE >



< Insérez ici votre illustration >

(non obligatoire)

Projet de semestre présenté par

< Prénom NOM >

Informatique et systèmes de communication avec orientation en
< Nom complet de votre orientation >

< Mois, année >

Professeur-e HES responsable

Mandant (si existant)

< Prénom Nom >

< Prénom Nom >

Légende et source de l'illustration de couverture :

TABLE DES MATIÈRES

<i>Remerciements (style « Titre 1 »)</i>	<i>vi</i>
<i>RÉsumÉ (style « Titre 1 »)</i>	<i>vii</i>
<i>Liste des acronymes.....</i>	<i>viii</i>
<i>Liste des illustrations.....</i>	<i>ix</i>
<i>Liste des tableaux.....</i>	<i>xi</i>
<i>Liste des annexes.....</i>	<i>12</i>
<i>Introduction</i>	<i>13</i>
<i>Chapitre 1 : Définition de Chatbot</i>	<i>16</i>
1.1. Un Chatbot, qu’ est-ce ?.....	16
1.2. Trois composants d’ un Chatbot	17
a) L’ interface.....	17
b) Le moteur de traitement de données	17
c) Le moteur de réponses	18
1.3. trois types de Chatbots.....	18
a) Les Chatbots de Menus.....	18
b) Les Chatbots de règles.....	21
c) Les Chatbots “intelligents”	23
<i>Chapitre 2 : Étude des solutions existantes.....</i>	<i>28</i>
2.1. Agenda.ch.....	28
2.2. Klara.ch.....	29
2.3. Meetme.io	30
2.4. Comparaison à ce projet	30
<i>Chapitre 3 : Détail technologique.....</i>	<i>32</i>
3.1. Natural Language Processing	32
a) Natural Language Understanding.....	37
b) Natural Language Generation	38
3.2. TAPAS.....	41
a) Transformers	42
b) BERT	43
c) TAPAS : Fonctionnement.....	47
<i>Chapitre 4 : Étalage des technologies existantes</i>	<i>51</i>
4.1. Chatbots par NLU	51
a) Les intents	51
b) Les Entités	52
c) Les Stories	52
d) Les Actions	53
e) Les Règles.....	53
f) Les Slots	53

g)	Les Lookup Tables	54
h)	Les Forms.....	54
4.2.	Rasa.....	54
a)	Rasa X	55
b)	Rasa Open Source	58
c)	Rasa Pro	60
d)	Duckling	62
4.3.	Chatterbot	63
4.4.	Microsoft Bot Framework.....	64
4.5.	NLTK	65
4.6.	OpenNLP	65
Chapitre 5 : Présentation des prototypes.....		66
5.1.	Prototypes NLP	66
a)	Java.....	66
b)	Python	67
5.2.	Prototype TAPAS	69
a)	Données de test	70
b)	Résultats	71
c)	Conclusion	71
5.3.	Prototype Chatbot	72
a)	Données d' entraînement	73
b)	Exemple de conversation	75
c)	Conclusion	75
Chapitre 6 : Choix technologiques		77
Conclusion (style « Titre 1 »)		77
Annexes (style « Titre 1 »).....		78
Annexe 1.....		79
Annexe 2.....		80
Annexe 3.....		81
RÉfÉrences documentaires (style « Titre 1 »)		82

NB : La table des matières est à actualiser par le menu contextuel « Mettre à jour l'index ».

Nom, Prénom – Titre abrégé – Projet de semestre – Mois, Année

< Insérez ici votre dédicace > (facultatif)

REMERCIEMENTS (STYLE « TITRE 1 »)

< Formulez ici vos remerciements aux personnes qui vous ont aidé dans la réalisation de votre travail. Les remerciements sont rédigés dans le style « Corps de texte » >.

NB : les Titres « Titre 1 non numérotés » sont basés sur le titre 1 mais sont en dehors de la numérotation des chapitres. Ils apparaissent néanmoins dans la table des matières

RÉSUMÉ (STYLE « TITRE 1 »)

< Insérez ici votre illustration >

(obligatoire)

Candidat-e :

Professeur-e(s) responsable(s) :

< NOM PRENOM >

< NOM PRENOM >

Filière d'études : ISC

En collaboration avec : < Nom de l'entreprise >

Travail de semestre soumis à une convention de stage en entreprise : <oui/non>

Travail soumis à un contrat de confidentialité : <oui/non>

Attention : Tout le résumé doit tenir sur une seule page

LISTE DES ACRONYMES

NLP Natural Language Processing

NLU Natural Language Understanding

NLG Natural Language Generation

I.A. Intelligence Artificielle

UX User Experience

POS Part-of-Speech

NER Named Entity Recognition

LLM Large Language Model

TAPAS TABLE PArSing

SDK Software Development Kit

CALM Conversational AI with Language

API Application Programming Interface

LUIS Language Understanding Intelligent Service

LISTE DES ILLUSTRATIONS

Illustration 1: Graphique de la relation entre les trois composants de base	17
Illustration 2: Graphique d'un Chatbot de Menus	19
Illustration 3: Arbre de décision d'un Chatbot de Menus.....	20
Illustration 4: Graphique d'un Chatbot de règles	21
Illustration 5: Exemple de sélection de règle pour Chatbot par règles	22
Illustration 6: Graphique d'un Chatbot Intelligent.....	24
Illustration 7: Pyramide de la difficulté technologique et UX selon le type de Chatbot	27
Illustration 8: Séquence de traitement NLP	33
Illustration 9: Représentation de la relation NLP, NLU et NLG.....	37
Illustration 10: Cas exemples du moteur google utilisant BERT	44
Illustration 11: Représentation de l'entrée du modèle BERT.....	46
Illustration 12: Entrée du modèle TAPAS.....	47
Illustration 13: Schéma de fonctionnement du modèle TAPAS.....	48
Illustration 14: Interface contenant le menu de Rasa X	55
Illustration 15: Page de gestion des modèles avec Rasa X	57
Illustration 16: Example de conversation parmis l'historique des conversations.....	57
Illustration 17: Architecture Rasa.....	59

Références des URL

- URL01 <https://www.ovationc xm.com/blog/3-kinds-chatbots-youll-meet>
- URL02 <https://www.javatpoint.com/nlp>
- URL03 <https://datasolut.com/natural-language-processing-vs-nlu-vs-nlg-unterschiede-funktionen-und-beispiele/>
- URL04 <https://www.youtube.com/watch?v=ioGry-89gqE>
- URL05 https://humboldt-wi.github.io/blog/research/information_systems_1920/bert_blog_post
- URL06 <http://arxiv.org/abs/2004.02349>

URL07 <https://www.datocms-assets.com/30881/1608730961-screenshot-2019-05-21-at-12-46-37.png>

URL08 <https://www.datocms-assets.com/30881/1608730956-screenshot-2019-05-21-at-12-44-11.png>

URL09 <https://www.datocms-assets.com/30881/1608730966-screenshot-2019-05-21-at-12-48-34.png>

URL10 <https://rasa.com/docs/rasa/arch-overview>

LISTE DES TABLEAUX

Tableau 1: Représentation du tableau fourni en entrée de TAPAS	47
Tableau 2: Tableau de dimensions de Duckling	63
Tableau 3: Extrait de Terrains.csv	70
Tableau 4: Contenu de Horaires.csv.....	70

Références des URL

URL01 <https://github.com/facebook/duckling>

LISTE DES ANNEXES

Annexe 1	79
Annexe 2	80
Annexe 3	81

INTRODUCTION

De nos jours et comme il a été le cas par le passé, que ce soit pour aller au médecin ou au restaurant ; Il est commun de placer des réservations. Or pour ce faire il est d'usage de passer un coup de fil ou dans d'autres cas de placer la réservation en se rendant au dit lieu du rendez-vous. Aujourd'hui, il est possible de placer ses réservations directement depuis de nombreux sites web. Qu'ils soient possédés par le particulier chez qui on souhaite placer rendez-vous ou par une autre entité chargée de gérer l'aspect de placement de réservations et/ou rendez-vous. Ces sites web permettant de prendre rendez-vous se présentent très régulièrement sous la forme de boutons cliquables et de quelques champs à remplir telle que la date du rendez-vous, le nom/prénom et autres informations. Cependant, il est bien plus rare qu'une alternative à cela existe et ne nécessitant pas de devoir se rendre sur une page internet quelconque. Le projet qui sera réalisé a pour optique d'apporter un autre moyen de placer une réservation se trouvant à l'intersection d'un appel téléphonique et un simple formulaire cliquable : Un Chatbot servant à prendre des rendez-vous depuis des applications de messageries telles que What's app ou Telegram ou alors pouvant être aisément inclus dans un site web existant et qui permettra de rendre plus fluide le processus de prise de rendez-vous.

Ce Chatbot s'apparente plus au premier abord à un composant applicatif auquel il est possible de fournir du texte et en recevoir en sortie. Bien que le but premier soit de fournir un Chatbot avec lequel il sera possible de converser sur une application de messagerie, il est tout à fait possible de mettre en place une section Chatbot sur un site quelconque et d'y lier le Chatbot, le rendant de ce fait facile à mettre en place sans avoir à se soucier de l'interface graphique de l'unique nécessité du Chatbot étant un champ textuel mis à disposition à l'utilisateur.

Une des raisons primaires de ma décision de réaliser ce projet étant qu'à présent, en 2024, de nombreux progrès ont été réalisés dans le secteur du Machine Learning et de

l’I.A. et bien que de nombreuses compagnies ont commencé à doucement mais sûrement à implémenter des technologies diverses en termes d’I.A. ou Machine Learning, il y a encore un grand nombre de secteurs dans lesquels la présence de ces dernières se fait encore discrète ou simplement car peu ont considéré faire usage de ces technologies dans ces derniers. Le secteur en question présentement est celui du Service Client et plus spécifiquement l’assistance à réservation et prise de rendez-vous. Je trouve tout particulièrement pertinent l’exploration de cette voie qu’est l’usage de ces technologies pour la réalisation d’un Chatbot pour prendre rendez-vous car moi-même devant de temps à autre prendre rendez-vous, il est bien difficile pour bon nombre de gens de passer des coups de fil, moi inclus. C’est pour cela que fournir une solution qui soit modulaire et ne nécessitant pas d’architecture particulière que simplement un moyen d’incorporer une boîte de dialogue ou champ textuel quel que soit la plateforme choisie faciliterait grandement son utilisation par les particuliers de tout secteurs et qui fournirait une alternative viable aux appels et sites web.

Ce rapport précède le travail de Bachelor Informatique et Systèmes de Communications à L’Haute Ecole du Paysage, d’Ingénierie et d’Architecture de Genève et a pour objectif d’expliciter les divers concepts proéminents dans le projet, l’étude des divers technologies existantes pouvant servir à la réalisation de ce projet, l’explicitation des choix technologiques découlant de cette étude, de montrer des exemples concrets de leur utilisation à l’aide de quelques prototypes et de finalement étaler le plan de conception du travail allant être réalisé.

Actuellement, la réalisation de ce rapport s’est effectuée de manière régulière chaque semaines lorsque le temps me le permettait et un état des lieux fut régulièrement effectué à l’aide d’entretiens hebdomadaires avec M. Niklaus Eggenberg. Afin de réaliser ce rapport, l’outil Git a servi à versionner et stocker les divers prototypes réalisés pour tester les technologies ainsi que le journal de bord tenu à jour régulièrement pour faire compte du travail effectué à chaque session de travail. Le stockage s’est effectué sur la plateforme

Gitlab mise à disposition par la HES et sur un projet Git possédé par M. Niklaus Eggenberg. L'outil Google Docs fut d'une grande aide lors de l'écriture d'ébauches, de prises de notes, d'explications préemptives des divers technologies recherchées et pour la formulation de mes idées concernant le travail à réaliser par la suite.

Ce rapport va par conséquent servir de fondation à la réalisation prochaine du Chatbot. Concernant les divers section abordées dans ce rapport, elles se comptent au nombre de sept chapitres **À VOIR EN FONCTION DE L'EVOLUTION DE LA REDACTION (CE QUI SUIT N'EST PAS FINAL)**. Le premier définira de manière concrète et conceptuelle ce qu'est un Chatbot. Le deuxième portera sur l'étude des solutions existantes de réservation et prises de rendez-vous se trouvant sur le marché et comment elles comparent à celle proposée ici. Le troisième sera une énonciation et explicitation des divers concepts piliers composant les divers structures de Chatbot énoncées au chapitre précédent. Le quatrième a pour but de faire un état des nombreuses technologies existantes et utiles pour la réalisation de ce projet. Le cinquième chapitre me servira à présenter les quelques prototypes que j'ai pu réaliser de certaines des technologies énoncées précédemment ainsi que mes réflexions les concernant. Le sixième a pour simple but de définir dans quelle direction le projet ira structurellement parlant et quelles technologies seront usées pour sa réalisation. Et finalement, le septième élaborera dans le détail toute la partie de planification et conceptualisation du projet.

Chapitre 1 : DEFINITION DE CHATBOT

Avant même de commencer à parler technologies, il est important d'expliciter concrètement ce qu'est un Chatbot.

1.1. UN CHATBOT, QU'EST-CE ?

Les Chatbots sont des applications disponibles sous une multitude de formes et plateformes et ayant pour but premier d'engager dans une conversation avec un utilisateur afin de tenter au mieux de répondre à quelque requête que ce soit. Les Chatbots font affaire à un grand nombre de requêtes variées comme fournir des renseignements, fournir un service ne nécessitant pas d'opérateur humain, faire office de service client afin de tenter de résoudre tout éventuel problème dans la limite des capacités du Chatbot (Le cas échouant, le bot peut rediriger l'utilisateur vers un opérateur humain), récolter du feedback, et bien d'autres usages encore. Bien que des Chatbots existent dans un nombre incalculable de formes et structures, il y a néanmoins une manière de décomposer n'importe quel bot en une composition de trois éléments distincts

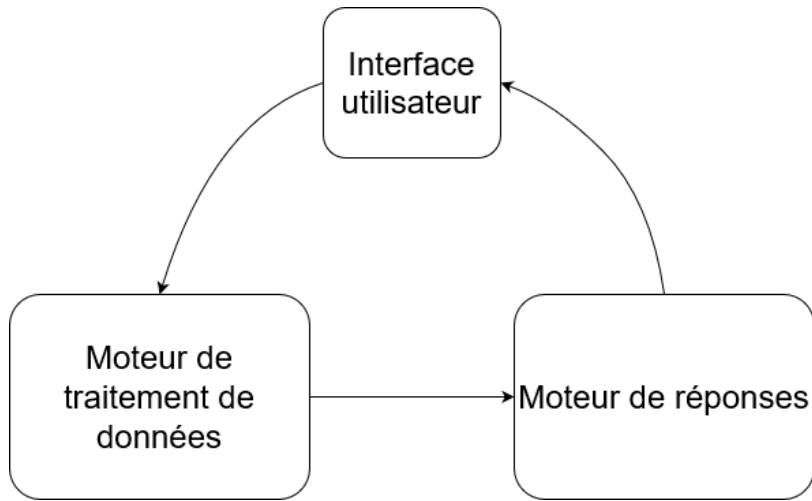


Illustration 1: Graphique de la relation entre les trois composants de base

Source : Rodrigues dos Santos Fabio

1.2. TROIS COMPOSANTS D’UN CHATBOT

a) L’INTERFACE

Aucun Chatbot ne pourrait avoir la dénomination de *Chatbot* s’il ne possédait pas d’interface avec laquelle l’utilisateur peut interagir. Au plus simple, une interface pourrait être une simple ligne de commande de terminal et au plus poussé, une fenêtre de dialogue ayant l’apparence d’un logiciel de messagerie comme sur nos smartphones. L’essentiel ici est que l’interface mise à disposition comporte au moins une zone de saisie textuelle ou autre support de communication tel que des boutons cliquables par exemple. C’est au travers de ce support que l’utilisateur fera parvenir ses diverses requêtes au bot pour traitement ultérieur.

b) LE MOTEUR DE TRAITEMENT DE DONNÉES

Une fois que l’utilisateur a exprimé sa demande au travers d’un des canaux de communications disponibles, cette demande sous forme de texte va être immédiatement acheminée au moteur de traitement de donnée. Les requêtes utilisateurs dans le cas où le

canal de communication est un champs de texte se présentent sous la forme de suites de caractères. Ce qui est tout à fait compréhensible pour un humain mais pas pour une machine, c'est donc pourquoi un traitement se doit d'être appliqué au texte utilisateur. Le texte se présentant sous formes de phrases sera verra le plus souvent mis sous la forme de mots-clés ou instructions pouvant eux être compris par la machine.

c) LE MOTEUR DE RÉPONSES

Une des dernières étapes du cycle d'échange entre l'utilisateur et le bot est le passage des données précédemment traitées dans le moteur de réponse. Son rôle est de faire sens des divers mots-clés et/ou instructions qu'il a reçues et d'envoyer à l'utilisateur la réponse correspondant le plus à sa requête.

Ces trois composants, une fois mis ensemble forment un Chatbot fonctionnel capable de recevoir des requêtes d'utilisateurs, de les rendre compréhensibles par le système et d'en sortir la réponse adéquate à l'utilisateur et ce ainsi de suite jusqu'à ce que l'échange prenne fin.

1.3. TROIS TYPES DE CHATBOTS

Ce simple cycle d'échange de données expliqué ci-dessus permet à présent de diviser cette fois les Chatbots en trois catégories distinctes étant les variantes de Chatbots les plus communes.

a) LES CHATBOTS DE MENUS

Une des formes les plus basiques de Chatbots qui existe, les Chatbots fonctionnant à l'aide d'un menu.

Les Chatbots de Menus, bien qu'étant factuellement très simplistes ne sont pour autant pas mis de côté du fait de leur aspect simpliste et parfois préféré par rapport à d'autres types de Chatbots. En effet, un Chatbot de Menus permet d'éviter tout soucis qui pourraient advenir lors de l'implémentation des autres types de Chatbots plus bas. Grâce à leur canal de communication on ne peut plus facile à utiliser pour l'utilisateur : Des boutons. L'intégralité de l'échange est réalisée grâce aux pressions successives des multiples boutons s'affichant à l'écran de l'utilisateur et ne nécessitant aucune autre forme d'interaction de ce dernier.

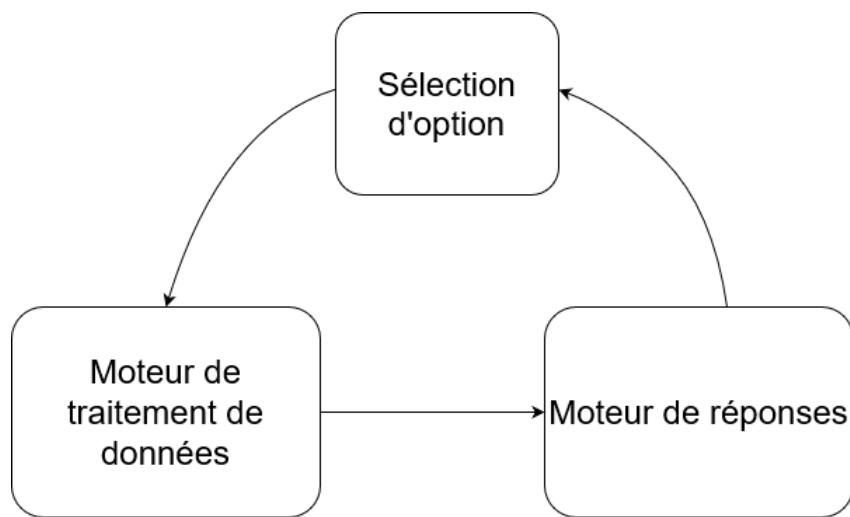


Illustration 2: Graphique d'un Chatbot de Menus

Source : Rodrigues dos Santos Fabio

Ce type de bots est notamment très prisé dans des scénarios où les requêtes pouvant être effectuées par l'utilisateur sont prédéfinies comme pour un système de commande de plats en ligne, un système de support qui permet d'imiter un système similaire existant déjà au format téléphonique étant la demande à l'utilisateur de presser des boutons sur leur téléphone pour spécifier quel type d'aide ce dernier nécessite; ici représenté sous la forme de boutons et qui permet ensuite de donner des informations à l'opérateur humain avant même qu'il n'ait à demander quel type de soucis le client rencontre ou encore lors de placement de réservations.

La séquence de pression de boutons peut être représentée sous la forme d'un arbre de décision. Comme le moyen d'interaction usuel de ce bot se représente sous la forme de boutons cliquables, le système impose un cadre prédéfini ainsi qu'un nombre limité d'interactions possibles et prohibe toute sortie de ce cadre. En effet, à chaque pression de bouton, l'utilisateur prends un des nombreux chemins existants dans cet arbre de décision jusqu'à arriver (ou non) à une réponse convenable et renvoyé à l'utilisateur au travers du moteur de réponses.

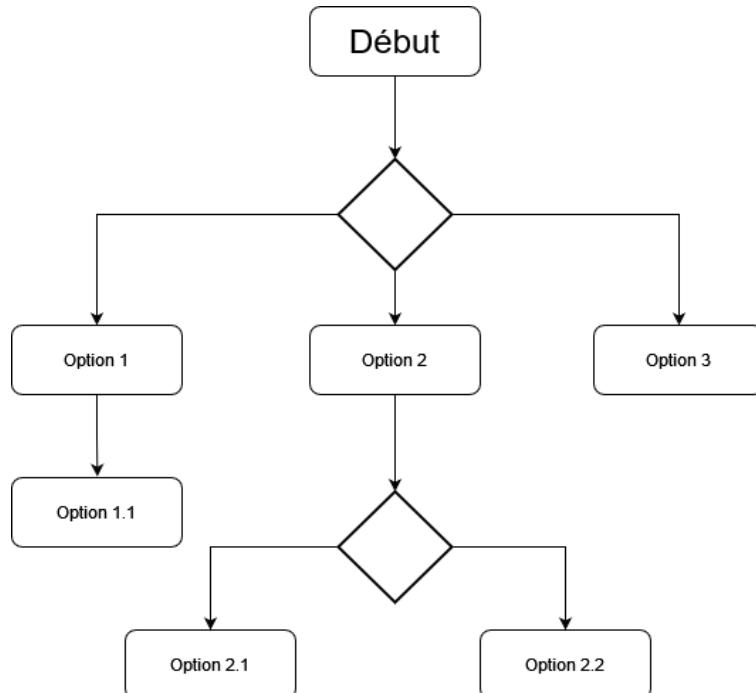


Illustration 3: Arbre de décision d'un Chatbot de Menus

Source : Rodrigues dos Santos Fabio

Un des inconvénients possibles de ce Bot est en termes de scalabilité. Plus l'on souhaite étendre le panel de décisions possibles plus il faudra adapter l'affichage en conséquence et entre autres, un panel trop large pourrait être un désavantage plus qu'un avantage car bien que donner le plus de choix que possible à l'utilisateur semble être optimal, en avoir trop et donc obliger l'utilisateur à cliquer sur une trop grande quantité de boutons peut ruiner l'expérience utilisateur.

Un autre inconvénient possible est que bien que dans certains cas, il est davantage souhaité d'avoir un cadre fixe, il y en a d'autant plus d'autres où l'utilisateur souhaite effectuer une requête qui ne correspond à aucune des requêtes existantes ce qui obligerait soit à créer une très grande quantité de variations dans l'arbre de décision et qui n'arrangerait pas plus le problème qu'il ne l'aggraverait.

b) **LES CHATBOTS DE RÈGLES**

Se trouvant un cran au-dessus niveau complexité, se trouvent les Chatbots de Règles. Ces bots ajoutent un degré de liberté qu'il n'est pas aussi aisément trouvable avec un Chatbot de menus grâce à l'utilisation de règles.

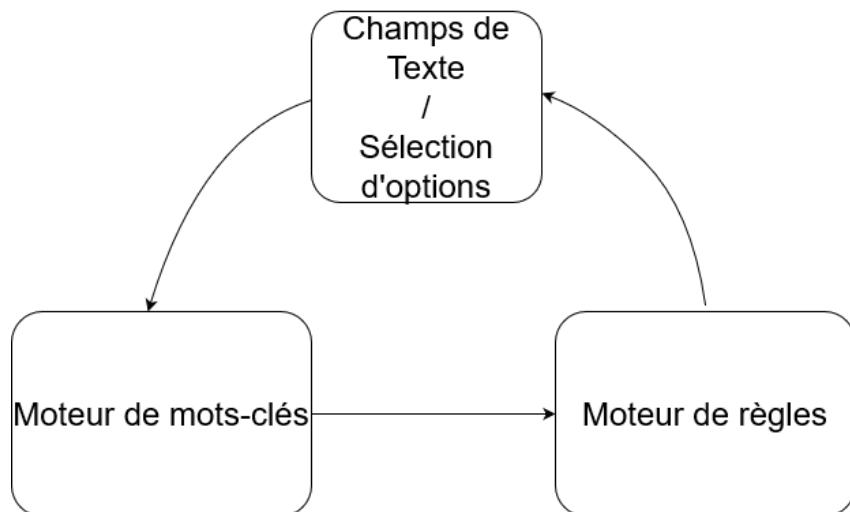


Illustration 4: Graphique d'un Chatbot de règles

Source : Rodrigues dos Santos Fabio

Ce type de bot est presque aussi populaire que ceux à base de menus car tout comme eux, ils répondent à des demandes similaires. Les interfaces souvent rencontrées avec ce genre de Chatbots sont soit des boutons ou cette fois-ci des champs de texte. Il est important de noter la distinction entre les Chatbots de menus et de règles, car l'un possède un chemin prédéfini avec des options se succédant, alors que l'autre possède un ensemble de règles qui selon les attributs ressortis de la requête utilisateur, la règle la plus adéquate sera sélectionnée.

Un exemple pour le cas avec boutons serait qu'au lieu qu'un bouton ne fasse simplement avancer dans l'arbre de décision, que chaque boutons soit pondéré et que même avec une succession de pressions de boutons différentes on puisse arriver à la même réponse.

Dans le cas où l'entrée utilisateur est un texte, la méthode la plus facile et répandue est le simple parage de mots-clés dits « Tokens ». Selon les tokens ayant été ressortis de la requête, une règle comprenant un certain nombre de tokens correspondant ou tous se verra sélectionnée.

Ceci est une phrase **d'exemple**, bien qu'elle paraisse importante



Illustration 5: Exemple de sélection de règle pour Chatbot par règles

Source : Rodrigues dos Santos Fabio

Il existe deux approches possibles à la tokenisation du texte

La première consistant à faire usage d'un dictionnaire de mots-clés établi au préalable. Lorsqu'une entrée utilisateur se voit traitée par le moteur de mots-clés, ce dernier ira simplement vérifier si le token est présent dans le dictionnaire. Si c'est le cas, alors il est ajouté à la suite de tokens qui sera envoyée au moteur de réponse.

La deuxième, elle, consiste à faire usage du NLP (Natural Language Processing) qui permet de faciliter davantage l'implémentation du dictionnaire. Car dans la première approche, une limitation qui pourrait rapidement se faire ressentir est pour commencer le fait que pour chaque tokens ajoutés dans le dictionnaire, il faudra les associer à des règles et donc plus il y a de tokens, davantage il faudra créer d'associations règles-tokens. De plus, pour chaque token il peut exister une infinité de variations à cause de fautes

d'orthographe ou tokens similaires en écriture et/ou sens ce qu'une simple comparaison mot à mot ne pourrait pas détecter correctement.

En d'autres termes, le problème sous-jacent à une implémentation si simpliste est un de scalabilité du projet. Pour un petit projet, ce n'est pas un grand soucis mais avec le temps cela pourrait vite devenir ingérable. C'est donc pour cette raison qu'il est judicieux de faire usage d'un moteur NLP si le projet commence à prendre de l'ampleur pour que la première approche soit viable car ce dernier permet d'éviter de devoir par exemple y inscrire toutes les variations d'un mot grâce à l'usage de divers techniques de NLP voire de regrouper des familles de mots sous un seul type de token. De ce fait, la taille du dictionnaire final se verra grandement réduite et donc même avec un nombre de règles, l'application reste bien plus maintenable et scalable.

Les inconvénients à prendre en compte venant avec l'utilisation d'un Chatbot de règles est que bien qu'il permette un plus grand degré de liberté avec l'usage de champs textes et par conséquent de tokens qui élargissent le panel de possibilités à disposition de l'utilisateur en termes de requête, on reste tout de même dans un cadre restreint car tous les comportements sortant du moteur de règles doivent être implémentés au préalable et continuellement mis à jour au besoin. Ce qui donne l'illusion à l'utilisateur qu'il peut poser quelque requête que ce soit mais il se verra vite confronté à une quantité de réponses limitées s'il sort trop du cadre initial par inadvertance.

c) LES CHATBOTS “INTELLIGENTS”

Le dernier type de Chatbot étant à la fois le plus complexe et pouvant apporter le plus de qualité en termes d'expérience utilisateur sont les Chatbots dit « Intelligents ».

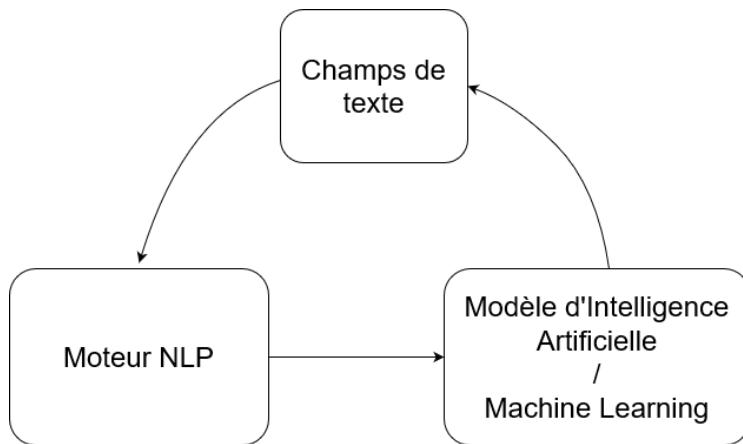


Illustration 6: Graphique d'un Chatbot Intelligent

Source : Rodrigues dos Santos Fabio

Les Chatbots intelligents permettent de pallier le problème commun aux deux types de Chatbots précédent étant le contexte restreint. Dans le cas du Chatbot de règles, il a bien été soulevé qu'ajouter de nouveaux tokens et règles rends le problème exponentiellement complexe dû à l'infinité des tournures que peut prendre une phrase. Or, un Chatbot intelligent est capable d'outrepasser ces limitations par sa capacité à comprendre le sens des mots, le contexte dans lequel ils sont employés et d'engager dans une conversation avec un utilisateur en se rappelant des informations que ce dernier a pu fournir tout au long de la discussion. Étant donné la grande variété de modèles d'intelligence artificielle, il est difficile de donner un exemple d'implémentation spécifique. Le point commun entre tous cependant est qu'ils ont tous pour but de simuler une discussion fluide et dynamique avec l'utilisateur, comme s'ils discutaient avec un agent humain.

Ce type de Chatbot est souvent retrouvé actuellement dans des assistants virtuels tels qu'Alexa, Google Assistant, Siri, Amazon Echo et bien d'autres. Il en existe aussi sous la forme de Chatbots web comme le très populaire ChatGPT, Google Bard, Bing AI qui lui est une intelligence artificielle de type Générative qui a pour but de générer du nouveau contenu à partir d'un contenu existant. Ce qui, bien que cela est intéressant, n'est pas le focus du projet ici.

Cependant, un aspect partagé par toutes ces I.A. est que dans tous leurs traitements de requêtes utilisateurs est l'usage de NLU (Natural Language Understanding). Selon le graphique présenté plus haut, une fois que l'application reçoit la requête utilisateur sous forme de texte, elle subit généralement un prétraitement grâce à l'utilisation d'un moteur NLP. Ce prétraitement peut être plus ou moins utile selon le modèle d'I.A. ou de Machine Learning utilisé car certains modèles comme des I.A. conversationnelles sont programmée pour se charger de toute la partie de Tokenisation et autres traitements NLP avant d'appliquer un traitement NLU afin de comprendre le contexte, intention et sens des phrases et mots présents dans la requête. Le tout étant finalement traité par le modèle correspondant et sa réponse renvoyée à l'utilisateur.

Il est évident qu'un Chatbot de ce type apporte une grande plus-value à l'expérience utilisateur car ce dernier donne l'impression de comprendre quelque requête que ce soit et de rendre l'expérience bien plus personnelle qu'un Chatbot avec règles ou menus pourrait fournir. Hélas, cela ne s'obtient pas aussi aisément car tout comme le moteur NLP qui nécessite une certaine quantité de données pour être entraîné et efficace, il faut une vaste quantité de données d'entraînement si l'on souhaite avoir un Chatbot capable de gérer toute situation se présentant ou alors une quantité assez large et variée pour au moins gérer la plupart des situations dans le cadre d'un projet de moins grande envergure. Cela implique qu'il n'est pas impossible de mettre en place un Chatbot intelligent pour un petit projet mais qu'il faut nécessairement assez de données à disposition. C'est pourquoi un prétraitement par un moteur NLP contribue à nécessiter le moins de données que possible si l'entrée utilisateur peut être transformée en quelque chose de plus général et moins propice à être des cas uniques non pris en compte.

L'essentiel à retirer de ces types de Chatbots se présente ainsi :

- Les Chatbots de Menus sont simplistes tant niveau interface que dans la manière de les utiliser. Ils permettent facilement d'arriver à une réponse en peu d'entrées utilisateur, la complexité d'implémentation étant basse les rend attractif selon le besoin, surtout si le cadre dans lequel il est utilisé est déjà un cadre restreint. Cependant, l'expérience n'est pas très personnelle et ne permet pas de sortir du cadre imposé par le menu lorsque la réponse à la requête utilisateur pourrait ne pas se trouver dans le cadre imposé par l'application. Très utilisé quand l'on sait d'avance que l'utilisateur ne peut pas sortir du cadre imposé et que l'on souhaite avoir un Chatbot nécessitant le moins de maintenance possible.
- Les Chatbots de règles permettent de réaliser des exploits similaires à ceux d'un Chatbot de Menus mais en permettant une plus grande liberté côté utilisateur avec la possibilité de mettre un champs de texte à sa disposition. En plus d'une liberté accrue, l'expérience fournie sera plus personnelle car l'utilisateur peut formuler par ses propres mots sa requête. Or, un soucis de cadre existant persiste car ce qui définit le cadre est la quantité de règles et mots-clés gérés par le Chatbot. S'il n'y a pas assez de règles ou mots-clés pour couvrir tous les cas d'usage, cela empêterait sur l'expérience utilisateur et faire en sorte qu'aucune solution viable ne soit trouvée.
- Les Chatbots intelligents sont très prisés par beaucoup en ce moment de par leur avantages attractifs étant la personnalisation de chaque discussions, de la liberté donnée à l'utilisateur en matière de requêtes tout en gardant un cadre et que le Chatbot tentera d'y ramener l'utilisateur s'il s'en écarte trop lors d'un échange, la facilité à implémenter un système dynamique sans avoir à se préoccuper de tout l'aspect de compréhension par NLU et traitement du texte par NLP et de n'avoir qu'à donner la sortie de ces traitements au modèle pré-entraîné qui trouvera la réponse adéquate à la requête utilisateur. Mais pour y

parvenir le plus grand obstacle reste toujours les données utilisées lors de l'entraînement des divers composants du Chatbot qui impactent grandement sa qualité. S'il n'y en a pas assez ou qu'elles ne soient pas assez variées pour coller aux spécificités souhaitées, le bot pourrait peiner en premier lieu à comprendre l'entrée utilisateur et par la suite à trouver la solution adéquate. De plus, l'usage de modèles d'I.A. ou de Machine Learning nécessitent un plus grand niveau de compréhension de ces derniers afin de déterminer quels modèles et implémentations sont les plus adéquates aux cas d'usages du projet.

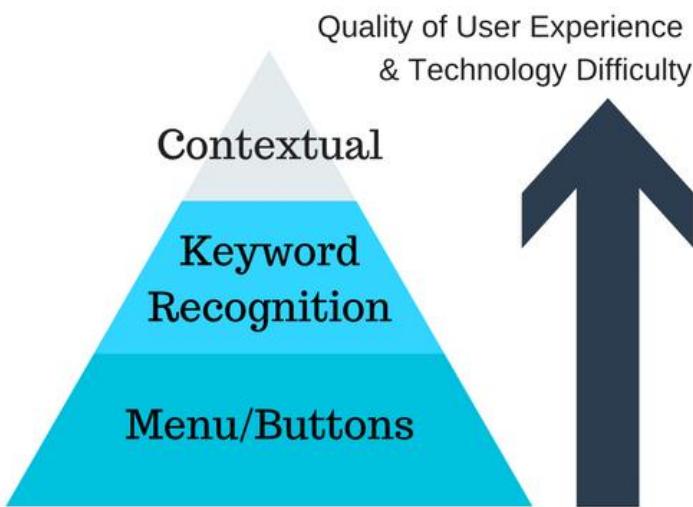


Illustration 7: Pyramide de la difficulté technologique et UX selon le type de Chatbot

Source : Ovationcsm, réf. : URL01

L'illustration ci-dessus démontre de manière visuelle la relation entre la difficulté de la technologie elle-même dans la création de Chatbots par Menus, par Règles ou Intelligents. Il est important de noter que le choix final du type de Chatbot souhaité ne dépend pas uniquement du niveau de qualité de l'expérience utilisateur fournie car sinon seul les Chatbots intelligents seraient utilisés mais la complexité d'implémentation, l'étendue des possibles interactions (tant larges que plus restreintes), ressources à disposition et autres paramètres sont à prendre en compte afin de choisir le Chatbot répondant au mieux aux attentes du projet.

Chapitre 2 : ÉTUDE DES SOLUTIONS EXISTANTES

2.1. AGENDA.CH

Agenda.ch est une application de réservation de rendez-vous Suisse basée à Genève et lancée en 2011. Agenda.ch présente actuellement 5 services dédiés et 1 personnalisable à la demande. Les services à disposition sont :

1. Application dédiée aux physiothérapeutes
2. Application dédiée aux thérapeutes et autres services médicaux
3. Application dédiée aux centres de beauté et Bien-être
4. Application dédiée aux centres sportifs et loisirs
5. Application à visée d'administration publique

Fondamentalement, Agenda.ch fourni une application qui peut s'apparenter à un Chatbot de Menus. Peu importe la forme, elle permet de réserver des ressources étant une salle de sport, une heure pour un rendez-vous médical, une séance pour une coupe de cheveux, etc., de choisir son créneau horaire et finalement tout autre information complémentaire.

L'aspect Chatbot n'est pas le seul utilitaire fourni par cette entreprise :

- L'application n'est pas fournie sous forme d'un module à intégrer ou d'une API à appeler mais elle est disponible chez les serveurs loués par Agenda.ch et tout l'aspect de réservation est mis en place chez Agenda.ch et non chez le client.
- Selon le type de service fourni, il est possible de stocker une multitude de fichiers pertinente à la réservation des clients sur les serveurs d'Agenda.ch
- Selon le type de service fourni, un grand panel de fonctionnalités sont disponibles.

- La possibilité de configurer des messages envoyés automatiquement aux clients pour des rappels de réservations, informations, newsletter, etc.

Ce qui ressort de l'analyse de l'application d'Agenda.ch est qu'elle ne demande aucune implémentation par le client lui-même et fourni l'ensemble de ses services depuis leur site web que ce soit la configuration ou l'utilisation même du système de réservations. L'application est donc détachée du site web du client ce qui peut être vu comme un avantage étant donné que cela facilite son usage car il ne suffit que d'intégrer un simple bouton « réserver » sur le site.

Si l'on se penche sur les tarifs, on remarque qu'il y en a trois. Un à 35 CHF/mois qui n'offre pratiquement que le système de calendrier et de rappel aux clients d'une réservation. Ce n'est qu'à partir de 60 CHF/mois que l'on obtient les fonctionnalités telles qu'un mini-site permettant aux clients d'effectuer leurs réservations dessus ainsi que d'avantages de fonctionnalités en ligne pour faciliter l'aspect réservation. Le dernier tarif étant à 80 CHF/mois propose principalement des systèmes de gestion de documents et ressources diverses directement effectuées par Agenda.ch, pratique si l'on ne possède pas de grande infrastructure ou que l'on ne souhaite pas avoir à se préoccuper d'en gérer une soi-même. Il existe également des modules supplémentaires ou agendas supplémentaires selon des coûts additionnels variables.

2.2. KLARA.CH

Klara.ch est une entreprise Suisse fondée en 2016 basé à Berne et a pour but de fournir un panel varié d'outils d'administration. Contrairement à la solution précédente, Klara.ch ne se spécialise pas uniquement dans la mise en place d'un système de réservations mais d'une panoplie d'outils d'administration comme : Des gestionnaires de client, gestionnaires de budget, gestionnaires d'inventaire, création de shop en ligne, mise en place d'un système de réservations et bien d'autres. L'application de réservation en ligne ainsi que les autres fonctionnent sous forme de widgets soit des modules applicatifs que le

client intègre directement à son site web. La configuration de ce dernier se fait directement au travers du panneau de configuration Klara. Le système de réservation se présente sous la forme d'un Chatbot de Menus, similaire à celui d'Agenda.ch. Le tarif pour le widget de réservation en ligne est de 39 CHF/mois.

La solution de Klara.ch semble intéressante de par son coût réduit par rapport à Agenda.ch et permet d'avoir l'ensemble des fonctionnalités disponibles sur le site client et ne nécessitant pas de devoir naviguer sur une autre page internet. Cependant le degré de customisation et de fonctionnalités supplémentaires spécifiques à certains services ne semble pas comparable à celui d'Agenda.ch, expliquant ainsi son coût plus élevé.

2.3. MEETME.IO

Meetme.io est une application de réservation de rendez-vous Suisse basée à Lausanne et ayant pour objectif de fournir un service de réservation en ligne et de gestion d'agenda grâce à une application ou depuis un panneau d'informations. Meetme.io possède un catalogue de fonctionnalités bien plus réduit comparé aux solutions précédentes ne fournissant uniquement un système de formulaire similaire à celui d'Agenda.ch dans le sens où il n'est accessible uniquement par un lien qui amène le client sur le site de meetme.io à l'url du formulaire de réservation du service souhaité et un système de rappel par sms. Similairement aux solutions précédentes, le formulaire de Meetme est comparable à un Chatbot de Menus. Le tarif pour avoir une quantité raisonnable réservation disponibles s'élève à 69 CHF/mois.

2.4. COMPARAISON A CE PROJET

Après analyse de quelques solutions Suisses disponibles sur le marché actuellement, il est clair qu'il y a une tendance à réaliser un système de réservation par internet. Tous partagent le point commun que le système fonctionne sous la forme d'un formulaire étant

ici l'équivalent d'un Chatbot de Menus. Bien que certaines solutions offrent parfois plus de fonctionnalités que simplement un système de réservation ou proposent des systèmes de réservation avec des fonctionnalités étendues et spécifiques à certains secteurs, aucun ne propose de solution par téléphone uniquement. C'est donc ici que va se placer la solution que je vais proposer étant donné qu'elle permettra de placer des réservations depuis un simple échange par texte avec un Chatbot par téléphone et d'ensuite éventuellement de récupérer facilement l'ensemble des réservation sous format Caldav ou d'exporter le tout en tant que calendrier google.

Chapitre 3 : DETAIL TECHNOLOGIQUE

Au cours de mes recherches concernant les diverses technologies associées à la réalisation de mon Chatbot, je me suis retrouvé à étudier quelques concepts jusque-là nouveaux pour moi et qui ont par la suite servi à affirmer mon choix technologique pour ce projet.

3.1. NATURAL LANGUAGE PROCESSING

La première technologie et une des plus importantes dans le cadre de ce projet est celle du **Natural Language Processing** (Traitement du langage naturel) (NLP). Cette technologie est une pierre angulaire au sein du domaine du machine learning et de l'intelligence artificielle. En effet, le principal intérêt de ce dernier est de permettre de prendre quelconque texte et d'appliquer divers traitements dessus en identifiant des patrons dans les phrases et mots et qui donne en sortie une suite de données structurée dont les divers modèles seront capables de les exploiter afin qu'eux aussi appliquent des traitements variés sur ces données pour finalement obtenir une réponse. Ce qui sans le NLP serait infiniment plus ardu.

L'être humain en quelque sorte applique une multitude de techniques de NLP tous les jours pour faire sens des paroles et textes que nous lisons. De manière générale, tout traitement NLP suit globalement cette séquence :

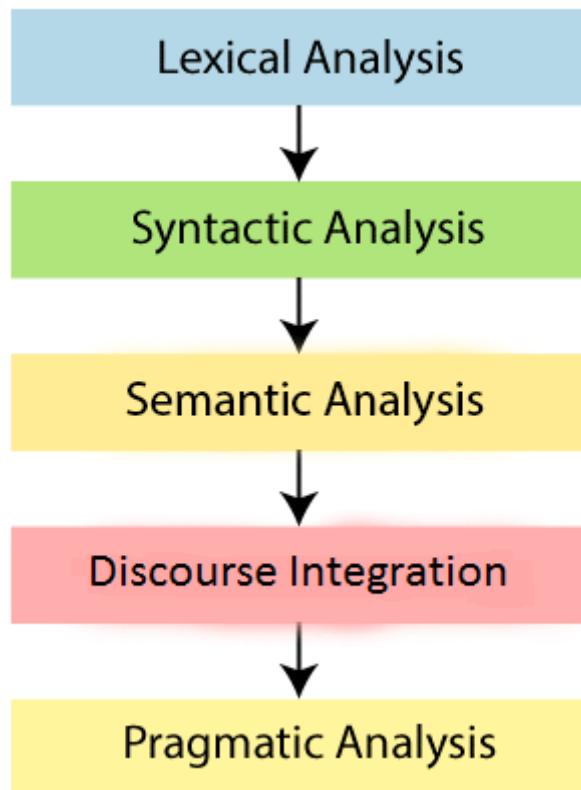


Illustration 8: Séquence de traitement NLP

Source : javapoint.com, réf. : URL02

Pour chaque étapes il existe une grande quantité de traitements divers, cependant je ne vais ici qu'en présenter une liste non exhaustive :

1. Analyse Lexicale

a. Tokenisation

La Tokenisation est une des premières techniques NLP qui va se voir être appliquée à un texte donné. Cette dernière a pour but de séparer tout ce qui s'apparente à des mots sous forme de Tokens.

Un exemple de Tokenisation serait « Je vais marcher dehors » qui deviendrait « ‘Je’, ‘vais’, ‘marcher’, ‘dehors’ ».

b. Lemmatisation et Stemming

La Lemmatisation et le Stemming sont présentés ensemble car ils ont un but très semblables étant de prendre un mot et de le réduire à une forme commune.

Pour se faire, la Lemmatisation va faire usage d'un dictionnaire de mots comprenant des groupes de mots similaires et étant tous lié par une même racine commune.

Par exemple : « Jouer, Joueur, Jeu, joué, ... » vont avoir pour mot racine « Jeu »

Ensuite, le Stemming lui, de par la signification du mot en anglais Stem voulant dire tige d'une plante et étant la base de cette dernière, consiste à prendre chaque mot et de les séparer en deux parties. Comme des mots de la même famille possèdent une base commune, retirer l'excès de lettres faisant permets d'en dégager une racine commune.

Par exemple : « Concourût, Concourir, Concourant, Concours » vont se voir réduits à « Concour ».

2. Analyse Syntaxique

a. Part-of-speech Tagging (POS)

Le POS Tagging permet de donner un tag/label à chaque mots dans un texte afin de lui assigner sa classe correspondante grammaticalement parlant. L'intérêt principal de ce traitement est de pouvoir premièrement par la suite interpréter la structure de la phrase d'une manière grammaticale. Ensuite, donner un sens clair à certains mots qui parfois selon la structure grammaticale de la phrase où ils se trouvent peuvent avoir un tout autre sens.

Par exemple : « Je me rends au pied de la montagne », les labels de POS tagging correspondants seraient « (Je, Pronom), (me, Pronom), (rend, Verbe), (au, Préposition), (pied, Nom), (de, Préposition), (la, Déterminant), (Montagne, Nom) ».

3. Analyse Sémantique

a. *Analyse de sentiments*

L’analyse de sentiments est une sorte de boite à outils dans laquelle il existe une multitude d’outils pour y parvenir mais que je ne vais pas élaborer sur. Le principe général de l’analyse de sentiments repose sur des dictionnaires de mots étant associés à des sentiments et ensuite passés au moteur NLP et si leur présence est détectée, selon la quantité présente de ces derniers ou même le manque de mots évoquant un sentiment, un ou plusieurs sentiments seront assignés au texte analysé. Il est possible dans certains cas d’associer des sentiments à des principes ou objets afin d’affiner l’analyse.

b. *Named Entity Recognition*

Le NER ou Reconnaissance d’entités nommées est un des nombreux outils NLP permettant de donner davantage de sens aux tokens présents. Selon un dictionnaire associant des mots à une catégorie prédéfinie ou même une association de classes grammaticale commune avec une catégorie d’entité, l’ensemble de tokens sélectionnés se verra attribuer un type d’entité ou dans le cas contraire, aucun.

Par exemple : « La semaine dernière, le cours de la bourse chez Twitter a chuté de 4.8% ». Les entités ressorties de cette phrase sont : « (Semaine dernière, Date), (Twitter, Organisation), (4.8%, Pourcent) »

4. Intégration de données

L'intégration des données est le simple fait de prendre les textes précédemment analysés tant dans la même session ou dans un jeu de donnée existant et de comparer les diverses analyses précédemment effectuées à celles réalisées sur le texte courant et selon le niveau de ressemblance en dégager un sens/contexte qui sera donné à ce dernier.

Un exemple serait que si l'on analyse une multitude de phrases ayant mentionné initialement un personnage homme et un personnage femme dans un livre et que par la suite le nom du personnage homme ne soit plus mentionné mais qu'à la place le pronom « il » est utilisé, par analyse des phrases précédentes il sera défini que les « il » font référence au personnage masculin.

5. Analyse Pragmatique

L'analyse Pragmatique n'indique pas spécialement une technique en particulier mais consiste davantage en la combinaison des techniques précédemment utilisées afin de déterminer le contexte global du texte. L'ensemble des tokens et labels ressortis précédemment vont, une fois comparés entre eux à l'aide de dictionnaires et autre jeu de données permettant d'aider à la définition du lien entre eux, de donner un contexte et sens au texte courant.

Le NLP en tant que tel sert principalement à traiter du texte de manière générale pour être utilisé par la suite par d'autre algorithmes. Le NLP peut être vu comme la catégorie principale en matière de traitement de texte, cependant il existe deux sous catégories au NLP étant le Natural Language Understanding ou Compréhension de langage Naturel (NLU) et le Natural Language Generation ou Génération de langage Naturel (NLG).

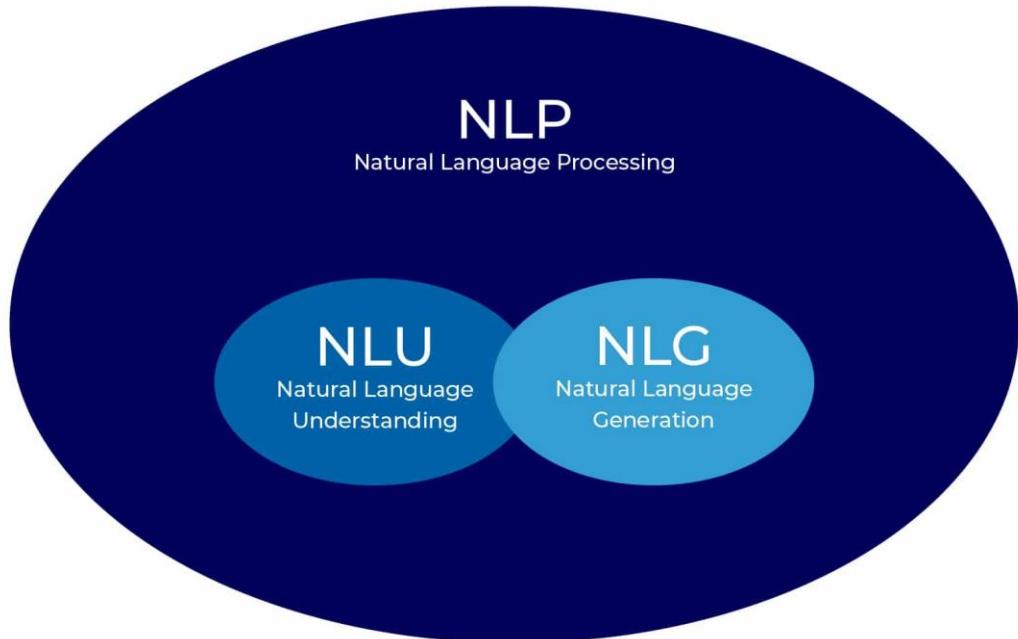


Illustration 9: Représentation de la relation NLP, NLU et NLG

Source : Datasolut, réf. : URL03

a) **NATURAL LANGUAGE UNDERSTANDING**

Selon Datasolut, “Le domaine du NLU s’occupe de la compréhension du langage naturel. Diverses méthodes de compréhension de texte sont utilisées à cet effet. Spécifiquement, la grammaire et le contexte de paires de mots ou mots uniques sont analysés afin d’en ressortir la signification de ces derniers et de la phrase. En outre, la sémantique, syntaxe, intention et émotion dégagée par un texte sont examinés.”¹

¹ WUTTKE, LAURENZ, 2023. NLP vs. NLU vs. NLG: UNTERSCHIEDE, FUNKTIONEN UND BEISPIELE. DATASOLUT GMBH [EN LIGNE]. 24 MAI 2023. DISPONIBLE À L’ADRESSE : [HTTPS://DATASOLUT.COM/NATURAL-LANGUAGE-PROCESSING-VS-NLU-VS-NLG-UNTERSCHIEDE-FUNKTIONEN-UND-BEISPIELE/](https://datasolut.com/natural-language-processing-vs-nlu-vs-nlg-unterschiede-funktionen-und-beispiele/) [CONSULTÉ LE 1 MARS 2024].

Deux concepts principaux pour parvenir à la compréhension de textes par NLU sont :

1. Détection de l'intention : Le processus d'identification des sentiments d'un utilisateur et de déterminer son objectif. Un des buts premiers du NLU étant l'établissement du sens du texte donné.
2. Détection d'entités : Un processus plus spécifique qui va chercher à identifier des entités spécifiques et d'extraire les informations les plus importantes de ces entités. Il existe deux types d'entités étant les entités nommées et numériques.

Les entités nommées sont groupées en catégories telles que des noms d'individus, de business et de lieux.

Les entités numériques sont reconnues en tant que quantités, dates, monnaies et pourcentages.²

Le NLU est une technologie prévalente dans le développement de Chatbots et autres assistants intelligents ayant pour but de converser avec un utilisateur et celle-ci leur permet de comprendre ce que ce dernier leur répond.

b) NATURAL LANGUAGE GENERATION

Le NLG est la deuxième sous-catégorie de NLP ici servant à générer du texte en langage naturel. Cette catégorie de NLP est une partie vitale de nombre d'I.A. conversationnelles car il ne suffit pas simplement de comprendre ce que l'utilisateur leur dit mais d'aussi pouvoir ressortir une réponse cohérente et intelligible par un humain. Pour se faire, on emploie de la génération de langage naturel.

² WHAT IS NATURAL LANGUAGE UNDERSTANDING (NLU)? | DEFINITION FROM TECHTARGET, ENTERPRISE AI [EN LIGNE]. DISPONIBLE À L'ADRESSE : [HTTPS://WWW.TECHTARGET.COM/SEARCHENTERPRISEAI/DEFINITION/NATURAL-LANGUAGE-UNDERSTANDING-NLU](https://www.techtarget.com/searchenterpriseai/definition/natural-language-understanding-nlu) [CONSULTÉ LE 1 MARS 2024].

Le fonctionnement du NLG varie selon son application, cependant un modèle général est détaillé par R. Dale et E. Reiter³ dans une de leurs publications comme suit :

1. Détermination de Contenu et plannification du discours : La détermination de contenu est une des premières étapes de NLG qui va se charger de trouver quelles informations vont être communiquée sous la forme d'un texte à l'utilisateur. Afin d'y parvenir, un filtre sera appliqué sur l'ensemble des données afin de ne récupérer que ce qui semblerait pertinent et de résumer chaque données afin de ne garder que l'essentiel.

La plannification du discours, lui, est l'action de réordonnancement des informations précédemment récupérées dans un ordre spécifique afin de ne pas exprimer ces dernières de manière disparate.

2. Aggrégation de phrases : Cette étape a pour but de prendre plusieurs messages et de les regrouper en une seule phrase. Par exemple si de multiples messages mentionnent des informations concernant un cours comme suit :

- “Le cours de Français est donné par Mr. Bournon”
- “Le cours de Français aura lieu en salle A201”
- “Le cours de Français commence à 15h”

Sera probablement aggrégué de la manière suivante : “Le cours de Français qui aura lieu en salle A201, donné par Mr. Bournon, commence à 15h”.

Il est important de mentionner qu'il existe divers techniques d'aggregation et que la phrase présentée ci-dessus n'est qu'un exemple parmi tant d'autres.

³ REITER, EHUD ET DALE, ROBERT, 1997. BUILDING APPLIED NATURAL LANGUAGE GENERATION SYSTEMS. NATURAL LANGUAGE ENGINEERING. VOL. 3, NO 1, PP. 57-87.
DOI 10.1017/S1351324997001502.

3. Lexicalisation : Cette étape du processus de NLG n'est pas tout le temps obligatoire mais elle sert à remplacer des données présentes sous forme de concepts ou entités et de les transformer en un ou plusieurs mots qui seront à la place affichés.

Un exemple de lexicalisation est : Dans un cadre d'une réservation de tables dans un restaurant ("RESERVATION" -> "Une réservation", "MARINIÈRE" "à la Marinière", "4 TABLES" -> "de quatre tables"), on obtiendrait donc après lexicalisation : "Une réservation de quatre tables à la Marinière"

4. Génération d'expressions référentielles : Cette étape va se concentrer sur la désambiguation des divers entités trouvées dans les informations à disposition en les remplaçants par des noms, pronoms, descriptions détaillées et autre type d'expressions référentielles.

Par exemple, si dans les informations que l'on a se trouve des références à Microsoft, il est possible qu'après un traitement par Génération d'expressions référentielles qu'initiallement Microsoft soit mentionné comme tel dans une phrase initiallement puis que par la suite ce dernier soit mentionné sous la forme de "Ce dernier", "La compagnie ayant crée le système d'exploitation Windows", "Ce géant de la Silicon Valley" et bien d'autres encore.

5. Réalisation : Cette dernière étape se charge surtout d'appliquer des correctifs orthographiques et de forme aux mots déjà présents dans la phrase mais ne se charge plus de la syntaxe, seulement de l'orthographe, conjugaison, accords et tout ce qui ne s'apparente pas à de la syntaxe.

Ce qui est intéressant à dénoter si l'on se réfère à nouveaux à *l'illustration 9*, le fait que la bulle NLU se retrouve partiellement couverte par le NLG n'est pas une coïncidence car comme on le comprends avec le modèle précédemment établi, le NLU est une partie intégrante du processus de NLG. Afin de pouvoir générer quelque phrase qui puisse, il

faut tout d'abord comprendre les données à la disposition du modèle afin de les assembler en une phrase éligible et faisant sens.

La technologie du NLP ainsi que ses deux sous-catégories enfants NLU et NLG sont des éléments fondateurs des modèles d'I.A. conversationnelles comme ChatGPT qui pour pouvoir comprendre les entrées des usagers et leur renvoyer une réponse utilise ces deux principes couramment.

3.2. TAPAS

La seconde technologie ici présentée est TAPAS. TAPAS ou Parseur de Tables est un modèle d'I.A. à faible supervision pré-entraîné développé par des ingénieurs de chez Google en 2020 et reposant sur le modèle existant BERT pour sa base et détaillé dans un article de recherche publié par l'ACL (Association for Computational Linguistics)⁴.

Avant sa conception, le parsage de textes sous forme de langage naturel se faisait principalement à l'aide de moules logiques décrivant en simples termes la forme que prends l'information que l'on cherche sémantiquement parlant. Ce qui demandait une quantité considérable de pré-calculs en amont du parsage du texte.

C'est donc pourquoi une approche populaire au parsage sémantique est l'usage d'une supervision faible, ce qui dans le contexte du machine learning est le fait qu'au lieu de chercher de manière exacte le label d'une donnée, les labels utilisés lors de l'entraînement du modèle seront volontairement incomplets, ayant du bruit sous forme de lettres en trop par exemple et peu précis comparé au label cible. Dans notre cas, cela voudrait dire que lors d'une requête de parsage de table de donnée, au lieu de chercher le label qui correspond exactement à celui fourni par l'utilisateur dans sa requête, on va aller chercher

⁴ HERZIG, JONATHAN ET AL., 2020. TAPAS: WEAKLY SUPERVISED TABLE PARSING VIA PRE-TRAINING. IN : PROCEEDINGS OF THE 58TH ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, PP. 4320-4333. 2020. DOI 10.18653/v1/2020_ACL-MAIN.398. ARXIV:2004.02349 [CS]

les données correspondant à un label s'approchant le plus de ce dernier sans pour autant chercher une exactitude forte. Or, cette approche reste tout de même couteuse car nécessitant la génération de moules logiques même pour ces labels.

L'approche que prends TAPAS est par conséquent de ne pas avoir à générer ces moules logiques pour trouver les bons labels mais d'effectuer une sélection de cellules ainsi qu'optionnellement d'appliquer divers opérateurs à cette dernière. Comme TAPAS est basé sur BERT, un framework NLP, il est donc possible d'effectuer de simples requêtes sous forme de langage naturel et qu'ensuite TAPAS comprenne cette requête et retourne l'ensemble de données souhaité similairement à une requête SQL mais sans avoir à rentrer une requête en langage SQL.

Afin de mieux comprendre le fonctionnement de TAPAS, il faut se pencher sur sa fondation étant BERT et pour se faire, penchons nous sur ce qu'est un Transformer, un des composants principaux de BERT.

a) TRANSFORMERS

Un transformer est “un réseau neuronal qui apprends le contexte et compréhension au travers d’analyse séquentielle de données”⁵. Afin d’y parvenir, un suite de N encodeurs vont analyser de manière parallèle des séquences de tokens et y appliquer le mécanisme d’attention.

Le mécanisme d’attention propre aux transformers est le simple fait de donner un poids à chaque paire de tokens et ce poids est affecté par à quel point un token est lié à un autre. De ce fait, après avoir passé notre séquence de tokens dans les divers couches d’encodeurs, le résultat de ces encodeurs seront passés au décodeur qui lui va permettre

⁵ THE ULTIMATE GUIDE TO TRANSFORMER DEEP LEARNING, [EN LIGNE]. DISPONIBLE À L’ADRESSE : [HTTPS://WWW.TURING.COM/KB/BRIEF-INTRODUCTION-TO-TRANSFORMERS-AND-THEIR-POWER](https://www.turing.com/kb/brief-introduction-to-transformers-and-their-power) [CONSULTÉ LE 2 MARS 2024].

de générer la séquence résultat selon le contexte dérivé de la valeur d'attention propre à chaque tokens.

Par exemple, si l'on a une phrase étant “As-tu bien dormis ?”, grâce à un set d'entraînement qui défini le lien entre les mots présents dans cette phrase et d'autres mots commun par une valeur d'attention élevée et selon un contexte donné comme ici étant une question, une sortie possible serait “Oui, j'ai bien dormis”.

En d'autres mots, les transformers sont fondamentaux à la compréhension de textes et prédition de phrases et suites logiques possibles à un texte donné pour le modèle BERT grâce à ce mécanisme d'attention.

b) BERT

BERT ou Bidirectional Encoder Representations from Transformers (Représentation d'encodeurs bidirectionnels de Transformers) a été crée en Octobre 2018 par Google. L'utilité principale de BERT est la compréhension de langage naturel à l'aide de transformers et étant utilisé dans des produits très mondialement connus comme pour ne citer que lui, le moteur de recherche Google :

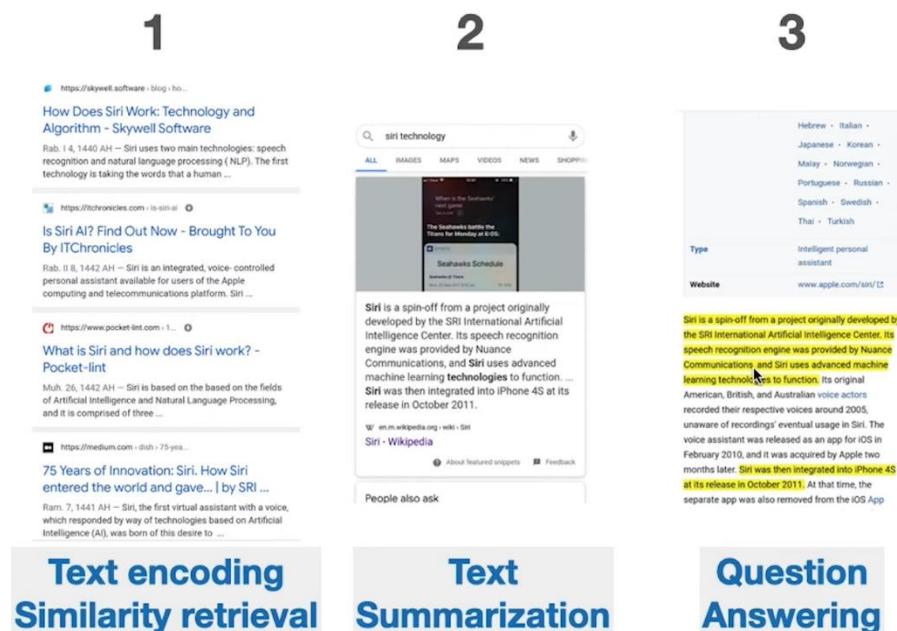


Illustration 10: Cas exemples du moteur google utilisant BERT

Source : Language processing with BERT by Jay Alammar, réf. :URL04

1. Récupération de similarité d'encodage de texte : La requête de l'utilisateur va être traitée par des techniques NLP et ressortie sous forme de tags. Ce qui va permettre à BERT de trouver des pages web qui possèdent des caractéristiques similaires à celles de la requête et lui retourner les pages les plus pertinentes.
2. Résumé de texte : Lors d'une recherche sur un sujet comme par exemple "Siri technology", l'aide de diverses techniques NLP présentes dans le modèle BERT vont permettre de générer un résumé cohérent et concis des informations que l'on cherche à savoir grâce à la compréhension de contexte et sens des mots dont BERT de part l'usage de transformers est particulièrement bon à faire.
3. Question-réponse : Un forme plus simple est en utilisant la requête utilisateur qui une fois passée dans le modèle BERT, ce dernier va chercher quel passage du texte wikipedia correspond le plus à une réponse valable à la requête demandée et dans le cas présent le texte sera surligné sur ledit site.

Un exemple plus concret de BERT en application est donné par Techtarget⁶ :

La phrase que BERT va recevoir ici est « The animal didn't cross the street because it was too wide ». À un moment donné le mot "it" sera sélectionné lors du passage du texte dans les couches de l'encodeur. Il sera ensuite mis face à tous les autres mots présents dans la phrase et selon l'algorithme NLP mis en place dans le modèle BERT courant, il sera déterminé que IT est un pronom qui désigne une entité. De ce fait, BERT va tenter

⁶ WHAT IS BERT (LANGUAGE MODEL) AND HOW DOES IT WORK?, ENTERPRISE AI [EN LIGNE]. DISPONIBLE A L'ADRESSE :

[HTTPS://WWW.TECHTARGET.COM/SEARCHENTERPRISEAI/DEFINITION/BERT-LANGUAGE-MODEL](https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model) [CONSULTE LE 4 JANVIER 2024].

d'identifier à quelle entité « IT » pourrait faire référence à et les trois mots possibles ici sont

- « The »
- « animal »
- « street »

Après usage du mécanisme d'attention qui va calculer dynamiquement la connexion entre chaque paires de mots dans la phrase et leur attribuer un poids, le mot ayant la plus forte connexion ici est « Street » et ayant un poids plus important est donc déterminé comme le mot auquel « IT » faisait référence à, donnant ainsi davantage de contexte.

Grâce à la lecture bidirectionnelle utilisée par BERT, le modèle est donc capable de donner un sens à des mots, phrases ou textes entiers selon la façon dont les mots sont employés et de par quelle manière chaque phrases sont formées. Sens qui est donné au travers de tags qui permettent à BERT d'être utilisé dans le cadre de :

- Question-Réponse à l'aide de divers textes donnés au modèle
- Capacité à résumer efficacement des textes
- Prédiction de phrases
- Génération de réponses lors d'une conversation, similaire à ChatGPT
- Analyse approfondie d'un texte afin de mieux cerner l'intention de ce dernier

L'entrée du modèle BERT est une suite de vecteurs nommés tokens et qui dans le cas de l'illustration qui suit représentent une phrase.

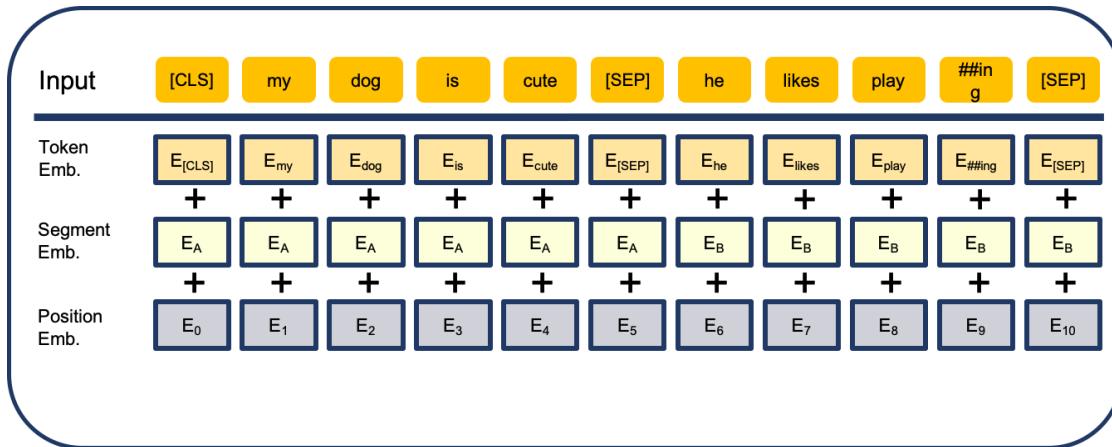


Illustration 11: Représentation de l'entrée du modèle BERT

Source : Humboldt Universität, réf. : URL05

Dans un input, le premier token est toujours un token de Classification **[CLS]**, suivi ensuite ici de la première moitié du texte, d'un séparateur **[SEP]** pour marquer la fin de la première partie et le début de la deuxième et marqué pour finir d'un nouveau séparateur et ainsi de suite. Cette suite de tokens est le Token Embedding.

Ensuite une autre couche est ajoutée étant le Segment Embedding. Ce dernier consiste à définir quels tokens appartiennent, dans le cas présent, à la première ou deuxième moitié du texte.

Pour finir, une dernière couche est ajoutée étant le Position Embedding qui permet de donner un index à chaque tokens.

c) TAPAS : FONCTIONNEMENT

En contraste avec l'entrée du modèle BERT, l'entrée pour le modèle TAPAS se présente sous la forme suivante :

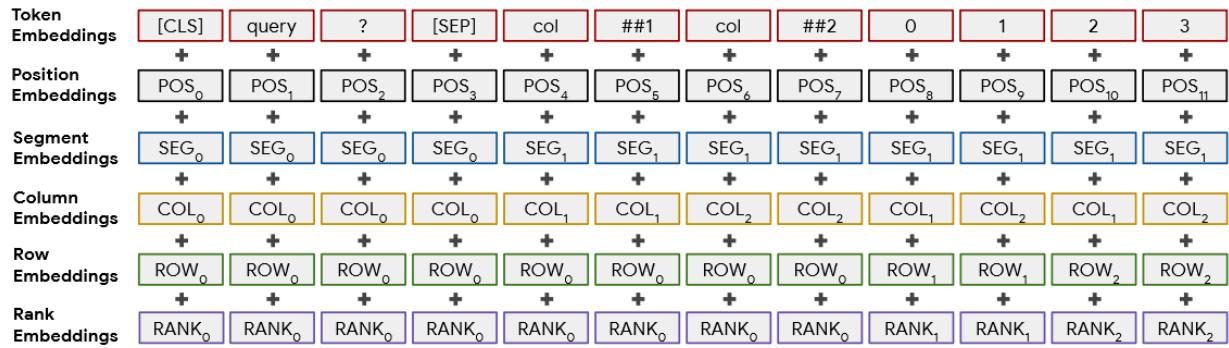


Illustration 12: Entrée du modèle TAPAS

Source : TAPAS: Weakly Supervised Table Parsing via Pre-training, réf. : URL06

Comparativement à l'entrée BERT, celle de TAPAS inclu avant la première séparation la requête de l'utilisateur et après le [SEP] le tableau à analyser mis à plat. Le “col” et “##1” ici représentent le nom de la première colonne et celui qui suit celui de la 2ème colonne. Il peut y en avoir autant qu'il y a de colonnes dans le tableau. Fait suite à cela les données qui sont assemblées de manière séquentielle de sorte à traverser le tableau de gauche à droite selon le nombre de colonnes et de haut en bas.

Si l'on devait créer le tableau que l'on a dans l'exemple il ressemblerait à cela :

Col1	Col2
0	1
2	3

Tableau 1: Représentation du tableau fourni en entrée de TAPAS

Source : Rodrigues dos Santos Fabio

La couche Position Embeddings est fonctionnellement identique à celle de BERT, servant à donner des indexées à chaque tokens.

La couche Segment Embeddings permet de définir si un token appartient à la requête ou au tableau.

La couche Column et Row Embeddings ont pour but de spécifier dans l'entrée même si une donnée provient d'une colonne et ligne donnée afin de ne pas perdre la structure initiale du tableau.

La dernière couche, Rank Embeddings, permet de donner un ordre aux divers tokens présents. Un type d'ordre peut simplement être un ordre numérique croissant comme ici : Si les tokens présents sont “3”, “4”, “2” et “7”, le rank correspondant à chaque tokens serait de l'ordre de “2”, “3”, “1” et “4”.

La procédure qui va survenir après avoir fourni à l'entrée du modèle la requête et le tableau duquel l'on souhaite extraire des données peut être représentée de la forme suivante :

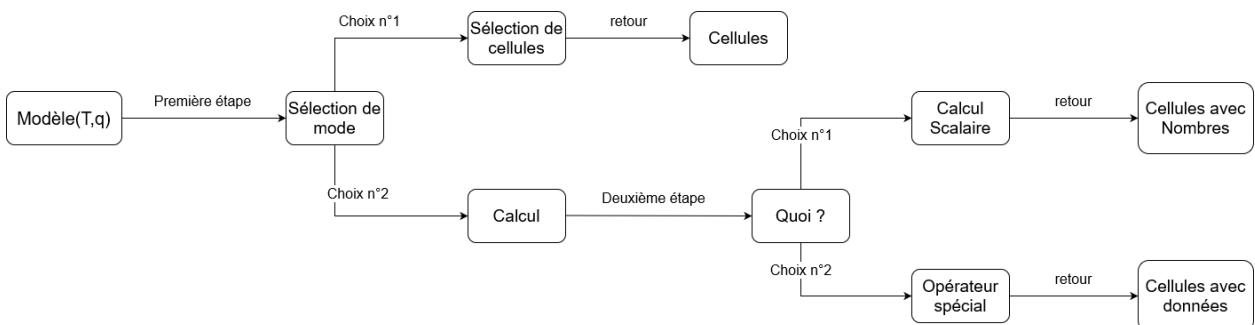


Illustration 13: Schéma de fonctionnement du modèle TAPAS

Source : Rodrigues dos Santos Fabio

La procédure définie ci-dessus démontre les divers choix que va effectuer le modèle afin de déterminer la bonne suite d'opérations à appliquer pour ressortir un résultat.

Premièrement, l'utilisateur fourni une requête et un tableau au modèle. Ensuite vient la première étape étant la sélection de mode. Les modes disponibles sont :

- Sélection de cellules
- Calcul

En premier lieu, si le mode choisi est une sélection de cellules, le résultat en sortie ne sera qu'une simple récupération de la donnée se trouvant dans la/les cellule/s permettant de répondre à la requête en entrée.

Sinon, le modèle n'aura d'autres choix que de choisir le mode calcul qui lui va faire basculer le modèle dans une deuxième phase qui a pour but de choisir quelles cellules doivent être sélectionnées pour ensuite appliquer un opérateur parmi les 4 suivants :

- Rien, on renvoie simplement les cellules sélectionnées indiquant un cas de simple sélection de cellules.
- Compter, on compte le nombre de cellules sélectionnées. C'est une réponse Ambigüe.
- Somme, on calcule la somme de toutes les cellules sélectionnées. C'est une réponse scalaire.
- Moyenne, on calcule la moyenne de toutes les cellules sélectionnées. C'est une réponse scalaire.

Afin de choisir les bonnes cases et opérations, un softmax (Chaque choix possible va se voir recevoir un pourcentage basé sur la cohérence avec la requête) est appliqué sur chaque cases et ensuite de même pour les opérateurs afin de trouver ce qui répond au mieux à la requête initiale.

En l'état, les opérateurs précédemment énoncés sont les seuls opérateurs disponibles présentement avec le modèle mais il est possible, selon le papier de recherche des ingénieurs de Google d'en ajouter davantage soi-même.

Le fonctionnement actuel des opérations effectuées par le modèle en son état actuel dans le projet de base de Google⁷ fait qu'un tuple contenant le type d'opération et les cellules

⁷ GOOGLE-RESEARCH/TAPAS [LOGICIEL] [EN LIGNE]. 29 FEVRIER 2024. GOOGLE RESEARCH. [CONSULTE LE 3 MARS 2024]. DISPONIBLE A L'ADRESSE : [HTTPS://GITHUB.COM/GOOGLE-RESEARCH/TAPAS](https://github.com/google-research/tapas) [CONSULTE LE 3 MARS 2024].

sélectionnées sont retournée, demandant donc un post-traitement des données afin d'appliquer l'opérateur choisi par le modèle.

La manière dont est entraîné par conséquent le modèle est que chaque opérateur, mode et sélection de cellules vont posséder, de manière inhérente à tout réseau neuronal, des poids probabiliste et leur valeur correspondante et en faisant la fonction de coût de l'erreur carrée basée sur le résultat correct et le modèle va effectuer une propagation arrière et ainsi de suite jusqu'à arriver de manière régulière à tomber sur les bons paramètres.

La seule particularité est quand l'on peut avoir une solution ambiguë qui nécessite un traitement particulier car le résultat ne se trouve pas explicitement dans le tableau comme dans le cas d'un comptage de cellules. Pour se faire, un softmax sur les deux possibilités étant une sélection de cellules ou une agrégation sera appliquée pour déterminer la solution optimale.

Une dernière fonctionnalité intéressante de TAPAS est que l'on peut faire usage d'un système de checkpoints ou points de sauvegarde afin de pouvoir poser des questions subséquentes. Or, dans le cadre de ce projet, ce n'est pas nécessairement pertinent.

Chapitre 4 : ÉTALAGE DES TECHNOLOGIES EXISTANTES

Après avoir énoncé et approfondi quelques technologies, je m’apprête à présent à présenter quelques implémentations de ces technologies qui pourraient servir à la réalisation du projet final.

Afin de pouvoir correctement assimiler le fonctionnement des bots qui vont suivre, il est important d’expliciter le fonctionnement de Chatbots par NLU.

4.1. CHATBOTS PAR NLU

À noter que les terminologies explicitées ci-dessous sont des termes trouvés principalement dans des Chatbots créés avec Rasa Open Source.

a) LES INTENTS

Le manière dont fonctionne des Chatbots par NLU commence premièrement par la détection d'**intents**.

Un intent, soit une intention en Français, est généralement un type d'action prédéfini dans le modèle d'I.A. utilisé par le Chatbot et servant à démarquer toutes les possibles actions que pourrait entreprendre l'utilisateur.

Un exemple d'intent peut être « **DETAILS_RESERVATION** » et « **RESTAURANT_RESERVATION** ». Dans un premier lieu, l'utilisateur va démarrer la conversation avec le Chatbot et à un certain point il pourrait dire quelque chose de la sorte : « J'aimerais réserver une table chez <Nom restaurant> ». L'intent qui sera dégagé ici sera très probablement **RESTAURANT_RESERVATION** car dans ce dernier on pourrait définir que le fait de trouver les mots suivants par l'usage de techniques de NLP : « <nom_restaurant> » et « réserver » consisterait à indiquer l'intention de l'utilisateur à réserver une table dans un restaurant. Ensuite il est possible de faire en sorte que le bot

demande à l'utilisateur des précisions sur sa réservation et si l'utilisateur fourni les paramètres de la réservation attendus par le bot, l'intention **DETAILS_RESERVATION** sera détectée et les informations fournies par l'utilisateur traitées en conséquence.

b) LES ENTITÉS

Un élément important composant bien souvent des intents sont les Entités ou Entities. Les entités peuvent être définies de multiples manières dans le contexte de développement de Chatbot par NLU mais elles sont généralement représentées par une liste de mots étant associés à une entité en particulier.

Par exemple, une entité définie comme "légumes" pourrait être trouvée lors de l'analyse de textes si les mots "Patate", "Carotte", "Choux blancs", "Poivron" et bien d'autres, soient détectés.

c) LES STORIES

Afin de faire sens d'une multitude d'intents, l'usage de stories est primordiale pour parvenir à mettre en place un flux logique de conversation et principalement pour l'entraînement du Chatbot qui sans stories, ne aurait de la peine à associer correctement les divers intents détectés à une action donnée par exemple.

Par exemple, si l'on a deux stories définies comme:

- Story 1
 - o Intent: intent_1
 - o Intent: intent_2
 - o Action: bonjour

Et

- Story 2

- o Intent: intent_1
- o Action: bienvenue

Et qu'ensuite l'utilisateur effectue une requête au Chatbot et que cette dernière après inspection contient l'intent_1 uniquement, la story 2 sera celle sélectionnée, sinon s'il y a aussi l'intent_2, la story 1 sera celle sélectionnée en fin de compte.

d) LES ACTIONS

Un élément présent dans l'explication des stories est la présence d'actions. Une action comme son nom l'indique est une suite d'instructions qui sera, dans le cas actuel, appelée lorsque la Story courante lui fait appel. Son fonctionnement est techniquement identique à celui d'une fonction en programmation mais incorporée dans le fonctionnement de Chatbots par NLU.

e) LES RÈGLES

Les règles sont des suites d'actions qui seront exécutées impérativement selon l'intent ou condition spécifiée au préalable. Les règles sont similaires fonctionnellement parlant aux stories mais avec la différence qu'elles sont activées même si l'on est déjà à l'intérieur d'une story.

Par exemple, si une règle est définie quand **intent_1** est détecté et est sensé afficher un message, si l'on se trouve dans quelconque story et qu'**intent_1** est trouvé, le message sera immédiatement envoyé en plus des autres possibles messages du Chatbot.

f) LES SLOTS

Lorsque la conversation avec l'utilisateur progresse, il est possible que des informations fournies par l'utilisateur se doivent d'être stockées afin d'influencer le reste

de la conversation. C'est dans ce cas présent que les slots entrent en jeu. Les slots sont une façon pour le Chatbot d'enregistrer en mémoire le temps d'une conversation avec un utilisateur donné le contenu d'entités ou de tout autre type d'information.

g) LES LOOKUP TABLES

Il existe une multitude cas où nous souhaiterions détecter une entité mais qu'il existe une variété de mots correspondant à la même entité. C'est pourquoi on fait usage d'une lookup table. Les lookup tables fonctionnement d'une manière similaire à une expression régulière dans le sens où si un mot se trouve dans une liste de mots associés à une entité, l'entité en question sera définie comme détectée dans le traitement du message utilisateur courant.

Par exemple, si l'on possède une entité meuble et que l'on ne souhaite pas à avoir à définir une entité pour chaque meubles, on peut alors définir une lookup table contenant "Table", "Chaise", "Armoire", "Canapé" et bien d'autres. Si un seul de ces mots figure dans le message de l'utilisateur, alors l'entité meuble sera détectée.

h) LES FORMS

Les forms ou formulaires sont des moules pouvant être utilisé dans de multiples scénarios où l'on veut s'assurer de récupérer un certain nombre de slots. Dans quel cas on défini un formulaire et chaque slots associés et une fois cela fait, on peut faire usage de ce formulaire.

Par exemple, si l'on possède un formulaire identité avec comme slots requis "nom" et "prénom", on peut paramétrier le Chatbot de sorte à ce qu'il persiste à demander dans la même boucle de conversation tant qu'il n'a pas reçu le **nom** et **prénom** de l'utilisateur.

4.2. RASA

Une des premières est dans le cadre de la réalisation d'un Chatbot. Rasa est un framework de création d'assistants conversationnels intelligents à l'aide de diverses techniques de machine learning créée en 2016.

a) RASA X

Avant toute chose, je trouve pertinent de mentionner qu'il existe un moyen d'entrainer, paramétrier et tester des chatbots créés avec une interface graphique mise à disposition par Rasa étant Rasa X.

Un bref aperçu nous est montré grâce à un tutoriel disponible sur la section blog de Rasa⁸:

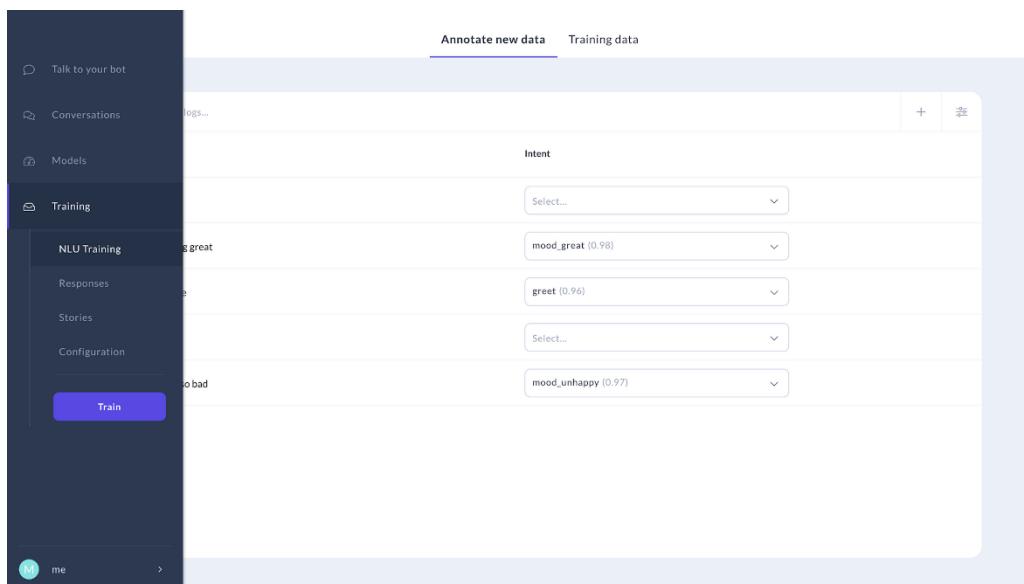


Illustration 14: Interface contenant le menu de Rasa X

Source : Rasa.com, réf. : URL07

Lorsque l'on démarre Rasa X, le menu que l'on voit ci-dessus présente les divers pages que l'on peut accéder afin de :

⁸ GETTING STARTED WITH RASA X AS AN EXISTING RASA USER | RASA BLOG, RASA [EN LIGNE]. DISPONIBLE À L'ADRESSE : HTTPS://RASA.COM/BLOG/RASA-X-GETTING-STARTED-AS-A-CURRENT-RASA-USER/ [CONSULTÉ LE 3 MARS 2024].

- Gérer les modèles existant
- Entrainer un modèle
- Modifier les stories
- Modifier la configuration de la pipeline d'un modèle
- Visualiser l'historique des conversations
- Ouvrir une fenêtre de chat

Si l'on ouvre Rasa X dans un répertoire contenant un projet existant et possédant déjà des modèles pré-entraînés, ces derniers ainsi que leurs configurations apparaîtront dans les diverses pages mentionnées ci-dessus.

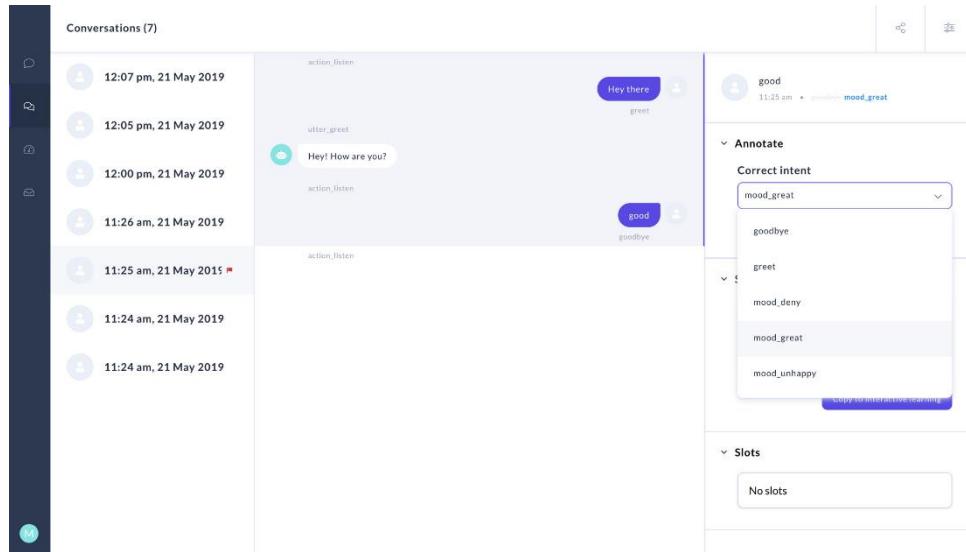


Illustration 16: Exemple de conversation parmi l'historique des conversations

Source : Rasa.com, réf. : URL08

Ci-dessus se trouve un exemple de conversation entre l'utilisateur et le chatbot et on peut apercevoir qu'il est possible de corriger presque en temps réel les prédictions faites par pas le bot si l'intent trouvé est erroné et d'autres paramètres tels que les slots.

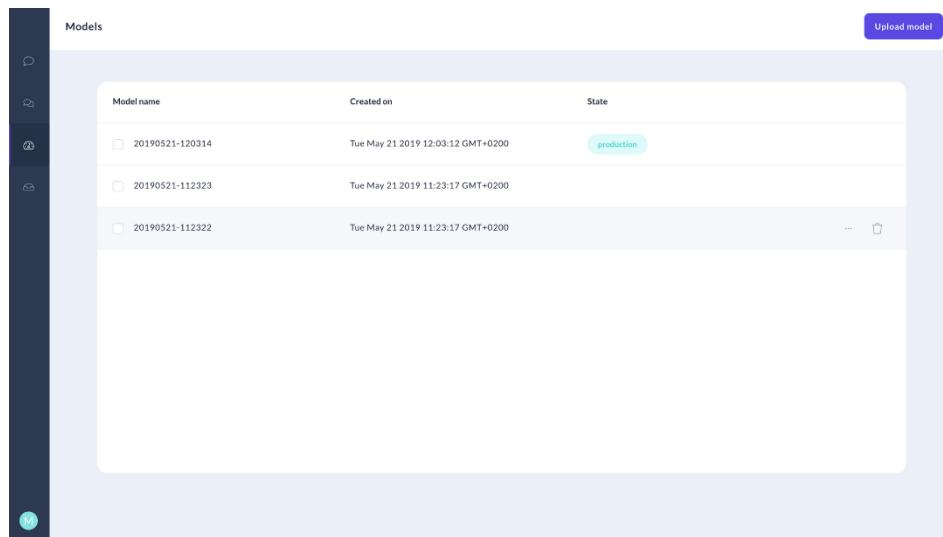


Illustration 15: Page de gestion des modèles avec Rasa X

Source : Rasa.com, réf. : URL09

Cette dernière image démontre la facilité de gestion des modèles existant, d'en supprimer ou ajouter de nouveaux au besoin et ce en quelques clics.

b) RASA OPEN SOURCE

Rasa Open Source est la solution Open Source de Rasa mettant à disposition tout un SDK disponible uniquement en python. Bien qu'il ne soit que disponible en Python il est toutefois possible d'utiliser quelconque autre language pour traiter les données reçues du Chatbot Rasa étant donné qu'il fonctionne grâce à un serveur REST avec des points de sortie pouvant être appelé au besoin.

Rasa Open Source mets à disposition un wiki très détaillé quand à l'usage du SDK. Le fonctionnement de Rasa Open Source peut être décomposé en trois parties principales.

La première, le Rasa NLU, correspond au moteur NLP défini plus tôt dans la définition d'un Chatbot. Ce dernier va se charger de détecter les divers entités et intents et de les extraire.

Le deuxième, le Rasa Core, correspond au moteur de réponses défini plus tôt dans la définition d'un Chatbot. Le Core va avoir pour rôle de gérer tout l'aspect conversationnel en intégrant les entités et intents extraits par le Rasa NLU et de les intégrer au système de dialogues définis par les stories et règles configurées au préalable.

La troisième, L'agent Rasa, permet de fournir l'interface logicielle qui permet d'intégrer avec les deux composants précédents. L'agent permet de démarrer un entraînement de modèle, gérer la réception de messages, le chargement de modèles de dialogue, la réception d'actions et la gestion de canaux externes afin de permettre au Chatbot Rasa de communiquer directement sur des applications de messageries connues comme Facebook Messenger, Slack, Mattermost et même de mettre en place une intégration sur son propre site personnel.

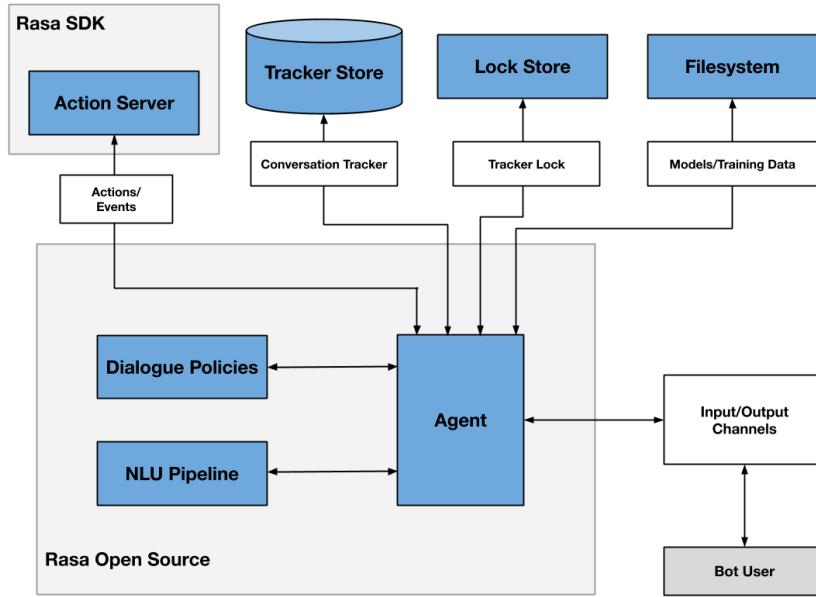


Illustration 17: Architecture Rasa

Source : Rasa.com, réf. : URL10

Rasa Core est représenté ici comme « Dialogue Policies »

Rasa NLU est représenté ici comme « NLU Pipeline »

Un quatrième composant que l'on peut faire usage est un Rasa Action Server qui lui utilise le SDK Rasa en python mis à disposition et qui permet fonctionnellement d'utiliser les mêmes actions implicites comme explicites que les diverses commandes Rasa peuvent faire ainsi que les instructions définies dans les configurations du Chatbot comme des Stories, affectations de slots, lancement d'évènement, etc. Le Rasa Action Server est utilisé dans le cas où l'on souhaite intégrer des actions personnalisées qui doivent, par exemple, interagir avec une API ou effectuer une multitude d'actions qui ne peuvent pas simplement être définies dans les fichier YAML du projet Rasa.

Un cinquième composant est Rasa Tracker Store. Un élément récurrent lors du développement avec le SDK de Rasa est le Tracker. Le Tracker est en pratique, la mémoire du Chatbot. Par défaut, le Tracker enregistre l'ensemble des conversations et évènements passés. Comme le Tracker enregistre le tout en mémoire, si le bot est redémarré, l'ensemble des données seront perdues. C'est dans ce cas qu'il est possible d'utiliser un Tracker Store

qui permet de lier le Tracker Rasa à une base de donnée pour stocker toutes ces informations.

Un dernier composant pouvant être utilisé est le Rasa NLG Server. Lors de l'entraînement du Chatbot, l'ensemble des réponses sont sauvegardées dans le modèle. Ainsi, si l'on souhaite modifier ne serait-ce qu'un caractère dans une réponse, il sera nécessaire de réentrainer l'ensemble du modèle. Afin d'éviter de perdre du temps à réentrainer le modèle constamment, on peut faire usage d'un NLG Server qui aura pour but de renseigner le Chatbot la réponse qu'il cherche. Et comme elle se trouve sur le serveur NLG, il ne suffit que de le redémarrer pour mettre à jour les réponses pouvant être retournées à l'utilisateur par le Chatbot.

Les divers composants présentés ci-dessus sont une liste non exhaustive des composants mis à disposition par l'outil Rasa et grâce à ceux-ci, on peut déjà si l'on le souhaite mettre en place un Chatbot bien assez robuste et efficace.

c) RASA PRO

Rasa Pro est un service fourni par Rasa qui contrairement à sa contrepartie Rasa Open Source, est elle Open Core. Open Core dans ce cas-ci voulant dire que toutes les fonctionnalités de Rasa Open Source sont utilisables par les usagers de Rasa Pro. Sauf que les usagers de Rasa Pro possèdent des fonctionnalités supplémentaires et propriétaires.

De ce que j'ai pu découvrir au long de mes recherches sur Rasa Pro, ce service fournit principalement un produit étant l'usage de CALM au sein de Chatbots Rasa.

CALM soit **C**onversational **A**I with **L**anguage **M**odels (I.A. conversationnelle avec des modèles de langage), permet de presque entièrement se débarasser du fonctionnement du Chatbot par NLU de Rasa et de faire usage de divers fournisseurs de LLM tels que OpenAI, Cohere, Hugging Face Hub, Llama-cpp ou Vertex AI. Il est possible de faire usage d'autres fournisseurs mais cela nécessite une configuration manuelle.

CALM permet de se débarasser du système d'entités et de stories et de faire à la place usage d'un couplage entre des Flows et un LLM au choix.

Un Flow est similaire à une story dans l'idée qu'il décrit la logique conversationnelle mais à la différence d'une story, un Flow n'a pas besoin de spécifier tous les branchements différents de conversation. À la place, il ne suffit que de définir une **description** qui sera utilisée lors de la communication du message de l'utilisateur au LLM afin de pouvoir déterminer l'action devant être entreprise ensuite. Il est aussi possible de faire usage de **Slots** et de **Conditions** pouvant être appliqués sur ces slots selon l'état de ces derniers. Similairement à des Forms il est possible de faire usage de **Collects** qui ne passera pas à la suite de la conversation tant que l'utilisateur n'a pas donné les informations souhaitées.

Il existe à actuellement des limitations aux Flows qui sont dûs à la taille des contextes mis à disposition par les divers LLMs disponibles. La quantité de Flows pouvant être gérés par Rasa Pro dépend de :

- La longueur des descriptions des Flows
- Le nombre de slots dans chaque Flows
- La longueur de la description de chaque slots
- Les types de slots et nombres de valeurs autorisées

De plus, grâce à l'usage d'un LLM, il est possible de l'intégrer au processus de NLG du Chatbot afin de pouvoir générer sur mesure et dynamiquement des réponses aux divers messages de l'utilisateur.

Une multitude d'autres fonctionnalités supplémentaires sont mis à disposition par Rasa Pro mais sont plus liés à de l'aide à la mise en place d'infrastructures autour du Bot et pas aussi conséquents que leur fonctionnalité CALM.

Il n'existe pas de prix fixe pour faire l'acquisition de Rasa Pro car le coût va varier selon le client et ce qu'il cherche à réaliser avec Rasa Pro. Pour en faire l'acquisition il faut contacter directement le département des ventes de Rasa.

d) DUCKLING

Un composant ayant été découvert lors de la réalisation du prototype énoncé plus tard est Duckling. Duckling est une librairie écrite en Haskell créée par Facebook en 2017 et servant à parser du texte en une structure de donnée.

Les divers types de données étant actuellement supportées par Duckling sont appelés **domaines** et les domaines actuellement supportés sont les suivants selon la documentation Duckling⁹:

Dimension	Exemple d'entrée	Example de sortie
Quantité d'argent	42€	{"value":42,"type":"value","unit":"EUR"}
Numéro de carte de crédit	4111-1111-1111-1111	{"value":"4111111111111111","issuer":"visa"}
Distance	6 miles	{"value":6,"type":"value","unit":"mile"}
Durée	3 mins	{"value":3,"minute":3,"unit":"minute","normalized":{"value":180,"unit":"second"}}}
Email	duckling-team@fb.com	{"value":"duckling-team@fb.com"}

⁹ FACEBOOK/DUCKLING [LOGICIEL] [EN LIGNE]. 21 FEVRIER 2024. META. [CONSULTE LE 21 FEVRIER 2024]. DISPONIBLE A L'ADRESSE : [HTTPS://GITHUB.COM/FACEBOOK/DUCKLING](https://github.com/facebook/duckling) [CONSULTE LE 21 FEVRIER 2024].

Nombre	Quarante-cinq	{"value":45,"type":"value"}
Nombre Ordinal	22ème	{"value":22,"type":"value"}
Numéro de téléphone	+1 (650) 123-4567	{"value": "(+1) 6501234567"}
Température	28°C	{"value":28,"type":"value","unit":"celcius"}
Adresse Web	https://api.wit.ai/message?q=hi	{"value": "https://api.wit.ai/message?q=hi", "domain": "api.wit.ai"}
Volume	4 litres	{"value":4,"type":"value","unit":"litre"}
Quantité	3 cuillères à soupe de sucre	{"value":3,"type":"value","product":"sucre","unit":"tablespoon"}
Temps	Le 24 janvier 2024 à 9h00	{"values": [{"value": "2024-01-24T09:00:00.000-08:00", "grain": "hour", "type": "value"}], "value": "2016-12-14T09:00:00.000-08:00", "grain": "hour", "type": "value"}

Tableau 2: Tableau de dimensions de Duckling

Il est d'ailleurs possible d'utiliser Duckling dans divers langages et même d'ajouter soit-même des dimensions personnalisées.

4.3. CHATTERBOT

Chatterbot est une librairie Python Open Source créée en 2014 qui permet l’élaboration de Chatbots capables d’engager dans des conversations avec un utilisateur. Chatterbot est capable de recevoir les messages utilisateur en entrée de par divers sources telles que par appels d’API, reconnaissance vocale, entrées console, etc.

À la différence de Rasa, Chatterbot ne fait pas usage de techniques de compréhension de texte avancées telles que du NLU mais fait usage de détection de patterns et de similarités dans le texte afin de déterminer la réponse adéquate dans un cas donné à l'aide d'algorithmes de machine learning entraînés sur des conversations préparées à l'avance.

4.4. MICROSOFT BOT FRAMEWORK

Le Microsoft Bot Framework est un framework créé par Microsoft en Mars 2016 permettant de mettre en place un Chatbot conversationnel dans divers langages tels que C#, JavaScript, Python et Java (À noter que le SDK de Java n'est plus supporté depuis Novembre 2023).

Ce framework offre une panoplie de fonctionnalités similaires à Rasa comme :

- Un SDK de création de Bot
- Intégration possible de divers applications de communication/messageries comme Microsoft Teams, Slack, Telegram et bien d'autres
- Possibilité d'utiliser du NLP et plus précisément du NLU avec LUIS
- Gestion de Dialogues et Conversations
- Outils d'analyses de performances et de surveillance
- Intégration facilitée avec d'autres services Microsoft comme Azure

Ce framework n'est hélas pas open source ou gratuit et nécessite un abonnement pour en faire usage pleinement. Il existe cependant un modèle gratuit mais avec des fonctionnalités

réduites et un nombre de messages pouvant être envoyés aux divers canaux de messageries limité.

4.5. NLTK

NLTK est un package Python créé en 2001 et permettant de faire usage d'une multitude d'outils NLP. NLTK en tant que tel ne peut pas créer de Chatbots mais son usage peut être très intéressant lorsque l'on souhaite appliquer des pré-traitements à une entrée utilisateur avant de fournir ce dernier au chatbot pour accroître la précision des réponses et efficacité du bot.

4.6. OPENNLP

OpenNLP un package développé par la fondation Apache depuis 2001 et ayant pour principale utilisation, comme NLTK, l'usage des divers outils NLP qu'il propose. Cependant, il est important de noter que contrairement à NLTK qui est en Python, NLTK ne peut être qu'utilisé en Java et langages supportant l'interoperabilité avec Java comme Scala ou Kotlin. De plus, le langage Python est souvent favorisé du à la grande quantité de modèles existant et le nombre de librairies à disposition pour divers processus de machine learning.

Chapitre 5 : PRÉSENTATION DES PROTOTYPES

Afin d'évaluer certaines technologies la faisabilité du projet en faisant usage de ces dernières, j'ai réalisé quelques prototypes

5.1. PROTOTYPES NLP

Dans un premier lieu, j'ai pensé faire usage uniquement de librairies fournissant des outils NLP afin de réaliser le moteur de traitement de données par NLU qui permettra de dégager les mots-clés nécessaires à l'établissement de règles pour mettre en place un Chatbot par règles.

Pour se faire, j'ai essayé deux librairies. Une en Java : OpenNLP et l'autre en Python : NLTK.

a) JAVA

Concernant mon premier prototype en Java avec OpenNLP, j'ai dans un premier temps implémenté un Tokenizer afin de pouvoir séparer chaque mots dans mes phrases en tokens afin de faciliter la suite des opérations.

```
Phrase en entrée : Je veux réserver un terrain de tennis
Je
veux
réserver
un
terrain
de
tennis
```

Comme on le remarque, la sortie de ce programme est bien la phrase correctement tokenisée. Cependant, la raison pour laquelle je n'ai pas poussé bien plus loin le développement de ce prototype est pour la simple et bonne raison que bien qu'il existe une certaine variété de librairies, modèles de machine learning pré-entraînés et jeux de données divers; ces derniers ne sont pas en nombre assez conséquent et reviendraient à demander une charge de travail bien plus grande que si je réalisais un prototype en Python.

b) PYTHON

Après la tentative précédente en Java, j'ai donc décidé de changer le langage pour passer à Python et faire usage de la librairie NLTK.

En l'état, le prototype permet de récupérer une entrée texte ou un fichier et de traiter ce dernier afin de recevoir en sortie une décomposition de chaque phrases avec chaque mots devenus des tokens et ayant leur classe grammaticale spécifiée. De plus, un traitement qui est aussi appliqué après le traitement par POS tagging est l'application d'un NER afin de détecter quand une date se trouve dans une phrase et lorsqu'un sport comme Tennis est détecté.

```
Texte de base : "Je souhaite réserver un terrain de tennis. Je veux réserver pour le 24 Novembre 2024."
```

```
NEXT SENTENCE:
```

```
(Je, PRON) (souhaite, VERB) (réserver, VERB) (un, DET) (terrain, NOUN) (de, ADP) (tennis, SPORT) (., PUNCT)
```

```
NEXT SENTENCE:
```

```
Special case(le 24 Novembre 2024, DATE)
```

```
(Je, PRON) (veux, VERB) (réserver, VERB) (pour, ADP) (('le', 'DET'), ('24', 'NUM')) (., PUNCT)
```

Afin de parvenir à ce résultat, divers jeux de données et librairies ont été utilisés :

- Punkt Tokenizer disponible avec NLTK après téléchargement : Punkt fournit des fonctions liées à la tokenisation de phrases et mots.

- Maxent_ne_chunker disponible avec NLTK après téléchargement : Maxent permet d'effectuer du « chunking » ce qui en d'autres mots est le fait de prendre de multiples mots ayant été traités par du POS Tagging et selon des patrons donnés, ressortir un chunk étant une nouvelle entité qui est le produit de la combinaison d'un certain nombre de tokens. L'exemple dans le cas du Prototype est que dans son implémentation actuelle, une date est détectée s'il y a au moins un déterminant, un nombre, un nom et à nouveau un nombre ce qui correspond à « Le 17 Janvier 2024 » par exemple. Le résultat sera donc l'ensemble de ces tokens agrégés en une entité de type **DATE**.
- Averaged_perceptron_Tagger disponible avec NLTK après téléchargement : Le but d'un tagger est de, selon un modèle fourni au préalable, de tagger les divers tokens selon leur POS correspondant ou classe grammaticale en Français. Le modèle de POS Tagging utilisé ici est celui de Stanford étant disponible en Français parmi d'autres langues¹⁰.
- Words Corpora disponible avec NLTK après téléchargement : Bien que ce package n'est pas directement utilisé dans le prototype, il aurait servi à l'avenir à l'identification d'erreurs orthographiques dans l'entrée utilisateur grâce à l'usage de la Distance de Levenshtein ou Edit Distance qui va permettre de mesurer la distance entre deux mots étant « Le nombre de suppressions, insertions ou substitutions requises pour transformer le mot initial à celui comparé »¹¹.

¹⁰ THE STANFORD NATURAL LANGUAGE PROCESSING GROUP, [EN LIGNE]. DISPONIBLE A L'ADRESSE : [HTTPS://NLP.STANFORD.EDU/SOFTWARE/TAGGER.SHTML](https://nlp.stanford.edu/software/tagger.shtml) [CONSULTE LE 5 MARS 2024].

¹¹ MEHTA, ANSH ET AL., 2021. SPELL CORRECTION AND SUGGESTION USING LEVENSHTEIN DISTANCE.. VOL. 08, NO 09.

Ce prototype fut intéressant car il m'a permis de prendre en main diverses techniques de NLP et de mieux assimiler leur fonctionnements afin de pouvoir davantage me projeter dans la bonne direction quant à l'usage de NLP pour la réalisation de mon Chatbot. Or, je l'ai vite compris que mon Chatbot ne serait pas réalisé de zéro avec seulement des librairies NLP pour réaliser un Chatbot de règles.

Cependant, je ne laisse pas complètement de coté ces librairies et jeux de données employé car ils pourraient être très utile dans le cadre d'un pré-traitement de texte avant de le passer au Chatbot comme pour appliquer une correction de fautes d'orthographe pour réduire la possibilité que le Chatbot ne comprenne pas un message.

5.2. PROTOTYPE TAPAS

En attendant, une autre problématique que j'ai tenté de résoudre était celle de la recherche de ressources. Dans le cadre de ce projet, l'utilisateur sera éventuellement capable de demander une réservation pour une ressource donnée et que l'entrée utilisateur soit utilisée par le Chatbot pour qu'il puisse se renseigner sur ces mêmes ressources. C'est donc là que m'est venu l'idée de faire usage de TAPAS, qui comme expliqué plus tôt, permet de parcourir un tableau de données par langage naturel ou en d'autres termes, du texte.

Le langage préféré par la majorité de ceux qui ont employé TAPAS est Python de par le fait que le modèle réalisé par Google est aussi en Python.

De ce fait, j'ai réalisé un petit prototype avec l'aide d'une fondation de code venant de Christian Versloot sur divers exemples d'implémentations de modèles de Machine Learning, dont TAPAS.¹²

¹² VERSLOOT, Christian. Machine Learning for Table Parsing: TAPAS. GitHub [en ligne]. Disponible à l'adresse : <https://github.com/christianversloot/machine-learning-articles/blob/main/easy-table-parsing-with-tapas-machine-learning-and-huggingface-transformers.md> [consulté le 5 mars 2024].

L'exemple que j'ai implémenté pour tester le modèle est le suivant :

a) **DONNÉES DE TEST**

Un certain nombre de Terrains de sport sont définis dans un fichier Terrains.csv qui comporte des données comme suit :

<i>Id</i>	<i>Sport</i>	<i>Nom Terrain</i>	<i>Horaire</i>
1	Tennis	1	1
4	Tennis	2	2
5	Badminton	1	1
8	Petanque	1	1

Tableau 3: Extrait de Terrains.csv

Source : Rodrigues dos Santos Fabio

<i>Id</i>	<i>Nom</i>	<i>Lundi</i>	<i>Mardi</i>	<i>Mercredi</i>	<i>Jeudi</i>	<i>Vendredi</i>	<i>Samedi</i>	<i>Dimanche</i>
1	<i>Horaire jour</i>	<i>9h00- 16h00</i>	<i>8h30- 15h50</i>	<i>8h30- 15h50</i>	<i>9h00- 17h00</i>	<i>11h30- 14h00</i>	<i>11h30- 14h00</i>	<i>None</i>
2	<i>Horaire nuit</i>	<i>18h00- 22h00</i>	<i>None</i>	<i>None</i>				

Tableau 4: Contenu de Horaires.csv

Source : Rodrigues dos Santos Fabio

b) RÉSULTATS

Le prototype va premièrement charger dans le modèle TAPAS **Terrains.csv** avec la requête suivante “What are all the available tennis courts by name column ?”. Le résultat sortant du modèle TAPAS après traitement :

```
[['Tennis 1', 'Tennis 2', 'Tennis 3', 'Tennis 4', 'Tennis 1', 'Tennis 4']]
```

Une fois parsé et utilisé comme moyen de récupérer les données dans le fichier .csv on obtient :

	id	sport	nom	terrain	horaire
	0	1	1	Tennis 1	1
	10	11	1	Tennis 1	2
	1	2	1	Tennis 2	1
	2	3	1	Tennis 3	1
	3	4	1	Tennis 4	2
	11	12	1	Tennis 4	1

Comme on le voit, nous avons bien reçu l'ensemble des terrains de Tennis comme demandé. Après cela, la requête qui suit est la suivante : « Which horaire by id allows for a reservation on Tuesday at 18h30 ? »

Le modèle nous réponds ensuite :

```
[['horaire nuit']]
```

c) CONCLUSION

Bien que dans le cadre de ces requêtes allant de simples à un niveau de complexité supérieur, une problématique principale posée par TAPAS est que le modèle n'est pas capable, comme une base de données traditionnelle, de faire usage de clés étrangères et de références à d'autres tableaux par un identifiant unique. De ce fait, nous sommes contraints d'appliquer des traitements supplémentaires entre chaque requêtes.

De plus, la raison pour laquelle les requêtes ci-dessus sont en anglais est que bien que le modèle soit capable de comprendre les requêtes effectuées en Français, il lui arrive bien plus souvent de ne pas comprendre assez clairement la nature de la demande et de ne pas retourner le résultat attendu.

Pour finir, un des problèmes majeurs est le simple format de données car dans le cas où le Chatbot parle avec une multitude de clients, ce dernier devra modifier en continu le contenu .csv en mémoire ou le fichier lui-même ce qui ajoute un temps de traitement considérable et pourrait résulter en des soucis de cohérence et de concurrence car plus il y aurait de clients, plus le bot serait lent.

En conclusion, TAPAS est un modèle très intéressant lorsque l'on souhaite plus aisément chercher des informations dans un tableau sans avoir à effectuer des manipulations NLP diverses soi-même. Or, dans le cadre de ce projet, l'implémentation de la gestion des réservations et des ressources serait bien trop complexe et lente comparé à une implémentation plus basique avec des requêtes à une base de données.

5.3. PROTOTYPE CHATBOT

Après les essais précédents et quelques recherches, je me suis finalement penché sur Rasa Open Source en Python pour réaliser un prototype de Chatbot très simpliste.

Tout d'abord, afin de pouvoir réaliser le Chatbot il faut des données pour l'entraînement du gestionnaire de dialogues afin que le Chatbot soit capable de les reconnaître par la suite.

a) DONNÉES D'ENTRAÎNEMENT

```
- lookup: resource
examples: |
  - Tennis
  - Tenis
  - TEnnis
  - tennnis
  - Badminton
  - badminton
  - pétanque
  - petanque
  - Pétanque
  - Petanque
```

```
- intent: book_date
examples: |
  - 29 Juillet 2024
  - 05/02/2024
  - 04.11.2025
  - le 19.01.2020
  - le 02/14/2022
  - le 21 Avril 2024
  - le 12 mars 2024
```

```
- intent: book_resource
examples: |
  - Je souhaite réserver un terrain de [tennis] (resource)
  - J'aimerais réserver un terrain de [Tennis] (resource)
  - Réserve un terrain de [Tenis] (resource)
  - Je souhaite réserver un terrain de [petanque] (resource)
  - J'aimerais réserver un terrain de [Pétanque] (resource)
  - Réserve un terrain de [Petanque] (resource)
  - Je souhaite réserver un terrain de [badminton] (resource)
  - J'aimerais réserver un terrain de [Badminton] (resource)
  - Réserve un terrain de [Badminton] (resource)
```

```
- intent: book_resource+book_date
examples: |
  - Je souhaite réserver un terrain de [petanque] (resource) pour le 6 Septembre 2025
  - J'aimerais réserver un terrain de [badminton] (resource) le 28 Octobre 2024
  - Je veux réserver un terrain de [tennis] (resource) le 4 Janvier
```

Divers Intent utilisés pour la réalisation du prototype de réservation

```
- story: reservation 2
steps:
  - intent: book_resource
  - action:
    action_reserve_resource
  - intent: book_date
  - action: action reserve date
```

```
- story: reservation 3
steps:
  - intent: book_resource+book_date
  - action: action_reserve_resource
  - action: action_reserve_date
  - checkpoint: confirm_reservation
```

```
- story: confirmation_cancelled  
steps:  
- checkpoint: confirm_reservation  
- intent: cancel  
- action: action_goodbye
```

```
- story: confirmation_confirmed  
steps:  
- checkpoint:  
confirm_reservation  
- intent: confirm  
- action:  
utter_final_reservation  
- action: action_goodbye
```

Les stories principales utilisées dans le prototype

Selon les intents actuellement présents dans le prototype, le Chatbot est capable de :

- Reconnaître quelques types de salles/terrains de sports selon le sport
- Reconnaître une demande de réservation étant soit une demande de réservation + date ou une demande de réservation simple suivie d'une date
- Reconnaître des dates grâce à l'utilisation de Duckling

Grâce ensuite aux stories décrites après, le Chatbot est capable de :

- Recevoir une demande de réservation et si aucune date n'est spécifiée, le bot demandera à quelle date
- Recevoir une demande de réservation et si une date est spécifiée, le bot va procéder à la suite de la conversation directement
- Une fois qu'une des deux manières de réserver est effectuée, le bot demandera une confirmation de la réservation et le client pourra accepter ou l'annuler.

b) EXEMPLE DE CONVERSATION

```
Votre message ->Je souhaite réserver un terrain de badminton
Rasa's response -> [{"recipient_id": "user", "text": "Et à quelle date ?"}]
Votre message ->Le 12 janvier cette année
Rasa's response -> [{"recipient_id": "user", "text": "Souhaitez-vous donc réserver pour le 2024-01-12T00:00:00.000-08:00 un terrain de badminton ?"}]
Votre message ->oui
Rasa's response -> [{"recipient_id": "user", "text": "Vous avez bien réservé pour le 2024-01-12T00:00:00.000-08:00 un terrain de badminton"}, {"recipient_id": "user", "text": "Au revoir"}]
```

```
Votre message ->Je souhaite réserver un terrain de pétanque le 29/05/2024
Rasa's response -> [{"recipient_id": "user", "text": "Souhaitez-vous donc réserver pour le 2024-05-29T00:00:00.000-07:00 un terrain de pétanque ?"}]
Votre message ->Oui
Rasa's response -> [{"recipient_id": "user", "text": "Vous avez bien réservé pour le 2024-05-29T00:00:00.000-07:00 un terrain de pétanque"}, {"recipient_id": "user", "text": "Au revoir"}]
```

Deux exemples de conversation avec le Chatbot du prototype

De par ces deux exemples de conversation avec le bot, on peut remarquer dans un premier temps le fait que le parsing des dates est correctement géré par Duckling de manière assez efficace, même quand une date comme “Le 12 janvier cette année” est fournie et que Duckling arrive à parser cette dernière correctement en “12 janvier 2024”.

De plus, on remarque bien avec les divers messages de rappels et de confirmation que le Chatbot enregistre bien les choix de l’utilisateur et les utilise pour déterminer les prochains messages qui vont suivre comme le fait d’inclure ou non la date de la réservation directement dans le message initial.

c) CONCLUSION

Avec ce prototype, même simpliste, de Chatbot conversationnel avec Rasa Open Source, on remarque à quel point avec un petit jeu de données il est possible de créer tout un dialogue de réservation de terrain de sport avec des branchements selon comment l’utilisateur réponds.

De plus, ce qui n'est pas montré ici mais est présent dans le code du prototype, actuellement le bot ne renvoi que des messages à l'utilisateur mais à chacune des actions définies dans les stories, le bot pourrait aller vérifier dans la base de données et selon ce qu'il reçoit mettre à jour des slots et influencer la conversation dans le cas ou par exemple une plage d'horaire n'est pas disponible ou s'il n'y a pas de terrains du type demandé par l'utilisateur etc. Avec l'implémentation de scripts de parage de ressources fournies en CSV il serait très facile de peupler les données du Chatbot afin d'ajouter davantage de ressources et de rendre ce dernier facilement modulable dans cet aspect.

Chapitre 6 : CHOIX TECHNOLOGIQUES

CONCLUSION (STYLE « TITRE 1 »)

Votre texte, votre texte (style « Corps de texte, interligne 1,5 »).

ANNEXES (STYLE « TITRE 1 »)

Imprimer idéalement cette page sur une page de couleur

Chaque annexe doit commencer sur une nouvelle page et doit être numérotée :

Annexe 1 puis Annexe 2, etc.

ANNEXE 1

ANNEXE 2

ANNEXE 3

RÉFÉRENCES DOCUMENTAIRES (STYLE « TITRE 1 »)