الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية التكنولوجيا

قسم: الإعلام الآلي

Mémoire de Master

Spécialité : Modélisation Informatique des Connaissances et du Raisonnement

Thème

Chatbot pour la relève des dérangements Cas pratique : Algérie Télécom

Présenté par :

AID Zahira

ZIANE Benameur

Dirigé par :

Dr ZAHAF Ahmed

Année universitaire 2022-2023

Remerciements

C'est pour nous un plaisir autant qu'un devoir de remercier toutes les personnes qui ont pu contribuer de près ou de loin à l'élaboration de ce projet, qui nous ont aidé, nous ont soutenu et ont fait en sorte que ce travail ait lieu

En premier lieu, Nous tenons à remercier notre encadreur : Dr ''ZAHAF AHMED'' pour nous avoir encouragés dans cette démarche et sa patience, sa gentillesse, ses conseils et ses orientations pour mener à bien notre travail.

Nos remerciements vont également aux membres du jury pour avoir accepté d'examiner notre travail, pour le temps qu'ils ont consacré pour lire ce mémoire et de l'enrichir par leurs propositions.

On désire aussi remercier notre collègue : « Dr CHALLOULI MOHAMED » pour sa gentillesse, ses conseils et ses orientations son soutien moral et intellectuel.

Nous souhaitons aussi adresser nos remerciements au corps enseignant et administratif de
L'Université Dr. Tahar MOULAY SAIDA
Faculté : Technologie
Département : Informatique
Qui a contribué à la réussite de nos études en Master II

A tous, ''MERCI'

AID Zahira et ZIANE Benameur

Dédicace

A mon cher père « AID Ahmed » que dieu bénisse son âme, pour ces conseils constants dans mon esprit et son image de combattant qui n'abandonne jamais,

A mon cher époux « BOUKERCHE Mohammed el Kebir », aucun mot ne saurait exprimer mon profond attachement et ma reconnaissance pour ton encouragement et ta présence dans toutes les circonstances à mes cotés

A ma cher mère « KIHEL Zoulilha » qui n'arrête pas de prier pour mon succès et ma réussite, Que dieu te garde en bonne santé,

A mes enfants « BOUKERCHE Zakaria », « BOUKERCHE Fatima Zahraa », « BOUKERCHE Youcef » et « BOUKERCHE Ahmed Houcine » Pour leurs appuis et leurs encouragements,

A ma chère sœur « AID Rafika » qui me donner le courage de continuer et de ne jamais abandonner, A mes chères frères et sœurs,

A toutes mes amies, pour leurs appuis et leurs encouragements,

Que ce travail soit l'accomplissement de vos vœux tant allégués, et le fruit de votre soutien infaillible, Merci d'être toujours là pour moi.

AID Zahira

Dédicace

Avec l'expression de ma reconnaissance, je dédie ce modeste travail

A mes parents, ma précieuse offre du Dieu, à qui doit ma vie, ma réussite, à qui m'ont doté d'une éducation digne, leur amour a fait de moi ce que je suis aujourd'hui :

Particulièrement à mon cher père « Zitouni », pour le goût à l'effort qu'il a suscité en moi, de sa rigueur, à vous tout mon respect

A toi ma chère mère « Mansoura », à celle qui a souffert sans me laisser souffrir, ceci est ma profonde gratitude pour ton éternel amour, que ce rapport soit le meilleur cadeau.

A toi ma chère épouse « Aïcha », pour ton soutien, tes encouragements et tes conseils, à celle qui n'a épargné aucun effort pour me rendre heureux.

A vous mes adorables enfants « Hadil » et « Houssem » qui procurent la joie et le bonheur dans notre vie, ils m'ont chaleureusement supporté tout au long de mon parcours.

A mon frère « Abdelkrim » et mes sœurs « Doulat » et « Khadidja » qui m'avez encouragé et soutenu toute au long de mes études.

A mes collègues de travail d'Algérie Télécom qui m'encouragent à chaque instant, et à qui je souhaite plus de succès

Sans oublier mon binôme « Mme Boukerche Aid Zahira » pour son soutien moral, sa patience et sa compréhension tout au long de ce projet et des études.

Que Dieu vous protège tous et vous offre la santé, la chance et le bonheur.

ZIANE Benameur

Table des matières

Tables des figures

Liste des acronymes

IA : Intelligente Artificielle

NLP : Natural Language Processing

ALICE : Artificial Linguistic Internet Computer Entity

AOL : America OnLine

MSN: MicroSoft Network

SIRI : Speech Interpretation and Recognition Interface

GPT : Generative Pre-trained Transformer

AIML : Artificial Intelligence Markup Language

XML : Extensible Markup Language

SVM : Support Vector Machine

ANN : Artificial Neural Network

RNN : Recurrent Neural Network

TAL : Traitement Automatique des Langues

NLU : Natural Language Understanding

NER : Named-Entity Recognition

LSTM : Long-Short-Term-Memory

FAQ : Frequently Asked Questions

SEQ 2SEQ : Sequence to Sequence

VHRED : Variable Hierarchical Recurrent Encoder-Decoder

BRNN : Bidirectional Recurrent Neural Networks

ADAM : Algorithm Development and Mining

AT : Algérie Télécom

VPN : Virtual Private Network

TPL : telephone

FTTH : Fiber to the Home

FTTx : Fiber-to-the-x

ADSL : Asymmetric Digital Subscriber Line

4GLTE : quatrième génération Long Term Evolution

RTC : Réseau Téléphonique Commuté

Atbot : Algérie Télécom Bot

KB : knowledge base

RDB : Relational Data Base

WORDNET : the machine-readable lexical databases modeled

NLG : Natural Language Generation

RDF : Resource Description Framework

URI : Uniform Resource Identifier

SPARQL : Simple Protocol And RDF Query Language

RDFS : Resource Description Framework Schema

OWL : Web Ontologie Language

APACHE :Serveur HTTP

API : Application Programming Interface

IBM : International Business Machines Corporation

PNL : Programmation Neuro Linguistique

TTS : Text-to-Speech

Introduction

1. Introduction :

Le contexte et la justification du sujet "Réalisation d'un chatbot de relève des dérangements - Cas pratique Algérie Télécom" mettent en évidence les raisons pour lesquelles ce projet est pertinent et nécessaire.

. Contexte et justification du sujet :

Ce Projet met en évidence l'importance du secteur des télécommunications et son impact sur la vie quotidienne des individus et des entreprises. Il aborde également le rôle d'Algérie Télécom en tant que principal fournisseur de services de télécommunications en Algérie. Aussi nous soulignons l'importance d'améliorer le processus de relève des dérangements chez Algérie Télécom en utilisant un chatbot. Nous mettons en évidence les avantages potentiels d'un chatbot dans la résolution rapide et efficace des problèmes des clients. Parmi les justifications possibles, on peut citer l'amélioration de l'expérience client, la réduction des délais de traitement et des coûts opérationnels, ainsi que l'optimisation des ressources humaines en redirigeant les tâches de routine vers un système automatisé.

. Problématique de l'étude :

La problématique principale s'agit de mettre en évidence les défis et les difficultés auxquels sont confrontés les clients d'Algérie Télécom lorsqu'ils doivent signaler un dérangement ou une panne dans leurs services. Il peut être question de lenteur dans le processus de signalement, d'attente prolongée pour obtenir une assistance ou de manque de disponibilité des canaux de communication traditionnels.

. Objectifs de recherche :

Les objectifs spécifiques de ce projet est de concevoir et de développer un chatbot capable de relever et de résoudre les dérangements courants, d'évaluer l'efficacité et la satisfaction des utilisateurs avec le chatbot, de comparer les résultats avec les canaux de support client traditionnels, ou encore d'analyser les avantages et les limites de l'utilisation d'un chatbot dans ce contexte.

le contexte et la justification du sujet "Réalisation d'un chatbot de relève des dérangements - Cas pratique Algérie Télécom" mettent en évidence les problèmes actuels du processus de relève des dérangements chez Algérie Télécom et justifient l'importance d'introduire un chatbot pour améliorer l'efficacité et la satisfaction des clients.

Chapitre I :
Etat de l'art et revu de littérature

1.1. Définition des chatbots et leur utilisation dans les services clients :

L'intelligence artificielle (IA) a influencé la façon dont nous nous engageons dans notre activités quotidiennes en concevant et en évaluant des applications et des dispositifs avancés, appelés agents intelligents, qui peuvent remplir diverses fonctions. Un chatbot est un programme d'intelligence artificielle et un modèle d'interaction homme-ordinateur

Un chatbot est « un programme informatique conçu pour simuler une conversation avec des utilisateurs humains, en particulier sur Internet ». Il utilise le traitement du langage naturel (NLP) et l'analyse des sentiments pour communiquer en langage humain par texte ou parole orale avec des humains. Les entités de conversation artificielles, les agents interactifs, les robots intelligents et les assistants numériques sont également appelés chatbots.

Les chatbots sont utiles dans divers autres domaines de l'éducation, des affaires et du commerce électronique, de la santé, des télécommunications et du divertissement. La productivité est la motivation la plus importante pour les utilisateurs de chatbot. Dans les entreprises, les chatbots sont devenus si courants car ils réduisent les coûts de service et peuvent gérer plusieurs clients simultanément Le succès d'un chatbot dépend de son comportement, de son apparence et le niveau auquel il est humanisé ceci dépend des caractéristiques visuelles, de la proximité de son nom avec une personne, de sa personnalité et de son efficacité à gérer le langage humain. L'émotion est un autre aspect essentiel pour humaniser un chatbot, et il y a eu de nombreuses approches pour construire un chatbot émotionnellement conscient.

1.2. Histoire du Chatbot :

Figure N°01 : John von Neumann avec l'ordinateur IAS vers 1951.

En 1950, Alan Turing se demandait si un programme informatique pouvait parler à un groupe de personnes sans se rendre compte que leur interlocuteur était artificiel. Le premier chatbot avec le nom ELIZA a été construit en 1966[1].

Figure N°02 : Aperçus d'ELIZA

ELIZA simulait l'opération d'un psychothérapeute, renvoyant les phrases de l'utilisateur sous la forme interrogative.

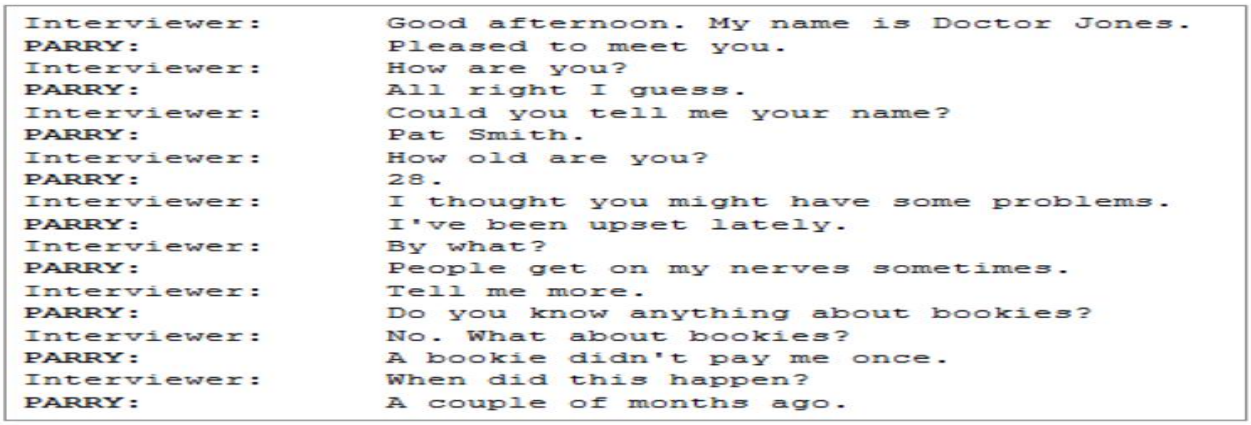


Figure N°03 : Aperçus de PARRY

En 1972, PARRY a agi comme un patient atteint de schizophrénie. Il est considéré comme plus avancé qu'ELIZA car il est censé avoir une "personnalité" et une meilleure structure de contrôle. Il définit ses réponses sur la base d'un système d'hypothèses et de « réponses émotionnelles » [2].

Figure N°04 : Aperçus de JABBERWACKY

L'Intelligence Artificielle est utilisée dans le domaine des chatbots avec la construction de Jabberwacky en 1988. Il a été créé par le développeur Rollo Carpenter en utilisant un langage basé sur des feuilles de calcul qui a facilité le développement de chatbots.

Le terme Chatterbot a été mentionné pour la première fois en 1991. Il s'agissait d'un joueur artificiel TINYMUD dont la fonction principale était de discuter. De nombreux vrais joueurs humains semblaient préférer parler à Chatterbot plutôt qu'à un vrai joueur. En 1992, un chatbot a été conçu pour afficher les voix numérisées que les cartes son étaient capables de produire. Il a joué le rôle d'un psychologue sans aucune sorte d'interaction compliquée [2].

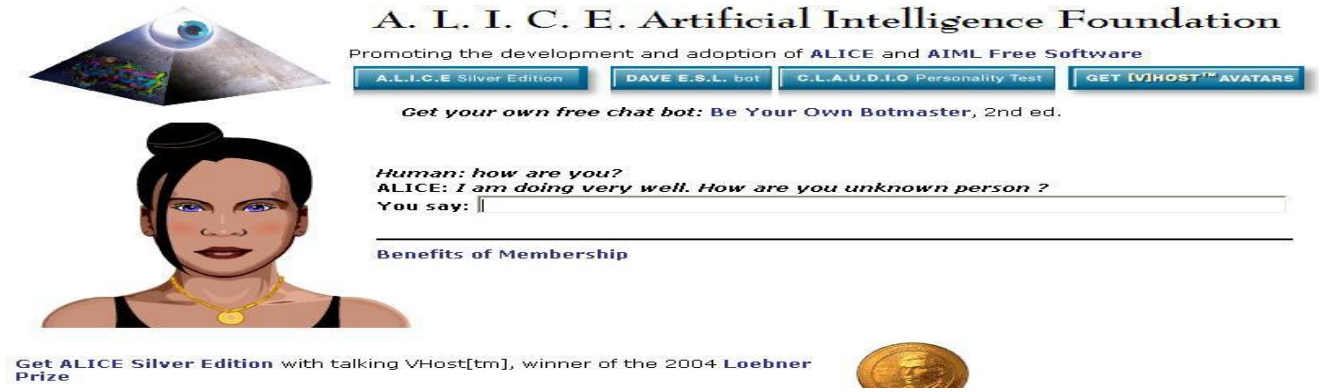


Figure N°05 : Aperçus d'A.L.I.C.E

Richard Wallace, un informaticien, a développé son programme de chatterbot A.L.I.C.E (Artificial Linguistic Internet Computer Entity) en 1995. Ce programme de chatterbot a remporté trois fois le prestigieux prix Loebner pour être considéré comme le plus humain des chatbots de

son temps, mais il n'a jamais passé le test de penser aussi intelligemment que les humains. La base de connaissances d'ALICE se composait d'environ 41 000 modèles et modèles associés, un grand nombre par rapport à ELIZA qui n'avait que 200 mots clés et règles. Cependant, ALICE n'avait pas de caractéristiques intelligentes et ne pouvait pas générer de réponses de type humain exprimant des émotions ou des attitudes [3].

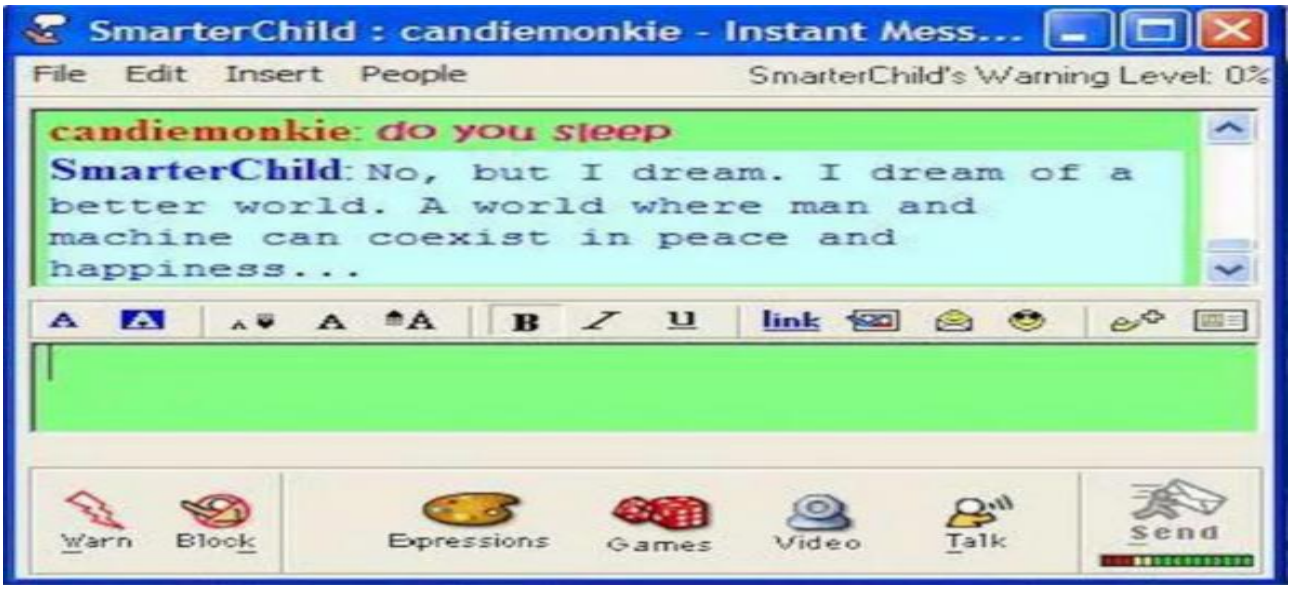


Figure N°06 : Aperçus de SmarterChild

En 2001, il y a eu une réelle évolution de la technologie des chatbots avec le développement de SmarterChild, qui était disponible sur des Messengers comme AOL et MSN. C'était la première fois qu'un chatbot pouvait aider les gens dans des tâches quotidiennes pratiques, car il pouvait récupérer des informations dans des bases de données sur les horaires des films, les résultats sportifs, les cours des actions, les actualités et la météo [4].

Le développement des chatbots d'IA est allé encore plus loin avec la création d'assistants vocaux personnels intelligents, intégrés aux smartphones ou aux haut-parleurs domestiques dédiés, qui comprenaient les commandes vocales, parlaient par des voix numériques et géraient des tâches telles que surveillance des appareils automatisés domestiques, des calendriers, des e-mails et autres

Figure N°07 : Logo d'Alexa, Google Assistante, Siri, IBMWATSON

Apple Siri, IBM Watson, Google Assistant, Microsoft Cortana et Amazon Alexa sont les assistants vocaux les plus populaires. Ils se connectent à Internet et, contrairement à leurs prédécesseurs, ils créent rapidement des réponses significatives [2].

Selon la figure N°08 une augmentation croissante de l'utilisation des chatbots a été observée, en particulier après 2016 [1].

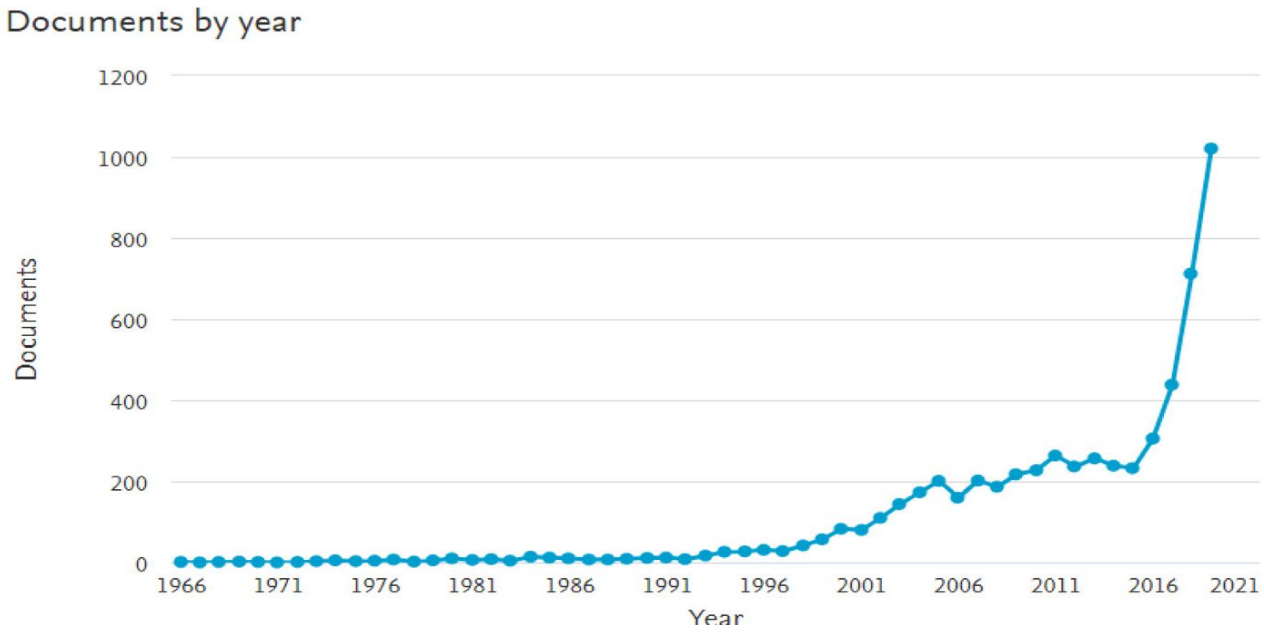


Figure N°08 : l'Utilisation des chatbot (Documents) par an

Selon la figure 09, le pays qui a montré le plus d'intérêt pour la recherche sur les chatbots est les États-Unis, tandis que le Royaume-Uni et le Japon suivent avec moins d'un tiers du nombre d'articles publiés aux États-Unis [1].

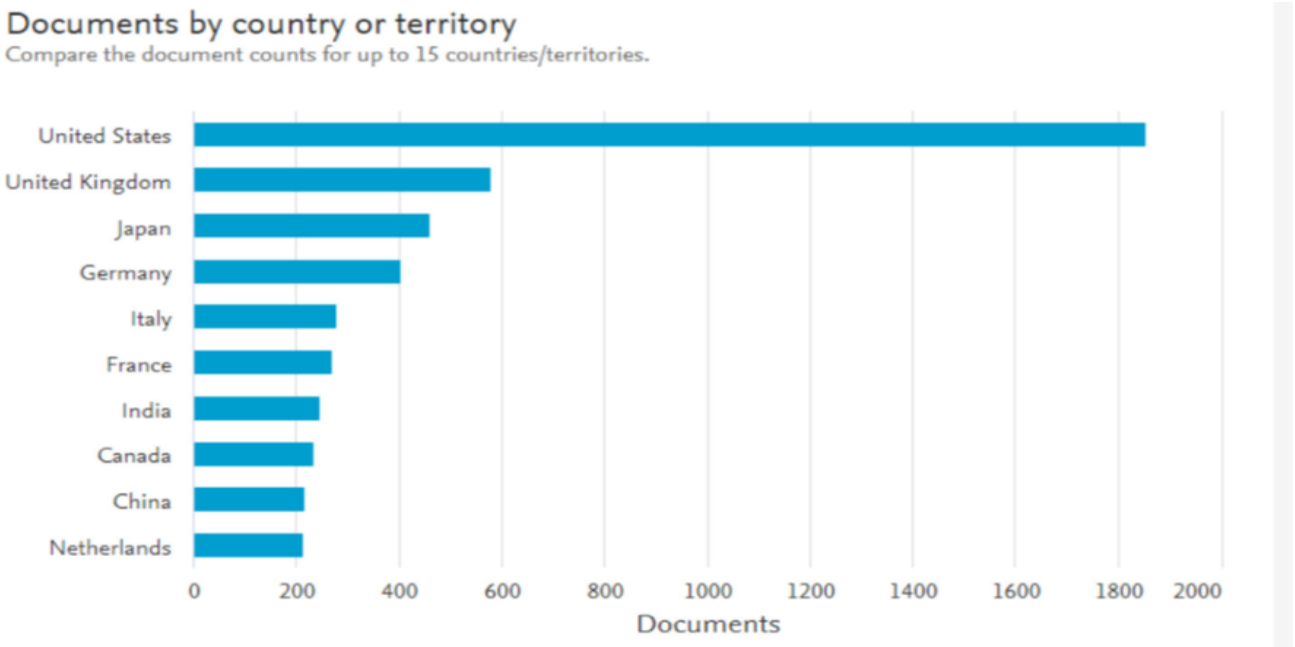


Figure N°09 : l'Utilisation des chatbot (Documents) par pays

Le GPT et l'intelligence artificielle

Le GTP est défini comme le plus grand réseau neuronal artificiel jamais créé à ce jour. Cette percée présentée par la société de recherche et de déploiement en intelligence artificielle (IA) OpenAI a entraîné un coût de plus de 4 millions de dollars pour Elon Musk et Sam Altman. Néanmoins, l'investissement s'est avéré rentable, car il ouvre un monde de possibilités pour l'IA au-delà de notre imagination actuelle.

Le GPT est l'abréviation de Generative Pre-training Transformer. Il s'agit d'un modèle de Deep Learning composé d'algorithmes capables de reconnaître des modèles de données et qui peuvent également apprendre par le biais d'exemples. À ce titre, il est considéré comme un réseau neuronal artificiel avec une mémoire à long terme. Le GTP utilise ses algorithmes pour générer du texte. Ces algorithmes ont été préalablement formés à l'aide d'une énorme base de données. Il évalue et traite toutes les données qu'il reçoit afin de combler les lacunes en matière d'information. Le GTP a été décrit comme la percée la plus importante et la plus utile en matière d'intelligence artificielle depuis des années. Il semble être – bien qu'il soit encore dans sa version bêta – le modèle d'intelligence artificielle le plus puissant actuellement disponible. Il est capable de générer des textes entiers en commençant par une seule phrase, puis en complétant le reste de l'écriture. Pour ce faire, il traite plus de 175 milliards de paramètres. C'est un fait très pertinent. Le GPT peut traduire des textes dans d'autres langues et les adapter à différents styles d'écriture, comme celui d'un article de journal, d'un roman de fiction, etc.

Il peut également écrire de la poésie ou nous livrer la meilleure réponse à toute question que nous lui posons. Le GPT peut s'adapter à tout ce qui est structuré comme langage : il peut répondre à des questions, rédiger des essais, résumer de longs textes, traduire, prendre des notes et même écrire du code informatique. Le GPT peut également programmer. À la grande surprise de tous, on a découvert qu'il est capable d'utiliser un plug-in pour Figma, un outil logiciel couramment utilisé dans la conception d'applications et de sites web. Cette caractéristique pourrait avoir des implications considérables sur la manière dont les logiciels seront développés à l'avenir. La quantité de choses qu'il est capable de faire peut sembler incroyable, mais ses capacités potentielles sont encore plus stupéfiantes [5].

Afin de le former et d'atteindre une capacité opérationnelle, le GPT a reçu des informations allant des textes de Wikipedia sélectionnés par OpenAI à environ 750 Go du corpus CommonCrawl. Ce corpus est un ensemble de données collectées en parcourant l'Internet et qui est accessible au grand public. En effet, un grand nombre de ressources informatiques et environ 4,6 millions de dollars ont été investis rien que pour faire suivre cette formation à GPT. Sa structure algorithmique est conçue pour prendre en compte une entrée linguistique et fournir comme résultat sa meilleure prédiction de ce qui serait le message le plus utile pour l'utilisateur concernant une telle entrée. Le GPT peut faire ces prédictions grâce à sa formation exhaustive avec une base de données aussi importante. C'est l'aspect clé qui le différencie des autres algorithmes qui ne sont pas capables de faire de telles prédictions. Pour élaborer des textes et des phrases, il utilise une approche d'analyse sémantique qui va au-delà de la signification des mots et prend également en compte la façon dont leur combinaison avec d'autres mots affecte leur signification en fonction du contexte global dans lequel ils se trouvent.

L'apprentissage du GPT est connu sous le nom d'apprentissage non supervisé. Cela signifie qu'il n'a pas reçu de retour d'information lui permettant de savoir si ses réponses sont correctes

ou incorrectes pendant sa formation. Le GPT obtient toutes les informations dont il a besoin en analysant les textes composant sa base de données.

Lorsqu'il commence une nouvelle tâche linguistique, il se trompe des millions de fois au début, mais finit par trouver le mot correct. GPT découvrira que son choix est le « bon » choix en vérifiant ses données d'entrée originales. Lorsqu'il sera certain d'avoir trouvé la bonne sortie, il attribuera un « poids » au processus algorithmique qui a produit le bon résultat. De cette façon, il apprend progressivement quels sont les processus les plus susceptibles de fournir des réponses correctes.

Certains problèmes que les spécialistes de l'intelligence artificielle ont mis en garde concernent la capacité épouvantable de produire de fausses nouvelles en masse. Ces algorithmes pourraient produire ce genre de nouvelles et surcharger les réseaux, provoquant une désinformation généralisée sans que nous nous rendions à peine compte de ce qui se passe. Vous pensez peut-être pouvoir distinguer les textes écrits par des machines de leurs homologues créés par l'homme ? Une étude réalisée par Adrian Yijie Xu a donné un résultat surprenant :

« Seuls 52% des lecteurs détectent quels textes ont été créés par GPT. »

Par conséquent, une partie importante de la population serait vulnérable à ces fausses nouvelles artificielles, les considérant vraies et contribuant à la désinformation générale. Un autre problème que pose cette technologie est qu'il s'agit d'un outil très coûteux, car il nécessite une énorme puissance de calcul pour pouvoir fonctionner. Par conséquent, son utilisation est limitée à un très petit nombre d'entreprises qui peuvent se le permettre.

Open AI n'a pas révélé tous les détails du fonctionnement de ses algorithmes, si bien que quiconque s'appuie sur le GTP pour obtenir des réponses ou développer ses produits est en quelque sorte aveuglé et ne sait pas exactement comment les informations récupérées ont été obtenues ou si l'on peut vraiment s'y fier.

Le système est prometteur, mais il n'est pas encore parfait : il peut élaborer des textes courts ou des applications de base, mais les résultats qu'il donne pour des tâches plus complexes relèvent davantage du charabia que d'une véritable réponse utile [5].

1.3. Analyse des différentes approches et technologies utilisées :

Il existe deux approches pour le développement d'un chatbot en fonction des algorithmes et des techniques adoptées : l'approche de correspondance de motifs et l'approche d'apprentissage automatique [1].

1.3.1. Approches de correspondance de modèles :

Les chatbots basés sur des règles associent l'entrée de l'utilisateur à un modèle de règle et sélectionnent une réponse prédéfinie à partir d'un ensemble de réponses à l'aide d'algorithmes de correspondance de modèles. Le contexte peut également contribuer à la sélection des règles et au format de la réponse. ELIZA et son successeur ALICE ont été les premiers chatbots à utiliser le pattern **matching**. Dans le même temps, alors que PARRY, PC Therapist III, Chatterbot in ''TinyMUD'', TIPS, FRED, CONVERSE, HEX, Albert et Jabberwacky utilisent cette technique.

L'inconvénient de l'Approches de correspondance de modèles est que les réponses sont automatisées, répétées et n'ont pas l'originalité et la spontanéité de la réponse humaine. D'autre part, le temps de réponse est rapide, car un examen syntaxique ou sémantique plus approfondi du texte d'entrée n'est pas effectué [1].

Trois des langages les plus courants pour la mise en œuvre de chatbots avec l'approche de correspondance de modèles sont décrits et comparés dans leurs fonctionnalités de base. Ces langages sont :

- Langage de balisage de l'intelligence artificielle (AIML)

Au cours des années 1995 à 2000, les développeurs ont créé le langage de balisage de l'intelligence artificielle (AIML), pour construire la base de connaissances des chatbots qui adoptent l'approche Pattern Matching. Il est basé sur XML et open-source. ALICE a été le premier chatbot avec une base de connaissances implémentée dans le langage AIML. Grâce à sa convivialité, sa facilité d'apprentissage et d'exécution et la disponibilité de collections AIML pré-écrites, AIML est le langage de chatbot le plus utilisé [1].

- RiveScript

RiveScript (langage de script d'intelligence artificielle— riveScript.com), créé en 2009, est un langage de script basé sur des lignes implémentant la base de connaissances dans des chatbots basés sur des règles. Il est open source et dispose d'interfaces disponibles pour de nombreux langages de programmation comme Java et Python.

Dans la syntaxe de RiveScript, le symbole « + » indique une entrée de l'utilisateur, tandis que « - » désigne la réponse du chatbot. L'interpréteur fait correspondre l'entrée de l'utilisateur avec les réponses stockées et détermine la réaction la plus appropriée à l'entrée de l'utilisateur [1].

- Chatscript

Sorti en 2011, et c'est un système expert pour développer des chatbots basés sur des règles avec un langage de script open source très compact. Il fait correspondre les entrées de l'utilisateur aux sorties du chatbot à l'aide de la correspondance de modèles. Un tagueur et un analyseur intégrés analysent l'entrée de l'utilisateur et l'améliorent en termes de grammaire, de syntaxe, et sémantique. ChatScript utilise des concepts qui sont des collections de mots similaires concernant le sens et d'autres parties du discours. Les bases de données de concepts existantes peuvent être utilisées directement par les développeurs facilitant la création d'un chatbot. Certains chatbots implémentés avec Chatscript sont Suzette, Rosette, Chip Vivant et Mistsuku [1].

En revanche, le principal inconvénient d'***AIML*** est que l'auteur doit écrire un modèle pour chaque réponse possible de l'utilisateur ce qui aide le chatbot à répondre rapidement et facilement. ***AIML*** est facile à apprendre et à mettre en œuvre, mais les connaissances sont présentées comme une instance dans les fichiers ***AIML*** ; par conséquent, il n'existe aucune disposition fiable pour la gestion des données à grande échelle. De plus, ***AIML*** est une série de mots correspondants basée sur des règles qui produit soit une réponse entièrement contenue, soit une substitution de mot d'entrée et est très inefficace lorsqu'il s'agit d'adresser de grandes bases de connaissances. Si des connaissances sont créées à partir de données obtenues sur

Internet, elles doivent être mises à jour périodiquement, car des mises à jour automatiques ne peuvent pas être effectuées. le principal défi est la grande quantité de données que le développeur doit saisir manuellement pour construire un chatbot fonctionnel, ***RiveSript*** offre des fonctionnalités intégrées supplémentaires et plus de balises qu'***AIML***. Plus précisément, il n'y a pas besoin de fichiers de configuration supplémentaires pour définir des informations sur le chatbot, par exemple, son nom, qui est nécessaire pour ***AIML***. De plus, il applique le principe d'héritage dans ses sujets, il y a des réponses aléatoires pondérées et des macros d'objets.

1.3.2. Approches d'apprentissage automatique :

Les chatbots qui adoptent des approches d'apprentissage automatique au lieu de la correspondance de modèles extraient le contenu de l'entrée de l'utilisateur à l'aide du traitement du langage naturel (NLP) et disposent de la capacité d'apprendre des conversations. Ils prennent en compte l'ensemble du contexte de dialogue, pas seulement le tour actuel, et ne nécessitent pas de réponse prédéfinie pour chaque entrée utilisateur possible.

En générale, ils ont besoin d'un ensemble de formation complet, dont la découverte peut constituer une difficulté cruciale car les ensembles de données disponibles peuvent être inadéquat. Les réseaux de neurones artificiels (ANN) sont utilisés pour la mise en œuvre de ces chatbots

1.3.2.1. Machine et Deep Learning :

Le Machine Learning et les chatbots ont une relation étroite et complémentaire. Le Machine Learning permet d'améliorer les capacités des chatbots en leur permettant d'apprendre à partir des données et de s'adapter aux interactions avec les utilisateurs.

La relation entre le Machine Learning et les chatbots :

1. Formation du modèle : Le Machine Learning est utilisé pour entraîner des modèles qui permettent aux chatbots de comprendre et de répondre aux requêtes des utilisateurs. Les modèles sont généralement formés à l'aide de données d'entraînement qui contiennent des exemples de questions et de réponses.

2. Classification des intentions : Les chatbots utilisent des techniques de Machine Learning, comme la classification, pour comprendre les intentions des utilisateurs. Cela leur permet de déterminer si un utilisateur pose une question, fait une demande de service ou exprime une plainte, par exemple.

3. Génération de réponses : Le Machine Learning peut être utilisé pour améliorer la génération de réponses par les chatbots. Les modèles d'apprentissage automatique peuvent apprendre à partir de vastes ensembles de données de conversations humaines pour générer des réponses plus naturelles et cohérentes.

4. Amélioration continue : Les chatbots peuvent utiliser le Machine Learning pour s'améliorer continuellement au fil du temps. En analysant les interactions passées avec les utilisateurs, les chatbots peuvent identifier les erreurs et les lacunes dans leurs réponses, ce qui leur permet de s'ajuster et de fournir des réponses plus précises et pertinentes à l'avenir.

5. Personnalisation de l'expérience utilisateur : Le Machine Learning permet aux chatbots de fournir une expérience utilisateur plus personnalisée. En analysant les données des utilisateurs, tels que leurs préférences et leurs comportements, les chatbots peuvent adapter leurs réponses et leurs recommandations pour répondre aux besoins spécifiques de chaque utilisateur.

Le Machine Learning joue un rôle essentiel dans le développement et l'amélioration des chatbots. Il leur permet d'apprendre à partir des données, de comprendre les intentions des utilisateurs, de générer des réponses plus naturelles et de fournir une expérience utilisateur personnalisée.

Le Deep Learning est une branche spécifique du Machine Learning qui utilise des réseaux de neurones profonds pour résoudre des problèmes complexes. Dans le contexte des chatbots, le Deep Learning peut être utilisé pour améliorer les performances et les capacités de compréhension des chatbots [6].

La relation entre le Deep Learning et les chatbots :

1. Compréhension du langage naturel : Le Deep Learning permet aux chatbots de mieux comprendre le langage naturel utilisé par les utilisateurs. En utilisant des réseaux de neurones profonds, les chatbots peuvent apprendre à reconnaître et à interpréter les structures, les significations et les nuances du langage humain, ce qui leur permet de répondre de manière plus précise et pertinente aux requêtes des utilisateurs.

2. Traitement automatique du langage naturel (NLP) : Le Deep Learning est utilisé dans le développement de modèles de NLP avancés pour les chatbots. Ces modèles peuvent comprendre la sémantique, la syntaxe, la contextualité et d'autres aspects du langage pour analyser les questions et les commentaires des utilisateurs et générer des réponses appropriées.

3. Génération de réponses plus naturelles : Les chatbots utilisant le Deep Learning peuvent apprendre à générer des réponses plus naturelles et cohérentes. En entraînant des modèles de génération de texte basés sur des réseaux de neurones profonds, les chatbots peuvent produire des réponses qui imitent davantage le style et le ton humains.

4. Adaptation aux variations du langage : Le Deep Learning permet aux chatbots de s'adapter à la variété du langage utilisé par les utilisateurs. En analysant de grandes quantités de données textuelles, les chatbots peuvent apprendre les variations du langage, y compris les abréviations, les fautes de frappe, les expressions informelles, etc., ce qui leur permet de mieux comprendre et de répondre aux messages des utilisateurs.

5. Amélioration continue : Le Deep Learning permet aux chatbots d'apprendre et de s'améliorer continuellement à mesure qu'ils interagissent avec les utilisateurs. Les chatbots peuvent collecter des données sur les interactions passées et les utiliser pour mettre à jour et affiner leurs modèles, ce qui leur permet de fournir des réponses de plus en plus précises et personnalisées.

Le Deep Learning joue un rôle clé dans l'amélioration des capacités de compréhension et de génération de réponses des chatbots. Il leur permet de mieux comprendre le langage naturel, de générer des réponses plus naturelles et d'adapter leur comportement aux variations du langage.

Cela conduit à une meilleure expérience utilisateur et à des interactions plus fluides avec les chatbots [6].

L'utilisation de Machine et Deep learning :

Le chatbot peut utiliser à la fois le machine learning et le deep learning, mais cela dépend de la conception et de la complexité du chatbot :

1. Le machine learning est une approche plus générale qui implique l'utilisation d'algorithmes et de modèles statistiques pour permettre au chatbot d'apprendre à partir des données et d'améliorer ses performances au fil du temps. Dans le contexte du chatbot, le machine learning peut être utilisé pour des tâches telles que la classification des intentions des utilisateurs, l'analyse des sentiments, la reconnaissance de la parole, etc. Les modèles de machine learning utilisés peuvent inclure des algorithmes tels que les arbres de décision, les forêts aléatoires, les SVM (Support Vector Machines), etc.

2. Le deep learning est une branche spécifique du machine learning qui utilise des réseaux de neurones profonds pour modéliser et comprendre des structures complexes des données. Dans le contexte du chatbot, le deep learning peut être utilisé pour des tâches de traitement du langage naturel avancées, telles que la génération de texte, la traduction automatique, la compréhension du langage naturel, etc. Les réseaux de neurones profonds, tels que les réseaux de neurones récurrents (RNN) et les réseaux de neurones transformer (Transformer), sont couramment utilisés dans le deep learning pour ces tâches.

Il est important de noter que le deep learning est souvent utilisé lorsque le chatbot nécessite une compréhension plus avancée du langage naturel et une génération de réponses plus sophistiquée. Cependant, l'utilisation du deep learning dépend également de la disponibilité de données d'entraînement suffisantes et de la complexité du problème à résoudre. Dans certains cas, des modèles de machine learning plus simples peuvent être utilisés avec succès pour des tâches spécifiques de chatbot.

Le chatbot peut utiliser à la fois le machine learning et le deep learning, selon les besoins spécifiques du projet et les capacités requises pour comprendre et générer des réponses.

1.3.2.2. Traitement du langage naturel (TAL) :

Le traitement du langage naturel (TAL) est un domaine de l'intelligence artificielle qui examine comment les systèmes informatiques peuvent interpréter et contrôler le langage naturel concernant le texte ou la parole. Des informations sont collectées sur la compréhension et l'utilisation du langage humain afin de créer les techniques appropriées pour que les systèmes informatiques gèrent le langage humain et effectuent diverses tâches [7].

La plupart des techniques de TAL reposent sur l'apprentissage automatique. Ils se composent de la compréhension du langage naturel, qui fait évoluer la mission de comprendre un texte, et de la génération du langage naturel , qui introduit la responsabilité de générer le texte couramment menée par les réseaux de neurones ANN [1].

1.3.2.3. Compréhension du langage naturel (NLU) :

Les chatbots utilisent la compréhension du langage naturel (NLU) pour récupérer le contexte de l'entrée utilisateur non structurée en langage humain et répondre en fonction de l'intention de l'utilisateur actuel. Les trois problèmes majeurs soulevés lors du processus NLU sont les mécanismes de pensée, l'interprétation et les connaissances générales de l'utilisateur.

NLU prend en charge la classification d'intention et l'extraction d'entités, en tenant compte des informations de contexte. Les entités peuvent être définies par le système ou définies par l'utilisateur. Les contextes sont des chaînes qui stockent l'objet auquel l'utilisateur se réfère. Le modèle de classification d'intention peut être un classificateur comme, par exemple, un algorithme SVM linéaire, ou il peut s'agir d'un modèle pré-entraîné qui a été créé par la classification manuelle des messages texte collectés des utilisateurs en sujets [7].

De même, le modèle d'extraction d'entités peut être préformé en annotant manuellement les entités dans les messages texte de l'utilisateur. Pour cette raison, un corpus d'apprentissage annoté peut être créé en associant une étiquette à chaque bloc de mots. Une fois les modèles formés, ils peuvent automatiquement classer les nouveaux messages texte de l'utilisateur dans les intentions et extraire des entités [1].

1.3.2.4. Réseaux de neurones artificiels (ANN):

Pour construire le modèle de réseau neuronal qui sera utilisé pour créer le chatbot, Keras, une bibliothèque Python très populaire pour les réseaux neuronaux sera utilisée. Tout d'abord, nous devons comprendre ce qu'est un Réseau Neural Artificiel (ANN).

Les ANN sont des modèles d'apprentissage machine qui tentent d'imiter le fonctionnement du cerveau humain, dont la structure est construite à partir d'un grand nombre de neurones connectés entre eux, d'où le nom de "réseaux neuronaux artificiels" ; (Artificial Neural Network). Le modèle ANN le plus simple est composé d'un seul neurone et s'appelle Perceptron, il se compose d'un neurone simple, qui prend la somme pondérée de ses entrées, leur applique une fonction mathématique, et produit son résultat [6]

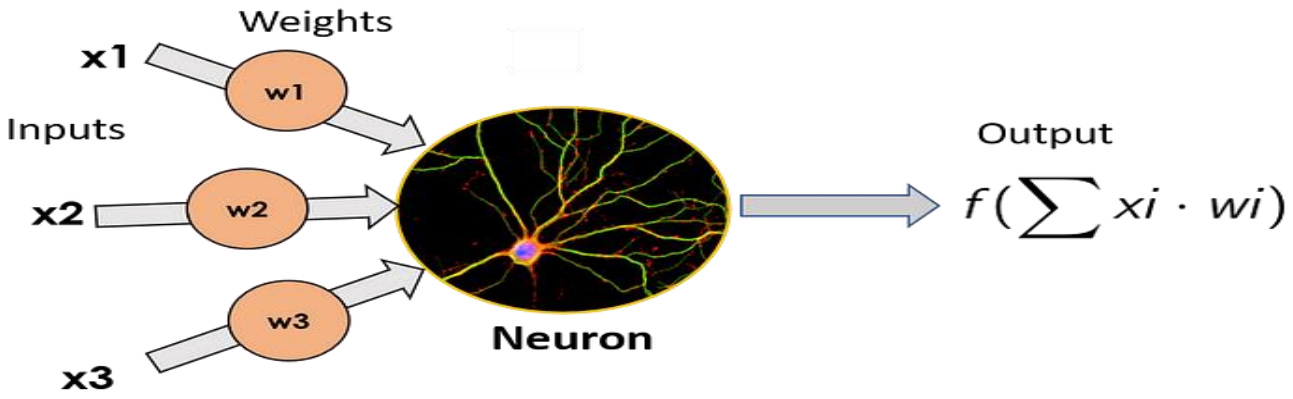


Figure N°10 : Image d'un seul neurone

Un perceptron ou neurone unique est caractérisé par les entrées à gauche, les poids qui multiplient chaque entrée, et le neurone lui-même, qui applique une fonction à la somme

pondérée des entrées et des sorties le résultat. Ces neurones individuels peuvent être empilés les uns sur les autres en formant des couches de la taille que nous voulons, et ensuite ces couches peuvent être placées séquentiellement les unes à côté des autres pour rendre le réseau plus profond. Quand les réseaux sont construits de cette façon, les neurones qui n'appartiennent pas aux couches d'entrée ou de sortie sont considérés comme faisant partie des couches cachées, dépeignant avec leur nom l'une des principales caractéristiques d'un ANN, leur manque d'interprétabilité produit des résultats prodigieux [6]

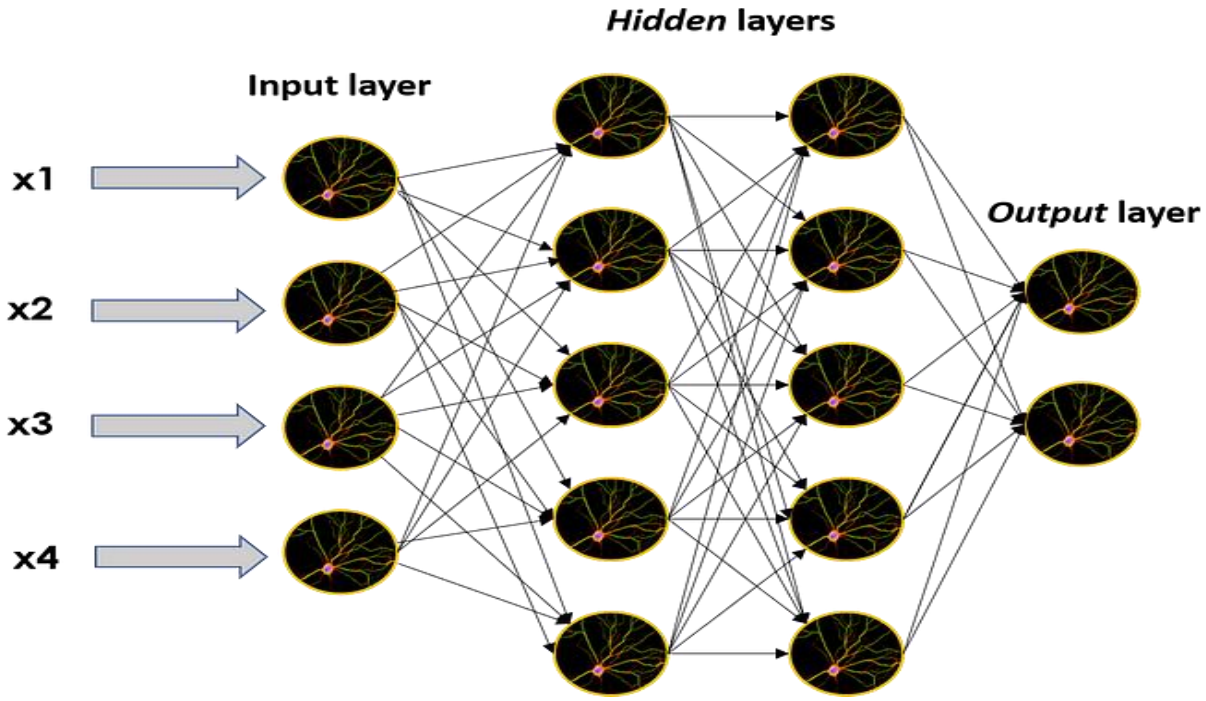


Figure N°11 : Image d'un réseau de neurones plus grand

Un réseau neuronal plus grand: composé de plusieurs neurones et couches individuelles : de couche d'entrée, de couches cachées et de couche de sortie.

Toute fois Les chatbots basés sur la récupération et la génération utilisent plusieurs types de réseaux de neurones artificiels. Le système prend l'entrée de l'utilisateur, calcule son représentations vectorielles, les transmet sous forme de caractéristiques au réseau de neurones et produit la réponse. Le processus de mappage de mots dans des vecteurs est appelé intégration de mots, et plusieurs méthodes peuvent être simples ou utiliser des techniques d'apprentissage en profondeur comme Word2vec.
Dans les systèmes basés sur la récupération, les réseaux de neurones artificiels sont formés à l'aide des vecteurs d'entrée et d'intention de l'utilisateur. Ils prennent en entrée le vecteur d'entrée de l'utilisateur et donnent une probabilité de chaque intention. La classification des entités dans l'entrée utilisateur est effectuée par des systèmes de reconnaissance d'entités nommées (NER), qui peuvent également utiliser des techniques d'apprentissage en profondeur [6].

Bien que les chatbots basés sur la génération soient utiles pour impliquer une personne dans des conversations informelles en domaine ouvert, ils ne sont pas idéaux pour les communications en domaine fermé dans lesquelles les chatbots basés sur des règles conviennent. Les chatbots

hybrides utilisent une approche basée sur la récupération et une approche générative pour répondre aux entrées de l'utilisateur s'il n'y a aucune correspondance avec l'une des règles.

1.3.2.5. Réseaux de neurones récurrents (RNN) :

Les réseaux neuronaux récurrents sont un type spécial de réseaux neuronaux qui sont conçus pour traiter efficacement des données séquentielles. Ce type de données comprend des séries temporelles (une liste de valeurs de certains paramètres sur une certaine période de temps), des documents texte, qui peuvent être vus comme une séquence de mots, ou audio, qui peut être vu comme une séquence de fréquences sonores [8].

Les RNN font cela en prenant la sortie de chaque neurone, et en la lui renvoyant comme entrée. En faisant cela, il ne reçoit pas seulement de nouvelles informations à chaque pas de temps, mais il ajoute également à ces nouvelles informations une version pondérée de la sortie précédente. Ces neurones ont donc une sorte de " mémoire " des entrées précédentes qu'ils ont eues, car elles sont en quelque sorte quantifiées par la sortie qui est renvoyée au neurone. Un neurone récurrent, où les données de sortie sont multipliées par un poids et réinjectées dans l'entrée

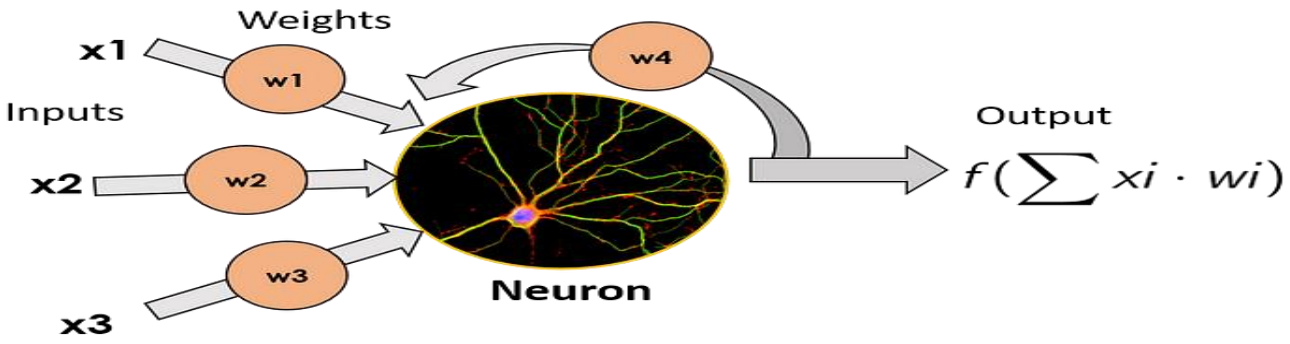


Figure N° 12 : Image d'un neurone récurrent

Les développeurs de chatbots utilisent les réseaux de neurones récurrents (RNN) pour tenir compte du contexte précédent dans une conversation. Dans RNN, la nouvelle entrée de l'utilisateur et les informations des données antérieures alimentent les neurones. De cette manière, une boucle transmet les connaissances d'une partie du réseau à la suivante.

De plus, lorsque les développeurs de chatbots ont besoin de se référer à des informations précédentes et d'apprendre des dépendances à long terme, ils utilisent des réseaux de mémoire à long court terme (LSTM), un type particulier de RNN. Jusqu'où le développeur souhaite remonter dans les informations d'une discussion peut être un paramètre configurable[13].

Un chatbot pour répondre aux questions fréquemment posées (FAQ) a été développé avec l'utilisation des réseaux de neurones récurrents LSTM, et les résultats expérimentaux ont montré que le chatbot pouvait reconnaître les questions et y répondre avec un très haut niveau de précision. De plus, RNN et Gated Recurrent Units (GRU) bidirectionnels ont été utilisés en plus d'un mécanisme d'attention pour générer les réponses appropriées en combinant le contexte de la conversation et les connaissances données [8].

1.3.2.6. Modèle séquence à séquence :

Un modèle typique basé sur la génération est Sequence-to-Sequence (Seq2Seq), qui génère une séquence cible en examinant la séquence source. La séquence source est l'entrée de l'utilisateur et la séquence cible est la réponse du chatbot. Deux RNN peuvent être utilisés comme encodeur et le décodeur, qui est la version la plus basique et originale du modèle, ou LSTM et GRU, peuvent être utilisés pour modéliser des phrases plus longues. Ces réseaux de neurones sont entraînés à l'aide de la rétro-propagation dans le temps, une version modifiée de la rétro-propagation [9].

Des chatbots émotionnellement conscients ont été implémentés en utilisant Seq2Seq avec GRU. Dans une autre recherche, un encodeur-décodeur RNN a été proposé pour générer une séquence cible à partir d'une séquence source en utilisant une unité cachée avec une unité de réinitialisation et de mise à jour pour contrôler les informations dont il se souvient. De plus, un chatbot qui inscrit un ami virtuel a été proposé en utilisant Seq2Seq [1].

1.3.2.7. Modèles Deep Seq2seq :

Les chatbots génératifs peuvent avoir une performance meilleure et plus humaine lorsque le modèle est plus approfondi et a plus de paramètres, comme dans le cas des modèles Seq2seq profonds contenant plusieurs couches de réseaux LSTM.

Dans une autre recherche un modèle d'encodeur-décodeur récurrent hiérarchique à variable latente (VHRED) est introduit, il maintient le contexte à long terme de la conversation et génère des sorties étendues.

Dans une autre recherche, un chatbot à domaine ouvert a été développé avec l'utilisation de réseaux de neurones récurrents bidirectionnels (BRNN) et de couches d'attention pour donner les réponses appropriées lorsque l'entrée de l'utilisateur consiste en plus de 20 mots. Un chatbot multilingue basé sur des modèles Seq2seq profonds avec des cellules GRU a été proposé pour soutenir les personnes dans leurs activités liées au patrimoine culturel [1].

1.3.2.8. L'optimiseur ADAM :

Adam (Adaptive Moment Estimation) est un algorithme d'optimisation couramment utilisé dans le domaine de l'apprentissage automatique, en particulier pour l'entraînement des réseaux de neurones profonds. Il est basé sur la méthode de descente de gradient stochastique (SGD) et combine des éléments de la méthode du moment et de la méthode de RMSProp.

L'optimiseur Adam maintient une estimation adaptative des moments du premier ordre (moyenne du gradient) et du second ordre (moyenne mobile des carrés des gradients). Ces estimations sont mises à jour au fil des itérations de l'entraînement pour ajuster les taux d'apprentissage des différents poids du modèle [10].

Voici les étapes principales de l'algorithme Adam :

1. Initialisation : Les paramètres initiaux, y compris les moments du premier et du second ordre, sont initialisés.

2. Calcul du gradient : Le gradient est calculé en utilisant un échantillon aléatoire du jeu de données.

3. Mise à jour des moments du premier et du second ordre : Les moments du premier ordre et du second ordre sont mis à jour en utilisant les gradients calculés.

4. Biais de correction : Les moments du premier et du second ordre sont corrigés pour compenser les biais introduits au début de l'entraînement.

5. Mise à jour des poids : Les poids du modèle sont mis à jour en utilisant les moments du premier et du second ordre corrigés.

L'utilisation de l'optimiseur Adam permet généralement une convergence plus rapide et une meilleure performance des modèles d'apprentissage automatique. Cependant, il convient de noter que le choix de l'optimiseur dépend du problème spécifique et qu'il existe d'autres alternatives telles que SGD, RMSProp, Adagrad, etc., qui peuvent également être efficaces selon le contexte.

AdamClasse :

Voici l'Optimiseur qui implémente l'algorithme Adam

```
tf.keras.optimizers.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,
    weight_decay=None,
    clipnorm=None,
    clipvalue=None,
    global_clipnorm=None,
    use_ema=False,
    ema_momentum=0.99,
    ema_overwrite_frequency=None,
    jit_compile=True,
    name="Adam",
)
```

Optimiseur qui implémente l'algorithme Adam.

Le Raisonnements de l'optimiseur :

learning_rate : A tf.Tensor, valeur à virgule flottante, un planning qui est un tf.keras.optimizers.schedules.LearningRateSchedule, ou un callable qui ne prend aucun argument et renvoie la valeur réelle à utiliser. Le taux d'apprentissage. La valeur par défaut est 0,001.

beta_1 : une valeur flottante ou un tenseur flottant constant, ou un appelable qui ne prend aucun argument et renvoie la valeur réelle à utiliser. Le taux de décroissance exponentielle pour les estimations du 1er moment. La valeur par défaut est 0,9.
beta_2 : Une valeur flottante ou un tenseur flottant constant, ou un appelable qui ne prend aucun argument et renvoie la valeur réelle à utiliser. Le taux de décroissance exponentielle pour les estimations du 2e moment. La valeur par défaut est 0,999.
epsilon : Une petite constante pour la stabilité numérique. Cet epsilon est "chapeau epsilon" dans l'article Kingma et Ba (dans la formule juste avant la section 2.1), pas l'epsilon dans l'algorithme 1 de l'article. Par défaut, 1e-7.
amsgrad : booléen. S'il faut appliquer la variante AMSGrad de cet algorithme de l'article "Sur la convergence d'Adam et au-delà". La valeur par défaut est False.
nom : Chaîne. Le nom à utiliser pour les poids de l'accumulateur de quantité de mouvement créés par l'optimiseur.
weight_decay : Flottant, par défaut sur Aucun. S'il est défini, la décroissance du poids est appliquée.
clipnorm : Flottant. S'il est défini, le gradient de chaque poids est coupé individuellement afin que sa norme ne soit pas supérieure à cette valeur.

clipvalue : Flottant. S'il est défini, le gradient de chaque poids est coupé pour ne pas être supérieur à cette valeur.
global_clipnorm : Flottant. S'il est défini, le gradient de tous les poids est tronqué afin que leur norme globale ne soit pas supérieure à cette valeur.
use_ema : booléen, par défaut à False. Si Vrai, la moyenne mobile exponentielle (EMA) est appliquée. L'EMA consiste à calculer une moyenne mobile exponentielle des poids du modèle (à mesure que les valeurs de poids changent après chaque lot d'apprentissage) et à remplacer périodiquement les poids par leur moyenne mobile.
ema_momentum : Flottant, par défaut à 0,99. Utilisé uniquement si use_ema=True. Voici la quantité de mouvement à utiliser lors du calcul de l'EMA des poids du modèle : new_average = ema_momentum * old_average + (1 - ema_momentum) * current_variable_value.
ema_overwrite_frequency : Int ou None, par défaut sur None. Utilisé uniquement si use_ema=True. A chaque ema_overwrite_frequencypas d'itérations, nous écrasons la variable du modèle par sa moyenne mobile. Si None, l'optimiseur n'écrase pas les variables de modèle au milieu de la formation, et vous devez écraser explicitement les variables à la fin de la formation en appelant optimizer.finalize_variable_values()(ce qui met à jour les variables de modèle sur place). Lorsque vous utilisez la boucle d'entraînement intégrée fit(), cela se produit automatiquement après la dernière époque et vous n'avez rien à faire.
jit_compile : booléen, par défaut à True. Si True, l'optimiseur utilisera la compilation XLA. Si aucun périphérique GPU n'est trouvé, cet indicateur sera ignoré.

1.4. Étude des chatbots appliqués au domaine des télécommunications :

L'étude des chatbots appliqués au domaine des télécommunications permet de comprendre comment ces technologies peuvent être utilisées pour améliorer l'expérience client, optimiser les processus de support technique et répondre aux besoins spécifiques de ce secteur. Voici quelques aspects clés de cette étude :

1. Automatisation du support client : Les chatbots peuvent être utilisés pour automatiser une partie du support client en fournissant des réponses instantanées aux questions courantes et en aidant les clients à résoudre les problèmes les plus simples. Cela permet de réduire les temps d'attente et d'améliorer la satisfaction client.

2. Réponse aux demandes fréquentes : Les télécommunications sont souvent confrontées à un grand nombre de demandes récurrentes, telles que des demandes de renseignements sur les forfaits, les factures, les services disponibles, etc. Les chatbots peuvent être programmés pour répondre à ces demandes de manière rapide et précise, offrant ainsi une assistance instantanée aux clients.

3. Gestion des dérangements : Les chatbots peuvent jouer un rôle clé dans la gestion des dérangements en permettant aux clients de signaler les problèmes, de suivre l'état de leurs demandes et d'obtenir des mises à jour en temps réel. Les chatbots peuvent également collecter des informations supplémentaires sur les dérangements afin de faciliter la résolution rapide et efficace des problèmes.

4. Promotion des offres et des nouveaux services : Les chatbots peuvent être utilisés pour informer les clients sur les offres spéciales, les promotions, les nouveaux services ou les mises à jour technologiques. Les chatbots peuvent recommander des forfaits ou des services adaptés aux besoins des clients en se basant sur des données collectées et des analyses prédictives.

5. Assistance à l'installation et à la configuration : Les chatbots peuvent guider les clients dans l'installation et la configuration de nouveaux équipements ou services, en fournissant des instructions étape par étape et des réponses personnalisées en fonction des besoins spécifiques des clients.

6. Collecte de commentaires et d'avis : Les chatbots peuvent être utilisés pour recueillir les commentaires des clients, les évaluations et les avis sur les services télécom. Cela permet à l'entreprise d'obtenir des informations précieuses sur la satisfaction client et d'identifier les domaines d'amélioration.

L'étude des chatbots appliqués au domaine des télécommunications met en évidence les avantages potentiels de ces technologies, tels que l'amélioration de l'efficacité opérationnelle, la réduction des coûts de support client et l'amélioration de l'expérience client globale. Cependant, il est également important de prendre en compte les défis spécifiques liés à la langue, aux variations régionales et aux caractéristiques propres à chaque marché.

1.5. Développement du chatbot :

1.5.1. Le processus de création d'un chatbot :

La création d'un chatbot commence par la planification des objectifs, des procédures et des besoins des utilisateurs. Le développement du chatbot à l'aide d'un langage de programmation ou d'une plate-forme de développement de chatbot suit, puis les fonctionnalités du chatbot sont testées localement.

Le chatbot est ensuite publié sur un site web en ligne ou un centre de données, et il est connecté à un ou plusieurs canaux pour envoyer et recevoir des messages. À ce stade, l'intégration correcte d'un chatbot dans une application est cruciale, et il existe trois méthodologies pour ce faire :

- l'intégration à l'aide d'API,
- l'intégration manuelle
- l'intégration tierce.

Enfin, l'évaluation est menée en recueillant des données pendant un temps déterminé après le déploiement afin que les développeurs puissent détecter les erreurs et utiliser les entrées pour améliorer les performances et les capacités du chatbot.

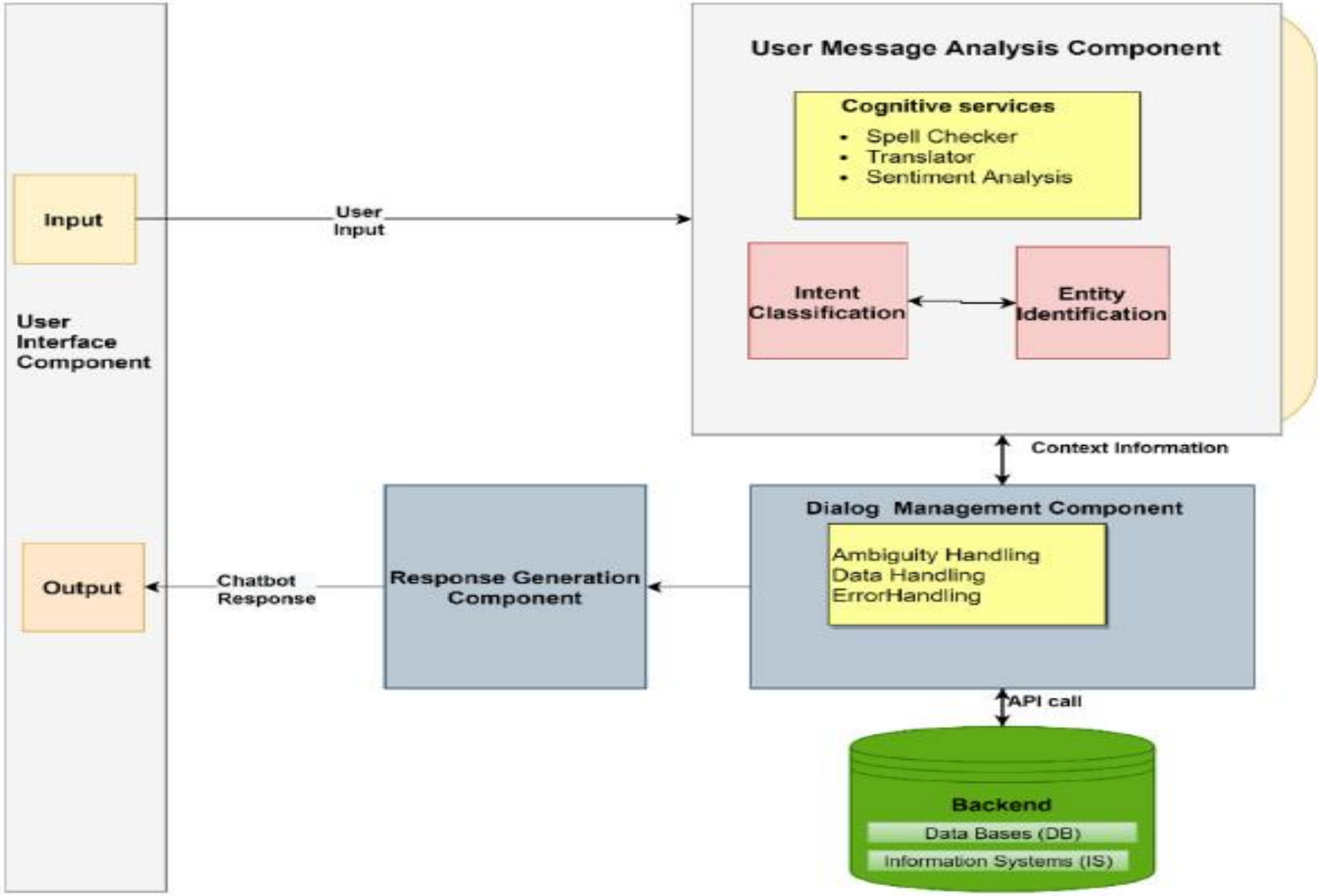


Figure N°13: Architecture Générale de Chatbot[1]

La sélection d'algorithmes ou de plates-formes pour créer des chatbots dépend de ce que le chatbot doit fournir aux utilisateurs et de la catégorie dans laquelle il se situe .Une sélection appropriée peut entraîner les avantages de la connectivité, de l'efficacité, des révisions de production rapides et simples et un effort minimal pour le concepteur. A noter qu'un chatbot est considéré comme plus efficace lorsque l'utilisateur peut se connecter directement sans le télécharger ni l'installer.

Un chatbot peut être développé à l'aide de langages de programmation tels que Java et Python ou d'une plate-forme de développement de chatbot qui peut être commerciale ou open source. Les plates-formes open source incluent: RASA, Botkit (blocs de construction pour créer des bots), Chatterbot, Pandorabots, Botlytics et Microsoft Bot Framework tandis que les plateformes commerciales incluent Botsify, Chatfuel, Manychat et Flow XO.

Certaines plates-formes cloud NLU alimentées par l'apprentissage automatique sont Google DialogFlow, Facebook wit.ai, Microsoft LUIS, IBM Watson Conversation, Amazon et SAP Conversation AI.

Les plates-formes open source mettent leur code à disposition et le développeur peut avoir le contrôle total de la mise en œuvre. Bien que les plateformes commerciales ne donnent pas le contrôle total aux développeurs, les développeurs bénéficient généralement de données efficaces pour former leurs chatbots [1].

De plus, les plateformes commerciales facilitent l'intégration aux autres produits de la plateforme. Par exemple, un chatbot construit avec Dialogflow peut être facilement interfacé avec Google Assistant et d'autres appareils Google. D'autres installations que de nombreuses plates-formes de développement peuvent offrir incluent :

- Ajout facile d'intentions et test facile du déclenchement d'intention.

- Gestion du contexte. La plupart des plateformes de développement prennent en charge la gestion du contexte. Une bonne gestion du contexte est essentielle pour la classification des intentions. Par exemple, si l'énoncé de l'utilisateur est « À Kavala, aujourd'hui ? », il est fort probable que la réponse résultera d'une intention non classifiée. Cependant, si un contexte comme « prévision météo » est activé, une réponse réussie indiquera la météo à Kavala le jour de la conversation.

- Prise en charge des entités prédéfinies et définies par l'utilisateur et détection automatique des valeurs d'entité.

- Prise en charge de la recherche dans des collections de documents de connaissances, comme des FAQ ou des articles, pour trouver des réponses.

Les plates-formes individuelles peuvent également offrir des installations supplémentaires. Par exemple, Google Dialogflow prend en charge les événements, à la fois définis par la plate-forme et personnalisés. Au lieu de l'entrée de l'utilisateur, les événements peuvent également déclencher des intentions. Par exemple, un événement survenu à un moment précis peut déclencher une intention pour alerter l'utilisateur que son temps de pause est terminé.

Enfin, les plateformes de développement peuvent être associées à des langages de programmation si les exigences du projet l'exigent.

Par conséquent, les plateformes de développement peuvent faire partie intégrante de la mise en œuvre d'un chatbot. Bien sûr, avant de choisir une plateforme commerciale, il faut considérer les coûts liés à son utilisation.

1.5.2. Former un chatbot :

Il existe de nombreux corpus disponibles pour entraîner les chatbots. Les approches de réseau de neurones Chitchat sont généralement formées sur des corpus de scripts de films ou des dialogues de plates-formes Web. Un chatbot formé sur de tels ensembles de données, formés par des discussions entre différents interlocuteurs, manque souvent de personnalité spécifique. De plus, divers problèmes peuvent survenir lors de l'utilisation d'un corpus de dialogue avec des exemples de dialogue humain pour recycler un chatbot.

Lorsque le corpus de dialogue n'est pas assez volumineux pour l'apprentissage, les réponses du chatbot peuvent contenir des erreurs syntaxiques ou sémantiques. Par conséquent, des méthodes ont été proposées pour pré-entraîner un chatbot à l'aide d'un grand corpus non dialogué, et pour le recycler à l'aide d'un petit corpus dialogué.

Parfois, des corpus de dialogue et des approches d'apprentissage automatique sont utilisés pour générer les règles AIML et donc former des chatbots basés sur des règles. De plus, diverses versions d'un chatbot dans différentes langues ont été crées automatiquement.

Les développeurs de chatbot conservent généralement les fichiers des conversations après le déploiement du chatbot. Cela les aide à mieux comprendre les demandes des utilisateurs et à améliorer le chatbot, qui apprend en permanence grâce à l'analyse de ces conversations. De nouveaux exemples d'apprentissage sont extraits des conversations auxquelles un chatbot participe lorsque la conversation se déroule bien ou que des commentaires sont demandés.

1.5.3. Connecter le chatbot à un canal :

Au cours des dernières années, les chatbots ont gagné en popularité et sont utilisés dans diverses applications de messagerie ou connectés à des sites Web.

Facebook est un réseau social populaire pour les chatbots, principalement pour effectuer des transactions ou des services plutôt que pour discuter avec les utilisateurs. Les chatbots Facebook Messenger font généralement partie des discussions de groupe et remplissent des fonctions telles que fournir des statistiques sur un match de sport, créer une liste de lecture musicale ou donner des réponses intelligentes comme informer sur les heures d'ouverture ou faire une réservation sans quitter la fenêtre de discussion. Par conséquent, les chatbots Facebook assistent principalement les utilisateurs de manière administrative plutôt que de communiquer avec eux.

Les chatbots Skype, de la même manière que ceux de Facebook, sont couramment utilisés dans les discussions de groupe à des fins fonctionnelles. Les chatbots Skype deviennent plus interactifs lorsqu'ils offrent aux utilisateurs l'alternative du chat vocal au lieu de taper.

Twitter adopte une nouvelle approche pour ses chatbots, offrant une plate-forme permettant aux entreprises de se connecter avec leurs clients et offrant une interaction agréable plutôt que transactionnelle.

Les entreprises utilisent les chatbots Slack en interne pour maximiser l'efficacité, améliorer la connectivité ou effectuer des tâches. Il existe deux catégories de chatbots Slack, ceux qui envoient des notifications et ceux qui effectuent des transactions spécifiques initiées par l'utilisateur.

Les avantages de l'encapsulation du chatbot dans les applications de messagerie sont inclus : l'interaction avec le chatbot peut être rapidement répartie sur le réseau social de l'utilisateur sans quitter l'application de messagerie, qui garantit l'identité de l'utilisateur. Les chatbots peuvent être insérés dans des discussions de groupe ou échangés comme n'importe quel contact, tandis qu'un système de notification réengage les utilisateurs inactifs. De plus, les systèmes de paiement sont intégrés à l'application de messagerie et peuvent être utilisés de manière sûre et efficace [10] .

Contrairement aux canaux mentionnés ci-dessus, les chatbots basés sur un site Web offrent aux développeurs un contrôle total. Sur le site Web, on peut dire précisément comment fonctionne le chatbot, y compris son objectif, son interface utilisateur et son expérience. De plus, les utilisateurs peuvent engager une conversation sans quitter la page en cours, en ayant un moyen simple de poser des questions.

1.5.4. Mode conversationnel des chatbots :

Lorsqu'un chatbot prend le personnage d'une personne célèbre, l'interaction avec son utilisateur semble être améliorée. Freudbot, un chatbot développé avec AIML, a été utilisé dans l'enseignement à distance et en ligne, et il s'est avéré être un outil d'enseignement et d'apprentissage utile. Il a été programmé selon les règles conversationnelles liées au tour de parole, fournissant des réponses avec des implications qui invitaient l'utilisateur à demander plus d'informations rendant la conversation plus longue .

Les chatbots devraient également offrir aux utilisateurs une expérience plus accessible et réaliste pour demander leurs paramètres de confidentialité dans les applications ou les sites Web.

De plus, les chatbots peuvent susciter l'intérêt et impliquer les utilisateurs dans des activités telles que remplir des questionnaires. Répondre aux questions du chatbot serait une solution amusante et souhaitable pour quelqu'un car cela ne prend pas autant de temps que les questionnaires.

Les utilisateurs et les chatbots interagissent les uns avec les autres, et il est intéressant d'examiner «l'agence symbiotique», un terme initialement utilisé pour l'agence proxy où les utilisateurs et les logiciels agissent dans l'interaction homme-technologie. Actuellement, le terme élargit ce concept de système de proxy pour considérer à la fois comment la technologie médiatise les pensées, les croyances et les attitudes de quelqu'un, et comment l'action humaine affecte l'utilisation des produits technologiques.

1.5.5. Autres considérations de mise en œuvre :

Lors de la conception d'un chatbot, il faut d'abord déterminer les objectifs qu'il servira. En fonction des objectifs, nous évaluerons le caractère principal du chatbot, par exemple, si nous avons besoin d'un chatbot générique, interdomaine ou domaine fermé. Dans les deux premiers cas, nous aurons probablement besoin d'utiliser des techniques de PNL. Avant de décider, nous devons prendre en compte si les données nécessaires à la formation du chatbot sont disponibles. A l'inverse, pour les chatbots de domaine fermé, l'utilisation d'un langage de script peut être préférable. En général, il n'est pas facile de trouver des données de formation appropriées à des fins spécifiques.

D'autre part, un chatbot à domaine fermé basé sur un langage de script peut, s'il est correctement conçu, guider efficacement l'utilisateur vers la réalisation d'objectifs spécifiques. Par exemple, un assistant d'apprentissage du vocabulaire peut facilement rediriger la conversation lorsqu'il ne parvient pas à classer l'intention de l'utilisateur en disant : "Je n'ai pas bien compris". Voulez-vous que je vous aide à étudier un peu de vocabulaire ? » Tant que la discussion reste dans le cadre de l'apprentissage du vocabulaire, les dialogues sont limités et peuvent être précisément conçus pour que le résultat soit satisfaisant. De plus, un tel chatbot peut être une première approche qui aidera à collecter des données à partir des phrases des utilisateurs et enrichira l'expérience de son fabricant afin que s'il soit jugé utile de passer à la deuxième phase en utilisant la technologie NLP.

Quelle que soit l'approche que nous suivons, nous devons anticiper que notre logiciel mesurera l'expérience utilisateur afin que, sur la base des mesures, nous puissions apporter les améliorations appropriées. Cependant, si nous choisissons la technologie NLP, nous devons considérer que le choix d'une plate-forme de mise en œuvre qui prend en charge l'abstraction cognitive nous fournira, dans une certaine mesure, des mises à jour automatiques sur les aspects en constante évolution de l'intelligence artificielle.

De plus, c'est une excellente pratique de fournir la possibilité de gérer les impasses avec les membres de notre personnel afin d'éviter des expériences utilisateur désagréables.

Une décision qui est également importante concerne les langues que nous devons prendre en charge. Une attention particulière est requise dans la communication vocale car tous les systèmes de synthèse vocale (TTS) et de reconnaissance automatique de la parole (ASR) ne prennent pas en charge toutes les langues.

Chapitre II :

Conception du chatbot de relève des dérangements pour Algérie Télécom

2.1. Présentation de l'entreprise et de son service client :

Algérie Télécom est l'opérateur historique des télécommunications en Algérie. Fondée en 2003, l'entreprise est une société d'État chargée de fournir des services de téléphonie fixe, de téléphonie mobile, d'accès à Internet et d'autres services de communication à travers le pays.

En tant qu'opérateur national, Algérie Télécom joue un rôle essentiel dans le développement des infrastructures de télécommunications en Algérie. L'entreprise est responsable de la gestion et de l'exploitation du réseau de télécommunications du pays, y compris les infrastructures de fibre optique, les stations de base pour les services mobiles et les équipements de commutation.

Algérie Télécom propose une large gamme de services aux particuliers, aux entreprises et aux administrations publiques. Cela inclut des services de téléphonie fixe traditionnelle, des services de téléphonie mobile à travers sa filiale Mobilis, des services d'accès à Internet haut débit, des services de données, des solutions de réseau privé virtuel (VPN) et d'autres services de communication.

L'entreprise s'engage également dans la modernisation de ses infrastructures et l'amélioration de la qualité des services offerts. Elle investit dans le déploiement de la fibre optique à haut débit pour fournir une connectivité plus rapide et plus fiable à ses clients. Algérie Télécom vise à répondre aux besoins croissants en matière de connectivité et à contribuer au développement de l'économie numérique en Algérie.

En ce qui concerne la gestion des dérangements, Algérie Télécom met en place des procédures pour recevoir, diagnostiquer et résoudre les problèmes de ses clients. Cela inclut la gestion des réclamations, les interventions techniques sur le réseau et les services de support client. La mise en place d'un chatbot de relève des dérangements pourrait être une solution innovante pour améliorer l'efficacité et la réactivité du service client en permettant aux clients de signaler et de suivre les dérangements de manière plus rapide et automatisée.

Algérie Télécom est l'opérateur national des télécommunications en Algérie, offrant une gamme complète de services de télécommunications. L'entreprise cherche continuellement à moderniser ses infrastructures et à améliorer la qualité des services offerts, tout en s'adaptant aux besoins changeants de ses clients. La mise en place d'un chatbot de relève des dérangements pourrait être une étape importante dans cette direction, en offrant une expérience client améliorée et une gestion plus efficace des dérangements.

2.2. Analyse des problèmes de dérangements rencontrés par les clients :

L'analyse des problèmes de dérangements rencontrés par les clients d'Algérie Télécom est essentielle pour comprendre les défis auxquels l'entreprise est confrontée et identifier les domaines dans lesquels un chatbot de relève des dérangements pourrait apporter des améliorations. Voici quelques problèmes courants auxquels les clients peuvent être confrontés :

1. Interruptions de service : Les clients peuvent rencontrer des problèmes tels que des interruptions de connexion Internet, des coupures de ligne téléphonique ou des pannes de services spécifiques. Ces interruptions peuvent entraîner une interruption des activités des clients et affecter leur productivité.

2. Problèmes de vitesse ou de qualité de connexion : Les clients peuvent éprouver des difficultés avec la vitesse de leur connexion Internet, une qualité audio médiocre lors des appels téléphoniques ou des problèmes de latence. Cela peut entraîner une expérience utilisateur insatisfaisante et un impact négatif sur les activités en ligne.

3. Défaillances matérielles : Les clients peuvent rencontrer des problèmes liés au matériel fourni par Algérie Télécom ou par d'autres fournisseurs, tels que des routeurs défectueux, des modems défaillants ou des équipements endommagés. Ces problèmes peuvent nécessiter des interventions techniques pour résoudre les dérangements.

4. Facturation incorrecte : Les clients peuvent rencontrer des problèmes de facturation incorrecte, tels que des frais excessifs, des services non facturés ou des erreurs dans les détails de la facture. Cela peut entraîner des litiges et une insatisfaction client.

5. Service client inefficace : Les clients peuvent faire face à des difficultés pour contacter le service client d'Algérie Télécom, des temps d'attente longs, des réponses non satisfaisantes ou un manque de suivi des problèmes signalés. Cela peut entraîner une frustration et une insatisfaction client.

Il est important de mener une analyse approfondie de ces problèmes, en recueillant des données sur leur fréquence, leur impact sur les clients et les tendances observées. Cela permettra d'identifier les problèmes les plus critiques et de déterminer comment un chatbot de relève des dérangements peut contribuer à les résoudre de manière efficace et rapide.

2.3. Identification des besoins et avantages d'un chatbot de relève des dérangements :

L'identification des besoins et des avantages d'un chatbot de relève des dérangements pour Algérie Télécom peut être réalisée en considérant les différents acteurs impliqués, tels que les clients et l'entreprise elle-même. Voici quelques éléments à prendre en compte :

Besoins des clients :

1. Facilité et rapidité : Les clients souhaitent pouvoir signaler un dérangement de manière simple et rapide, sans avoir à passer par des processus complexes ou à attendre longtemps au téléphone.

2. Disponibilité 24/7 : Les dérangements peuvent survenir à tout moment, et les clients apprécient la possibilité de signaler un problème et d'obtenir des informations à toute heure, même en dehors des heures d'ouverture des services client.

3. Automatisation des processus : Les clients préfèrent souvent des solutions automatisées qui leur permettent de signaler et de suivre les dérangements sans avoir à interagir directement avec un agent.

Avantages pour les clients :

1. Rapidité de résolution : Un chatbot de relève des dérangements peut permettre une réponse instantanée aux clients, en fournissant des informations sur la nature du problème et en proposant des solutions préliminaires.

2. Suivi en temps réel : Les clients peuvent suivre l'état d'avancement de leur demande de dérangement grâce aux notifications et aux mises à jour fournies par le chatbot, ce qui réduit l'incertitude et améliore la satisfaction client.

3. Disponibilité continue : Un chatbot fonctionnant 24/7 permet aux clients de signaler un dérangement à tout moment, sans dépendre des heures de travail des agents.

Besoins de l'entreprise :

1. Gestion efficace des dérangements : Un chatbot de relève des dérangements peut aider à automatiser le processus de signalement et de suivi des dérangements, ce qui permet à l'entreprise de gérer efficacement un grand volume de demandes.

2. Réduction de la charge de travail des agents : En automatisant certaines tâches, le chatbot peut alléger la charge de travail des agents du service client, leur permettant de se concentrer sur des problèmes plus complexes et de fournir un support plus personnalisé.

3. Collecte de données et analyses : Un chatbot peut collecter des données sur les dérangements signalés, ce qui permet à l'entreprise de réaliser des analyses pour identifier les tendances, les problèmes récurrents et prendre des mesures proactives pour améliorer la qualité des services.

Avantages pour l'entreprise :

1. Amélioration de l'efficacité opérationnelle : L'automatisation du processus de relève des dérangements réduit les délais de traitement et permet à l'entreprise de gérer un plus grand volume de demandes avec moins de ressources.

2. Réduction des coûts : En réduisant la dépendance à l'égard des interactions humaines, un chatbot peut contribuer à réduire les coûts liés au support client, notamment en diminuant le besoin de recrutement et de formation d'un grand nombre d'agents.

3. Amélioration de la satisfaction client : En offrant une solution rapide et facile pour signaler les dérangements, un chatbot peut améliorer l'expérience client globale et renforcer la satisfaction des clients à l'égard des services d'Algérie Télécom.

L'implémentation d'un chatbot de relève des dérangements pour Algérie Télécom répondrait aux besoins des clients en termes de rapidité, de disponibilité et d'automatisation des processus. Il offrirait des avantages tels que la rapidité de résolution, le suivi en temps réel et la disponibilité continue. Pour l'entreprise, cela permettrait une gestion plus efficace des dérangements, une réduction de la charge de travail des agents, une collecte de données et des analyses pertinentes, ainsi qu'une amélioration de l'efficacité opérationnelle et de la satisfaction client.

2.4. Conception du chatbot de relève des dérangements :

2.4.1. Analyse des exigences fonctionnelles et non fonctionnelles :

Dans cette section, l'analyse des exigences fonctionnelles et non fonctionnelles du chatbot de relève des dérangements sera effectuée. Les exigences fonctionnelles décrivent les fonctionnalités spécifiques que le chatbot doit fournir, tandis que les exigences non fonctionnelles concernent les aspects de performance, de sécurité, d'utilisabilité, etc.

Exemples d'exigences fonctionnelles :

- Le chatbot doit être capable de comprendre et d'interpréter les requêtes des utilisateurs concernant les dérangements.

- Le chatbot doit être en mesure de fournir des réponses précises et appropriées aux utilisateurs.

- Le chatbot doit pouvoir collecter les informations nécessaires sur les dérangements, telles que l'emplacement, la nature du problème, etc.

- Le chatbot doit être capable de guider les utilisateurs à travers les étapes de résolution des dérangements, en fournissant des instructions claires.

Exemples d'exigences non fonctionnelles :

- Le chatbot doit avoir une interface utilisateur conviviale et intuitive.

- Le chatbot doit être disponible 24 heures sur 24, 7 jours sur 7.

- Le chatbot doit offrir une réponse rapide et en temps réel aux utilisateurs.

- Le chatbot doit être capable de gérer plusieurs requêtes simultanément.

- Le chatbot doit garantir la confidentialité et la sécurité des informations échangées avec les utilisateurs.

2.4.2. Architecture et choix technologiques :

L'architecture du chatbot peut inclure des composants tels que le moteur de traitement du langage naturel (NLP), la gestion des dialogues, l'intégration avec les systèmes existants,[....] Par conséquent, plusieurs conceptions architecturales ont été proposées. Certains auteurs proposent une architecture, mais elle est spécifique aux chatbots basés sur des règles. D'autres présentent une architecture spécifique aux chatbots basés sur Retrieval et ils ne comportent pas de dessin. La Base de connaissance du chatbot est connectée à d'autres bases de données et systèmes d'information qui donnent des réponses aux requêtes de l'utilisateur. Cependant, il y a une absence de services vitaux comme l'analyse des sentiments et la gestion des ambiguïtés.

Les choix technologiques dépendront des besoins spécifiques d'Algérie Télécom, des compétences disponibles, des ressources et des contraintes techniques. Certains choix technologiques courants pour la conception d'un chatbot comprennent l'utilisation de frameworks de développement de chatbots tels que Dialogflow, Rasa ou Microsoft Bot Framework, l'intégration de services de traitement du langage naturel tels que Google Cloud Natural Language Processing ou IBM Watson NLP, et l'utilisation de langages de programmation tels que Python, JavaScript, ou Java.

Dans notre chatbot Atbot, nous avons adopté les architectures et les choix technologiques suivants :

1. Le réseau de neurones pour la classification d'intentions. Le modèle est chargé à partir du fichier `Atbot_model.h5`.b Le modèle est construit à l'aide de la bibliothèque Keras, qui est une bibliothèque populaire pour l'apprentissage profond.

2. La bibliothèque NLTK (Natural Language Toolkit) pour le prétraitement qui inclut la tokenisation du texte en mots individuels et la lemmatisation des mots pour réduire les mots à leur forme de base.

3. Le fichier JSON nommé `intents.json` qui est utilisé pour stocker et charger les données d'intention dans le chatbot et pour mettre à jour les motifs et les réponses lors de l'interaction avec l'utilisateur.

4. L'interface graphique du chatbot est construite à l'aide de la bibliothèque Tkinter, qui est une bibliothèque standard de Python pour créer des interfaces graphiques. La fenêtre de chat est créée avec des composants tels que la zone de texte pour l'affichage des messages, la zone de texte pour la saisie de l'utilisateur et le bouton pour envoyer les messages.

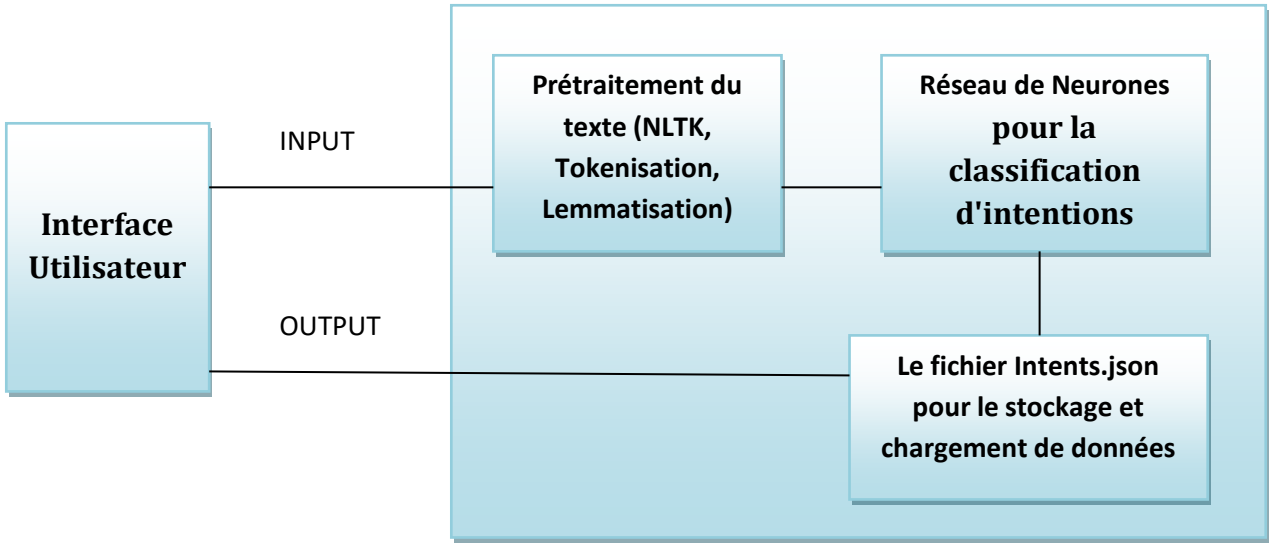


Figure N°14: Architecture et choix technologiques Atbot

Ces choix technologiques et architectures permettent à notre chatbot de comprendre les intentions de l'utilisateur, de générer des réponses en fonction de ces intentions et d'interagir avec l'utilisateur via une interface graphique conviviale [6]

2.4.3. Conception des dialogues et des scénarios de conversation :

Dans cette étape, la conception des dialogues et des scénarios de conversation entre le chatbot et les utilisateurs sera réalisée. Cela implique de définir les différentes étapes de la conversation, les questions à poser aux utilisateurs, les réponses à fournir en fonction des informations fournies, etc. La conception des dialogues doit être intuitive, claire et orientée vers l'objectif de résolution des dérangements.

Le composant de gestion de dialogue contrôle et met à jour le contexte de la conversation. Il conserve l'intention actuelle et les entités identifiées jusqu'à ce point de la conversation. Si le chatbot n'est pas en mesure de collecter les informations de contexte nécessaires, il demande des informations de contexte supplémentaires à l'utilisateur pour répondre aux entités manquantes. Il pose également des questions de suivi une fois l'intention reconnue.

Le composant de gestion de dialogue comprend généralement les modules suivants:

- Gestion des ambiguïtés : Ce module donne des réponses lorsque le chatbot ne peut pas trouver l'intention de la demande de l'utilisateur ou si aucune entrée n'est reconnue. Le chatbot peut indiquer qu'il n'a pas eu de réponse, demander des éclaircissements, démarrer une nouvelle discussion ou donner une réponse générale qui couvre une variété de questions afin que l'utilisateur soit satisfait même s'il a demandé la question la plus imprévisible.

- Le traitement des données : Les informations de l'utilisateur sont stockées dans un fichier. De cette façon, le chatbot peut modifier ses réponses en fonction de l'utilisateur donnant l'impression d'être plus intelligent.

- La gestion des erreurs : Le module de gestion des erreurs fait face aux erreurs inattendues pour assurer le bon fonctionnement du chatbot

Voici quelques modèles de dialogues entre Atbot et utilisateur :

Figure N°15 : Modèle question/réponse utilisateur avec Atbot

2.4.3.1. Composant de génération de réponse :

Le composant de génération de réponses produit des réponses à l'aide d'un ou plusieurs des trois modèles disponibles : modèles basés sur des règles, basés sur la récupération et basés sur la génération.

- Le modèle basé sur des règles sélectionne la réponse à partir d'un ensemble de règles sans générer de nouvelles réponses textuelles. Le composant de gestion de dialogue transmet les valeurs d'espace réservé qui peuvent être nécessaires pour remplir le modèle de réponse au module de génération de réponse. Les modèles basés sur des règles utilisent une base de connaissances (KB) organisée avec des modèles conversationnels.

- Le modèle basé sur la récupération est plus flexible car il sélectionne la réponse la plus appropriée avec la vérification et l'analyse des ressources disponibles à l'aide d'API.

Le modèle génératif utilise la génération de langage naturel (NLG) pour répondre dans un langage naturel de type humain basé sur les dernières entrées et les précédentes. Ce modèle nécessite un ensemble complet de données pour la formation afin d'établir une conversation fructueuse. Lorsque le corpus d'entraînement est petit, des erreurs grammaticales sont commises, en particulier dans les phrases longues.

Dans notre cas pratique, le composant de génération de réponse repose sur les éléments suivants :

1. Prédiction de l'intention : Le texte saisi par l'utilisateur est envoyé à la fonction `predict_class()` pour prédire l'intention de l'utilisateur. Cette fonction utilise le modèle entraîné chargé à partir du fichier `Atbot_model.h5` pour prédire l'intention en se basant sur le sac de mots (bag of words) du texte d'entrée.

2. Récupération de la réponse : Une fois que l'intention de l'utilisateur est prédite, la fonction `getResponse()` est appelée pour récupérer la réponse appropriée. La fonction `getResponse()` parcourt les intentions définies dans le fichier `intents.json` et recherche l'intention correspondant à celle prédite. Une fois que l'intention est trouvée, une réponse est sélectionnée au hasard parmi les réponses associées à cette intention.

3. Affichage de la réponse : La réponse est affichée dans la fenêtre de chat à l'aide de la méthode `ChatBox.insert()` pour ajouter le texte de la réponse à la zone de texte.

Ces composants fonctionnent ensemble pour prédire l'intention de l'utilisateur et générer une réponse appropriée en fonction de cette intention. La logique de récupération des réponses à partir du fichier `intents.json` permet de personnaliser les réponses en fonction des intentions spécifiées dans le fichier.

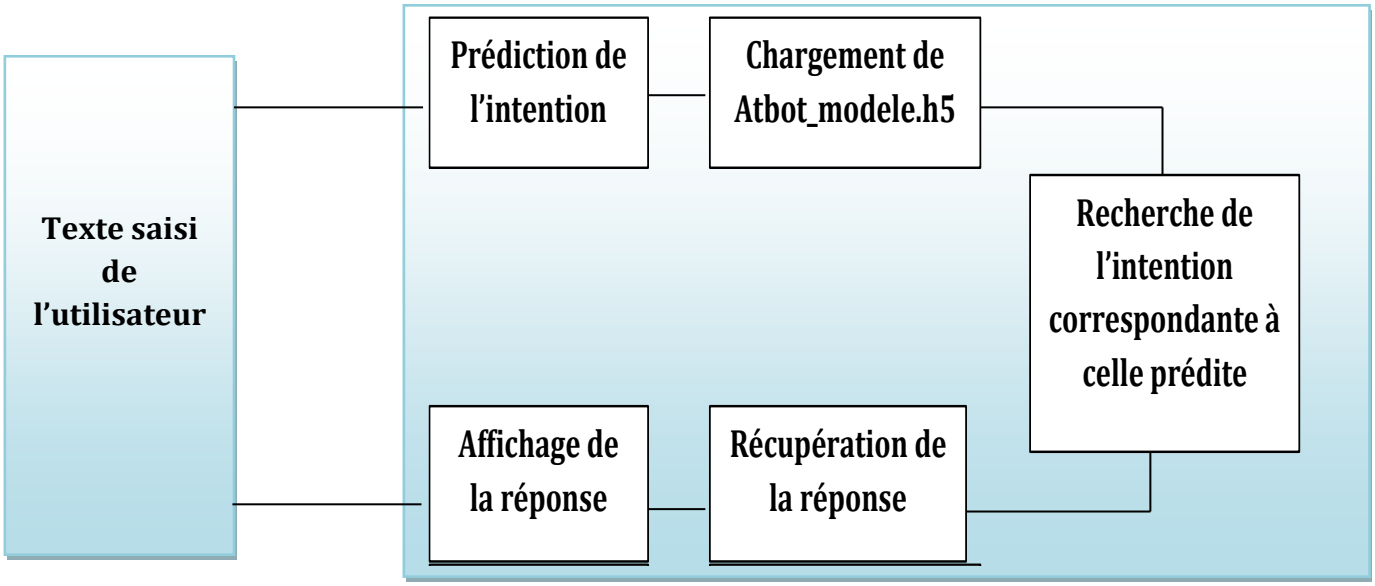


Figure N°16 : Modèle de composants et génération de réponse

2.4.3.2. Backend :

Le chatbot récupère les informations nécessaires pour répondre à l'intention de l'utilisateur à partir du backend via des appels d'API externes ou des requêtes de base de données. Une fois les informations appropriées extraites, elles sont transmises au module de gestion des dialogues, puis au module de génération de réponses. Lorsque des chatbots basés sur des règles sont utilisés, il existe une base de connaissances (KB). Il comprend une liste de réponses manuscrites qui correspondent aux entrées de l'utilisateur. Pour qu'un chatbot reste ferme, la base de connaissances doit couvrir une grande variété de requêtes d'utilisateurs et contenir une variété de réponses à la même entrée d'utilisateur pour éviter la redondance des réponses.

Une base de données relationnelle (RDB) peut être utilisée pour que le chatbot puisse se souvenir des conversations passées, rendant ainsi la communication plus cohérente et pertinente. Cette approche apporte cohérence et précision au dialogue car elle permet au chatbot d'accéder à l'historique des informations précédentes.

La création de la base de connaissances d'un chatbot est une tâche nécessaire mais souvent exigeante et chronophage car elle est développée manuellement. Une méthode a été proposée qui construit automatiquement la base de connaissances d'un nouveau chatbot à partir d'un chatbot existant. De plus, un programme transforme un corpus en base de connaissances AIML d'un chatbot. Souvent, les chatbots basés sur des règles complètent leur base de connaissances en posant des questions à l'utilisateur et en l'encourageant pour de longues conversations.

La base de connaissances peut également prendre en charge des ontologies (réseaux sémantiques) comme Wordnet ou OpenCyc. Le chatbot met à jour l'état de la conversation et recherche les nœuds du Knowledge Graph pour établir des connexions pour les concepts utilisés dans la conversation. Aussi, la base de connaissances AIML peut être enrichie à partir d'une base de connaissances big data avec la connexion du chatbot à l'environnement big data.

L'utilisation d'astuces linguistiques dans la base de connaissances d'un chatbot le rend plus humain. Ces astuces simulent le comportement des gens dans une conversation, comme les réponses stéréotypées, les erreurs de frappe, la manière de taper, l'existence d'une personnalité et même les réponses irrationnelles.

2.5. Intégration des outils de traitement automatique du langage naturel (NLP) :

L'intégration des outils de traitement automatique du langage naturel est essentielle pour permettre au chatbot de comprendre et d'interpréter les requêtes des utilisateurs de manière précise. Cela peut inclure l'utilisation de modèles de langage, de modèles de classification de texte, de techniques de reconnaissance d'entités nommées, etc. L'intégration de ces outils permettra au chatbot de traiter efficacement le langage naturel et d'offrir des réponses cohérentes et pertinentes.

Le contrôleur d'interface utilisateur dirige la demande de l'utilisateur vers le composant d'analyse des messages utilisateur pour trouver l'intention de l'utilisateur et extrait les entités en suivant des approches de correspondance de modèles ou d'apprentissage automatique. Le message de l'utilisateur peut être conservé sous forme de texte brut, ce qui conserve toutes les structures grammaticales et syntaxiques de l'entrée inchangées ou traitées par le traitement du langage naturel (NLP).

Plus précisément, à travers leur contribution au chatbot, les utilisateurs expriment leur objectif, qui est l'intention. Le chatbot doit comprendre l'intention de l'utilisateur et effectuer les actions requises. Différentes entrées utilisateur déclenchent différentes intentions et peuvent inclure des paramètres, appelés entités, pour déterminer des détails précis à leur sujet.

Certains services cognitifs peuvent être liés au composant d'analyse des messages utilisateur pour améliorer la précision :

• Un correcteur orthographique corrige les fautes d'orthographe de l'utilisateur.

• Un traducteur automatique est utilisé dans le cas d'utilisateurs de chatbot multilingues. La langue de l'utilisateur est identifiée et traduite dans la langue de la NLU du chatbot.

• L'analyse des sentiments est appliquée à l'entrée de l'utilisateur pour voir à quel point l'utilisateur semble satisfait ou irrité. L'analyse des sentiments détecte une opinion positive ou négative dans un texte.

La conception du chatbot de relève des dérangements nécessite une analyse approfondie des exigences, des choix technologiques appropriés et une conception soignée des dialogues et des scénarios de conversation. Ces éléments jetteront les bases d'un chatbot performant et efficace pour Algérie Télécom.

L'intégration des outils de NLP dans notre chatbot se fait principalement par des étapes clés du traitement du texte, de la prédiction de l'intention et de la récupération de la réponse :

1. Nettoyage et prétraitement du texte : La fonction `clean_up_sentence()` est utilisée pour nettoyer et prétraiter le texte de l'utilisateur avant de le passer aux étapes de prédiction et de récupération de la réponse. Cette fonction utilise la bibliothèque NLTK (Natural Language Toolkit) pour effectuer la tokenisation des mots et la lemmatisation, ce qui réduit les mots à leur forme de base.

2. Sac de mots (Bag of Words) : La fonction `bag_of_words()` est utilisée pour représenter le texte d'entrée sous forme de sac de mots. Cette fonction utilise les mots prétraités et le vocabulaire stockés dans les fichiers `words.pkl` pour créer un vecteur binaire qui indique la présence ou l'absence de chaque mot dans le texte d'entrée.

3. Prédiction de l'intention : La fonction `predict_class()` utilise le modèle de réseau neuronal chargé à partir du fichier `Atbot_model.h5` pour prédire l'intention de l'utilisateur en se basant sur le sac de mots du texte d'entrée.

4. Récupération de la réponse : La fonction `getResponse()` recherche l'intention prédite dans le fichier `intents.json` et sélectionne une réponse appropriée en fonction de cette intention. Cette récupération de la réponse peut être considérée comme une forme simple de NLP où les intentions sont associées à des réponses prédéfinies.

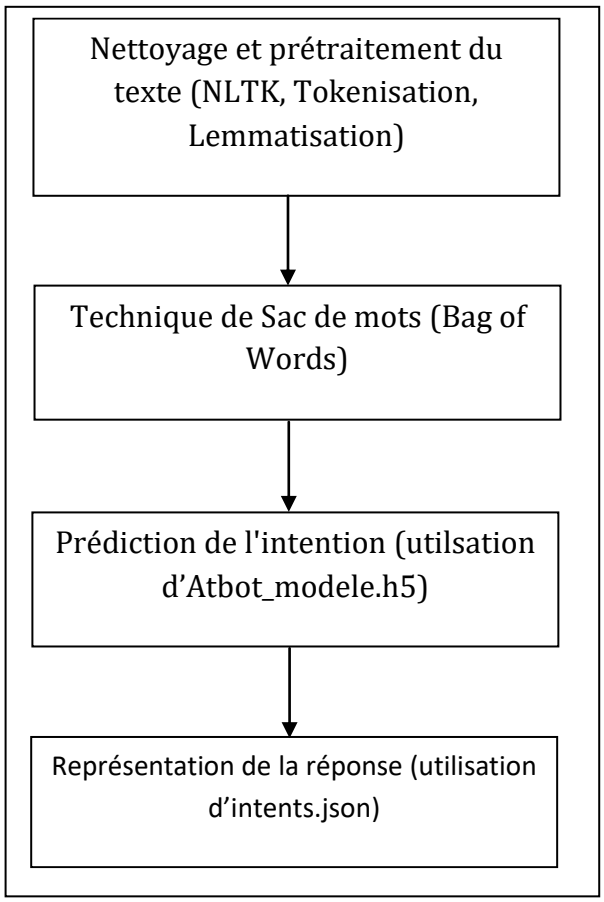


Figure N°17 : les Outils NLP utilisés dans Atbot

Chapitre III : Implémentation du chabot

L'implémentation du chatbot de relève des dérangements sera réalisée. Cela comprend la collecte et la préparation des données, le développement des fonctionnalités et des interfaces utilisateur, l'intégration des systèmes de gestion des dérangements d'Algérie Télécom, ainsi que les tests et l'évaluation des performances du chatbot.

3.1. Collecte et préparation des données :

La collecte et la préparation des données sont cruciales pour entraîner le chatbot à comprendre et à répondre aux requêtes des utilisateurs de manière précise. Cela implique la collecte de données de dérangements réels rencontrés par les clients d'Algérie Télécom, ainsi que la préparation de ces données en les organisant, les nettoyant et les annotant si nécessaire. Les données peuvent inclure des exemples de requêtes des utilisateurs, des réponses associées, ainsi que des métadonnées pertinentes telles que l'emplacement, la nature du dérangement, etc.

3.1.1. Ontologie Algérie télécom :

L'Objectif Consiste à la construction et la formalisation d'une ontologie qui définit un modèle de connaissances et références de vocabulaires au niveau d'une entreprise des Télécoms. Dans la conception de l'ontologie nous avons besoins de toutes les informations sur le domaine des télécommunications en se basant sur la téléphonie fixe, internet et 4G LTE. Notre ontologie de domaine décrit les concepts d'un environnement commercial dans la téléphonie fixe: les acteurs (Client, Abonné, et Produit), les produits (ligne fixe, FTTH, internet ADSL, 4G Lte…etc.). Ensuite on va faire une structuration de ces informations par la conception d'une ontologie.

Spécification des besoins :

Une ontologie ne peut être construite qu'après la phase de spécification, il s'agit d'établir un document informel de spécification des besoins, au niveau de ce document, nous décrivons l'ontologie à construire à travers des aspects représentés comme suit :

a- **Domaine:** service télécommunication.

b- **Objectif** : créer une ontologie pour le domaine d'offre de service en téléphonie et internet qui essaye de répondre à la majorité des besoins de cet environnement comme la recherche, la saisie, la vente, la consultation, suivi des dérangements et de construire un système qui interroge cette ontologie et répond aux exigences des utilisateurs (le client et l'opérateur).

c- **Utilisateurs** : les clients, les opérateurs, les administrateurs

d- **Les termes importants** : client, abonné, ligne téléphonique, internet...etc.

e- **La conceptualisation :**

Une fois que la majorité des connaissances est acquise, on doit les organiser et les structurer en utilisant des représentations semi-formelles qui sont faciles. Cette phase contient plusieurs étapes qui sont :

- Le glossaire de termes.
- Construction des hiérarchies de concepts
- Construction du diagramme de relations binaires et de classification de concepts.
- Dictionnaire de concepts.
- Table des relations binaires.
- Table des attributs.

f- Construire un glossaire des termes :

Construire un glossaire de termes est la première tâche effectué dans l'étape de conceptualisation, ce glossaire contient les noms et les descriptions de tous les termes relatifs au domaine qui seront représentés dans l'ontologie final.

g- Le glossaire des termes de l'ontologie de domaine

concept	Définition
personne	c'est l'acteur qui interagir avec la plate-forme. Il peut être un client, un opérateur, ou un administrateur.
Administrateur	c'est la personne qui gère la plate-forme.
Opérateur	c'est la personne responsable de la tâche d'enregistrement, de vente d'un produit
Client	c'est la personne qui adhère à un abonnement téléphonique ou internet
Ligne téléphonique	c'est le produit qui permet au client de communiquer avec téléphone (RTC ou FTTH)
Internet	c'est le produit qui permet au client de communiquer avec internet (ADSL ou FTTx)
LTE 4G	c'est le produit qui permet au client de communiquer avec technologie sans fil LTE 4G

h- Construction des hiérarchies de concepts

Chaque hiérarchie organise un groupe de concepts sous forme d'une taxonomie.

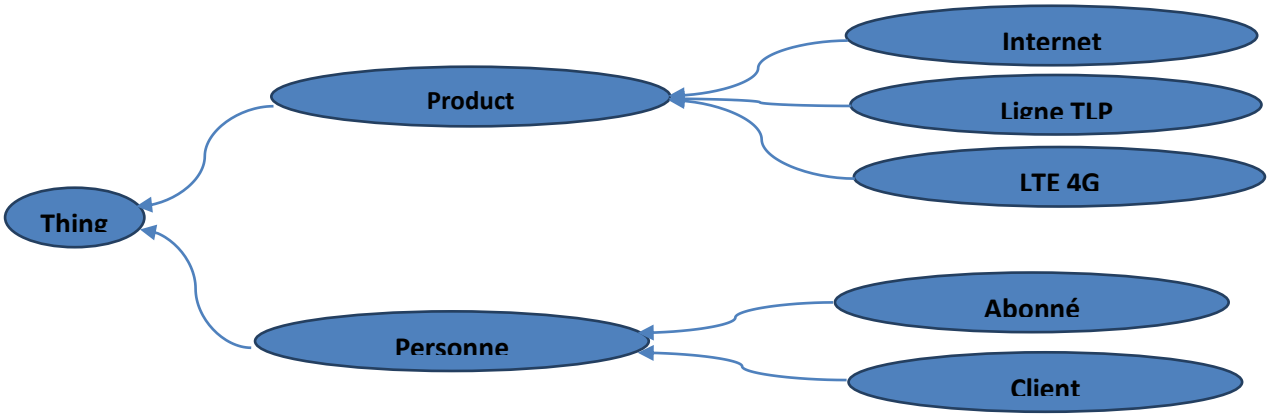


Figure N° 18 : La hiérarchie de concepts de l'ontologie Algérie Telecom

i- Construction d'un diagramme des relations binaires

Ce diagramme permet de représenter de manière graphique les différentes relations qui existent entre les divers concepts de même ou de différentes hiérarchies.

NB : Afin de renforcer notre ontologie on a téléchargé l'ontologie de GOOGLE (schema.org), on a importé cette ontologie sous la nôtre et on a fusionné les deux ontologies.

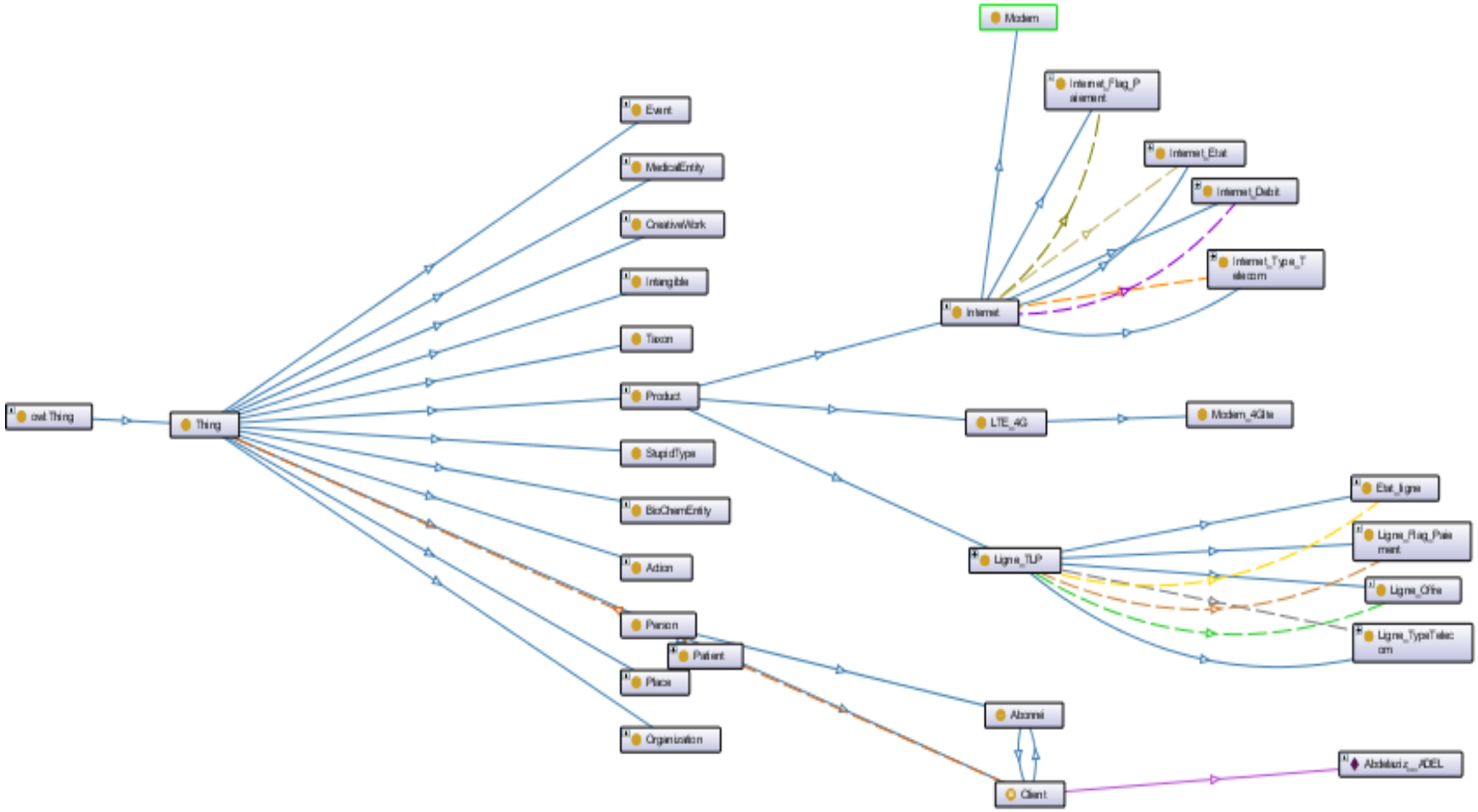


Figure N°19 : Graphe de l'ontologie Algérie Telecom intégrée a celle de Google

j- Construire le dictionnaire des concepts

Dans cette étape, nous essayons de décrire chaque concept (classe) de façon détaillé qui contient :

- la liste des attributs (concept, ses attribut, concept père et ses synonymes si existent)
- la liste des relations (relation, prédécesseur, successeur et la définition). Qui sont définit dans les tableaux suivant :

k- Liste des attributs de différents concepts.

Concept	les attributs	Synonymes	Concept père
Person	-Nom -Prénom -Date de naissance -Adresse		
Client		Abonne	Person
Abonne		Client	Person
Product			
Internet	-Internet_ND -Internet_Expiration_Abonnement		Product
Ligne_TLP	-NumeroAppel -Ligne_NumeroPort		

l- La liste des relations entre les concepts.

Relation	Prédécesseur	Successeur	Définition
a_abonné_à1	Client	Ligne_TLP	Un client a abonné à une ligne téléphonique
a_abonné_à2	Client	Internet	Un client a abonné à internet
a_abonné_à3	Client	LTE_4G	Un client a abonné à un abonnement LTE 4G

3.1.2. Base d'apprentissage et de test :

Lors du développement d'un chatbot de relève de dérangement, l'une des étapes cruciales est la construction de la base de données. Cette base de données servira à entraîner le modèle du chatbot afin qu'il puisse comprendre et répondre aux requêtes liées aux dérangements signalés par les utilisateurs. Nous explorerons les différentes composantes de la base de données, y compris les exemples d'intentions, les entités associées et les réponses attendues.

Collecte des exemples d'intentions : Pour commencer, il est essentiel d'identifier les principales intentions que les utilisateurs peuvent avoir lorsqu'ils signalent un dérangement. Cela peut inclure des intentions telles que "Salutations", "derangement", "lenteur de connexion", etc. Il est important de couvrir un large éventail d'intentions pour garantir que le chatbot puisse répondre efficacement aux demandes des utilisateurs.

Chaque intention doit être accompagnée d'une liste d'exemples de modèles de phrases correspondants. Par exemple :

- pour l'intention "Salutations", les exemples de modèles de phrases pourraient inclure : "Salem", "bonjour", "salut" ;

- pour l'intention "derangement", les exemples de modèles de phrases pourraient inclure :

"J'ai un derangement", "j'ai un brobleme", ou tout simplement "derangement" ;

….etc. Ces modèles de phrases représentent les différentes façons dont les utilisateurs peuvent exprimer leur intention.

Identification des entités associées : Outre les intentions, il est également important d'identifier les entités ou les informations spécifiques que les utilisateurs peuvent fournir lorsqu'ils signalent un dérangement. Cela peut inclure des entités telles que le type de dérangement (internet, téléphone). L'identification de ces entités est essentielle pour extraire les informations pertinentes des requêtes des utilisateurs et fournir des réponses plus précises.

Chaque exemple de modèle de phrase doit être étudié pour identifier les entités potentielles qui y sont associées. Par exemple, dans l'exemple de modèle de phrase "derangement internet", l'entité associée peut être "internet". Cela permettra au chatbot de comprendre le type de dérangement signalé par l'utilisateur.

Définition des réponses attendues : Une fois que les exemples d'intentions et les entités associées ont été identifiés, il est nécessaire de définir les réponses attendues du chatbot pour chaque intention. Les réponses peuvent varier en fonction de la nature du dérangement signalé et des informations fournies par l'utilisateur. Il est recommandé d'avoir une variété de réponses pour chaque intention afin de rendre les interactions avec le chatbot plus dynamiques et naturelles. Par exemple, pour l'intention "j'ai un derangement", les réponses attendues pourraient inclure des messages de confirmation tels que "vous avez un derangement internet ou telephone" ou des demandes de renseignements supplémentaires tels que "Pouvez-vous fournir plus de détails sur le problème que vous rencontrez ". le tableau ci-dessous illustre plus d'exemple

	Tag	Patterns	Responses
Examlpe 01	"Salutations",	[" Salem", "cest quoi ton nom?", "Salut", "jai un probleme de connexion", "jai un problemme internet", "salem", "Salem", "salem", "salem", "salem", "bonjour", "salem"],	["Bonjour, Je suis ATBot, comment puis je vous aider aujourd"hui?",
Examlpe 02	"greeting",	["what is your name?", "Hello", "Hi", "jai un probleme de connexion", "jai un problemme internet", "hi", "jai un debit 20 M je ne recoit que 10 M"],	["Hello, I am ATBot, How can i help you to day?",
Examlpe 03	"au revoir",	["au revoir"],	["au revoir", "Bienvenue.. Algerie Telecom a votre service"],
Examlpe 04	"goodbye",	["OK, bye", "bye", "bye", "bye"],	["See you later", "Goodbye", "Goodbye", "See you later"],
Examlpe 05	"thanks",	["Thanks", "Thank you"],	["Happy to help!", "you are welcome"],
Examlpe 06	"merci",	["merci", "Merci a vous", "MERCI", "Bienvenue.. AlgÃ©rie TÃ©lÃ©com Ã votre service"],	["heureux de vous aider", "je vous en prie", "heureux de vous aider"],
Examlpe 07	"pas de question",	[""],	["Sorry, can"t understand you", "Desole, je ne peux pas vous comprendre", "Please give me more information", "Merci de me donner plus d"information"],
Examlpe 08	"options",	["Comment pourriez-vous m"aider ?", "que peux tu faire pour moi?", "wa andi dergt", "drgt", "tel", "makanche tana", "makanche tona", "drgt", "by", "babata", "homosse", , "oui dernagement", "tel", "comment va", "comment t"appel tu", "trico", "ok", "jai une drgt", "farine"],	["je peux vous aider a relever votre probleme ou derangement de telephone ou de connexion internet ","veuillez reformuler votre question SVP", "je m"appel Atbot", "je peux vous aider a relever votre probleme ou derangement de telephone ou de connexion internet ", , "veuillez reformuler votre question SVP", "je peux vous aider a relever votre probleme ou derangement de telephone ou de connexion internet ",],
Examlpe 09	"option",	["How you could help me?", "What you can do?"],	["I can help you to solve your problem or trouble with your phone or internet connection"],
Examlpe 10	"signalisation",	["048", "que peux tu faire pour moi?"],	["c"est pris en charge,votre derangement est signale aux service concernes "],
Examlpe 11	"signage",	["what can you do for me?"],	["it is taken care of, your disturbance is reported to the departments concerned "],
Examlpe 12	"derangement",	[" derangement", "j"ai un derangement", "jai un probleme", "je ne suis pas content", "jai derangement", "jai un derangement", "j"ai un derangement", "jai un derangement", "je ne suis pas satisfaite"],	["vous avez un derangement internet ou telephone", "Pouvez-vous fournir plus de dÃ©tails sur le problÃ¨me que vous rencontrez"],
Examlpe 13	"disturbance",	[" disturbance", "I have a problem", "I"m not happy", "qui es tu"],	["you have an internet or telephone problem", "you have an internet or telephone problem"],

Examlpe 14	"derangement de telephone",	["telephone", "Telephone coupe", "Derangement de telephone", "ligne occupe", "telephone", "j"ai un probleme de telephone", "yaw telephone"],	["veuillez brancher votre appareil telephonique avec la prise telephonique murale et verifiez s"il a de tonalite "],
Examlpe 15	"telephone trouble",	["phone", "Telephone disconnected", "Telephone fault", "no dial tone", "line busy", "phone", "I have a phone problem"],	["please connect your telephone device with the telephone wall socket and check if it has a dial tone"],
Examlpe 16	"pas de tonalite",	["pas de tonalite", "il y a du bruit dans la ligne", "pas tonalite", "pas tonalite", "pas de tonalite", "pas de tonalite"],	["veuillez changer le cordon reliant l"appareil telephonique a la prise murale", "veuillez changer le cordon reliant l"appareil telephonique a la prise murale", "veuillez changer le cordon reliant l"appareil telephonique a la prise murale",],
Examlpe 17	"no tone",	["no dial tone", "there is noise in the line"],	["please change the cord connecting the telephone set to the wall socket"],
Examlpe 18	"toujours pas de tonalite",	["j"ai change le cordon, toujours pas de tonalite", "je l"ai fait"],	["vueillez changer votre appareil telephonique et reverifier la tonalite"],
Examlpe 19	"still no dial tone",	["I changed the cord, still no tone"],	["please change your telephone device and recheck the dial tone"],
Examlpe 20	"derangement telephonique necessite AT",	["toujours rien, pas de tonalite", "le probleme de tonalite persiste", "toujours rien", "toujours pas de tonalite", "toujours rien", "rien"],	["donnez moi votre numero de telphone je vais vous envoyer une equipe d"intervention durant les deux jours qui suis pour regler votre probleme de ligne"],
Examlpe 21	"telephone fault requires AT",	["still nothing, no dial tone", "the tone problem persists"],	["give me your phone number I will send you an intervention team during the two days that follow to solve your line problem"],
Examlpe 22	"lenteur de connexion",	["internet lente", "chute de debit", "connexion lente", "cest une chute de debit flagrante", "chute de debit", "toujour connexion lente"],	["Essayez de brancher le modem directement avec la prise murale sans filtre et desactiver le wifi du modem puis faire un test de debit directement avec un ordinateur branche avec un cable reseau", "Essayez de brancher le modem directement avec la prise murale sans filtre et desactiver le wifi du modem puis faire un test de debit directement avec un ordinateur branche avec un cable reseau"],
Examlpe 23	"slow connection",	["slow connection", "drop in flow"],	["Try to connect the modem directly with the wall socket without filter and deactivate the wifi of the modem then do a speed test directly with a computer connected with a network cable"],
Examlpe 24	"derangement Internet",	["internet", "pb internet", "Internet coupe", "Derangement internet", "pas de connexion", "perturbation internet", "tjr pas tonalita", "internet", "internet", "internet"],	["veuillez verifier votre voyant internet du Modem s"il est allume en vert ou en rouge sinon veuillez verifier votre voyant adsl du modem s"il est fixe ou clignotant"],
Examlpe 25	"internet disruption",	["perturbation internet", "Internet cut", "Internet disturbance", "no connection", "internet disturbance"],	["please check your modem internet light if it is green or red if not please check your modem adsl light if it is steady or flashing"],

Examlpe 26	"voyant intrenet rouge",	["rouge", "le voyant internet du modem est rouge", "rouge", "rouge", "rouge"],	["veuillez recharger votre compte ADSL ou configuer votre modem , pour cela veuillez vous rapprocher de la plus proche de nos agence pour vous prendre en charge"],
Examlpe 27	"red internet light",	["red", "the internet light on the modem is red"],	["please recharge your ADSL account or configure your modem, for this please contact the nearest of our agencies to take care of you"],
Examlpe 28	"voyant intrenet vert",	["vert", "le voyant internet du modem est vert", "vert", "non vert", "vert"],	["Donc vous n"avez pas de probleme avec algerie telecom, veuillez verifier votre mot de passe wifi ou votre cable reseau"],
Examlpe 29	"green internet light",	["green", "the internet light on the modem is green"],	["So you have no problem with algerie telecom, please check your wifi password or your network cable"],
Examlpe 30	"vouyant adsl fixe",	["fixe", "le voyant adsl du modem est fixe", "il est fixe", "Mon numero de teephone est 048413337"],	["veuillez recharger votre compte ADSL ou configuer votre modem , pour cela veuillez vous rapprocher de la plus proche de nos agence pour vous prendre en charge"],
Examlpe 31	"fixed adsl light",	["fixed", "the adsl light on the modem is fixed"],	["please recharge your ADSL account or configure your modem, for this please contact the nearest of our agencies to take care of you"],
Examlpe 32	"voyant adsl clignotant",	["clignotant", "le voyant adsl du modem est clignotant"],	["donnez moi votre numero de telphone je vais vous envoyer une equipe d"intervention durant les deux jours qui suis pour regler votre probleme d"internet"],
Examlpe 33	"blinking adsl light",	["turn signal", "the modem"s adsl light is flashing"],	["give me your phone number I will send you an intervention team during the two days that follow to solve your line problem"],

Tableau des ntents utilisés dans la base de données

Prétraitement des données :

Une fois que tous les exemples d'intentions, les entités associées et les réponses attendues ont été collectés, il est temps de prétraiter les données. Cela implique généralement le nettoyage des textes, la tokenisation, la lemmatisation, la création de sacs de mots (bag-of-words) et l'encodage des intentions et des entités.

Nous avons utilisé des bibliothèques de traitement du langage naturel telles que NLTK (Natural Language Toolkit) pour faciliter ces tâches de prétraitement. Ces étapes permettent de représenter les données de manière appropriée pour l'entraînement du modèle du Atbot.

Division de la base de données :

Une étape essentielle dans la construction de la base de données est la division en une base d'apprentissage et une base de test. Cette division permet d'évaluer les performances du modèle d'atbot sur des données qu'il n'a pas encore vues. Dans notre cas, nous utilisons l'apprentissage supervisé, où nous fournissons des exemples étiquetés à notre modèle pour qu'il apprenne à associer les requêtes aux bonnes réponses.

Pour effectuer cette division, nous utilisons la technique de division aléatoire. Nous utilisons une proportion de 80% pour la base d'apprentissage et 20% pour la base de test. Cela signifie que 80% des exemples d'intentions et de leurs réponses associées sont utilisés pour l'apprentissage du modèle, tandis que les 20% restants sont réservés pour évaluer les performances du modèle.

Une fois que les exemples et les réponses ont été séparés en deux listes distinctes, nous utilisons des fonctions de division de données, telles que celles fournies par la bibliothèque scikit-learn en Python, pour effectuer cette division aléatoire. Cela garantit que les exemples sont répartis de manière équilibrée entre la base d'apprentissage et la base de test, évitant ainsi tout biais potentiel dans l'évaluation du modèle [10].

Avant d'utiliser les bases d'apprentissage et de test, il est important d'effectuer les étapes de prétraitement des données mentionnées précédemment, telles que le nettoyage des textes, la lemmatisation et la création de sacs de mots, sur chaque ensemble de données. Cela permet de préparer les données de manière cohérente pour l'apprentissage et l'évaluation du modèle.

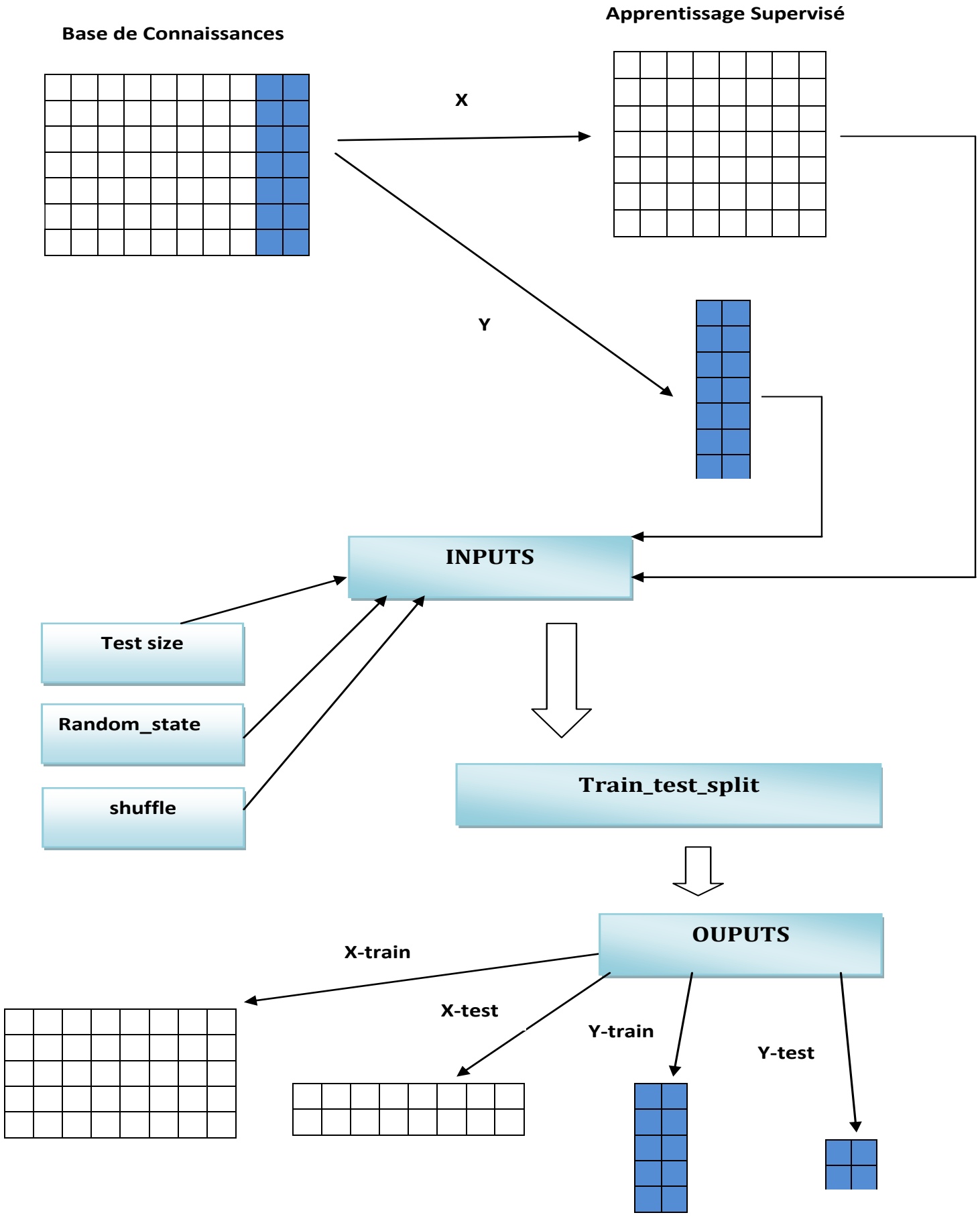


Figure N°20 : Schéma de division de base de données en base d'apprentissage et une base de test[10].

En conclusion, la division de la base de données en une base d'apprentissage et une base de test, en utilisant l'apprentissage, est une étape cruciale dans la construction d'un chatbot de relève de dérangement.

Evaluation du modèle sur les données de test :

L'inclusion du modèle graphique d'évolution de la précision et de la perte dans notre modèle entrainement pour Atbot revêt une importance significative. Ce modèle graphique permet de visualiser de manière concise et claire les performances du modèle au cours de l'entraînement. Le premier graphique affiche l'évolution de la précision du modèle au fil des époques. Chaque époque est représentée sur l'axe horizontal, tandis que la précision est représentée sur l'axe vertical. Cette représentation visuelle permet de suivre facilement les variations de la précision et d'identifier les améliorations ou les dégradations du modèle au fil du temps. Le deuxième graphique présente l'évolution de la perte du modèle au cours de l'entraînement. La perte est un indicateur de l'erreur du modèle, et un modèle performant aura une perte minimale. En observant ce graphique, nous pourrons détecter les tendances de la perte et prendre des décisions éclairées pour ajuster les paramètres du modèle et améliorer ses performances. Ces graphiques fournissent également une validation visuelle de l'efficacité de l'approche de modélisation et de l'optimisation du modèle. Ils permettent de communiquer efficacement les résultats obtenus lors de l'entraînement.

La formule utilisée pour évaluer les performances d'un modèle de machine learning sur un ensemble de données de test est:

1. `model.evaluate` : C'est une méthode qui calcule les métriques de performance du modèle sur un ensemble de données de test. Il prend en entrée les données de test (`test_X`) et les labels correspondants (`test_y`), et retourne les métriques de performance telles que la perte (loss) et l'exactitude (accuracy).
2. `np.array(test_X)` et **`np.array(test_y)`** : Ces instructions convertissent les données de test (`test_X`) et les labels correspondants (`test_y`) en tableaux NumPy, qui sont des structures de données couramment utilisées en calcul scientifique en Python.
3. `test_loss` : C'est une variable qui stocke la valeur de la perte (loss) obtenue lors de l'évaluation du modèle sur les données de test.
4. `test_accuracy` : C'est une variable qui stocke la valeur de l'exactitude (accuracy) obtenue lors de l'évaluation du modèle sur les données de test.

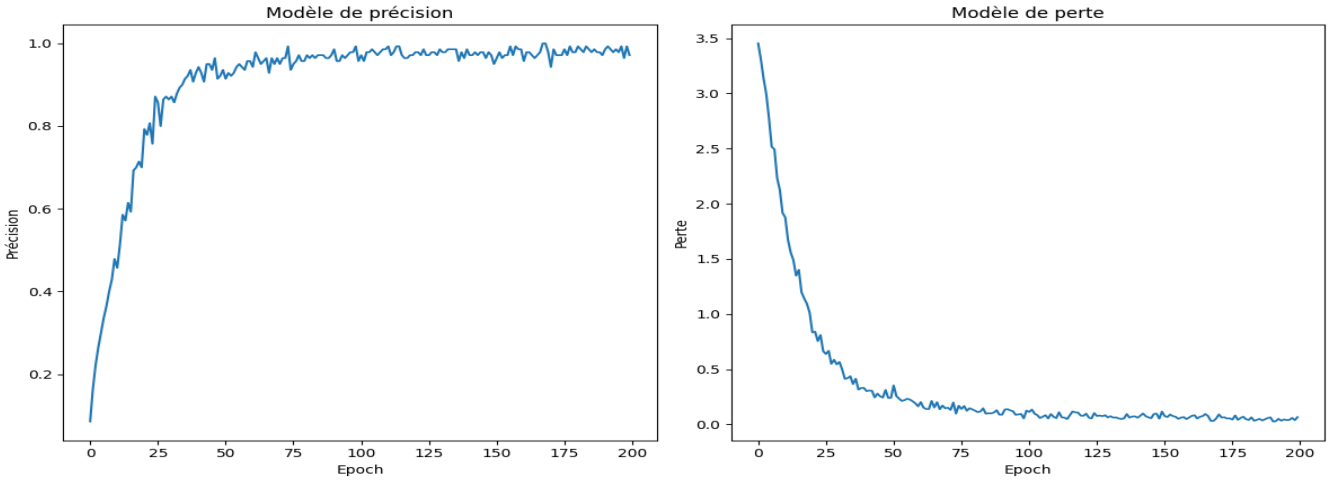


Figure N°21 : Évolution de la précision et de la perte du modèle Atbot au cours de l'entraînement

3.2. Implémentation du Atbot

Pour implémenter le chatbot, nous avons utilisé Keras, une bibliothèque Deep Learning et NLTK, qui est une boîte à outils de traitement du langage naturel ou Natural Language Processing (NLP). Nous avons aussi besoin de quelques bibliothèques utiles.

Tous les chatbots relèvent des concepts NLP (Natural Language Processing). La NLP comprend deux choses :

- NLU (Natural Language Understanding) : C'est la capacité des machines à comprendre le langage humain comme l'anglais. le français etc…
- NLG (Natural Language Generation): C'est la capacité d'une machine à générer du texte similaire à des phrases écrites par des humains.

Imaginez un utilisateur posant une question à un chatbot :

« Hi, quelles sont les nouvelles d'aujourd'hui ? »

Le chatbot va diviser la phrase d'utilisateur en deux choses :

1. L'intention et
2. L'entité

- L'intention de cette phrase pourrait être avoir_nouvelles car elle fait référence à une action que l'utilisateur veut faire.
- L'entité donne des détails spécifiques sur l'intention, donc « aujourd'hui » sera l'entité.

Ainsi, un modèle d'apprentissage automatique est utilisé pour reconnaître les intentions et les entités du chat.

Nous allons voir chaque fichier. Cela nous donnera une idée de la façon dont le projet sera mis en œuvre.

- **Train_Atbot.py** : Dans ce fichier, nous allons créer et former le modèle de deep learning ou apprentissage profond. Ce dernier va classer et identifier ce que l'utilisateur demande au robot.
- **Gui_Atbot.py** : C'est dans ce fichier que nous allons créer une interface utilisateur graphique pour tchatter avec notre chatbot formé.
- **Intents.json :** Le fichier d'intents contient toutes les données que nous allons utiliser pour former le modèle. Il comprend une collection de balises avec leurs modèles et réponses correspondants.
- **Atbot_model.h5 :** Il s'agit d'un fichier de format de données hiérarchique dans lequel nous allons sauvegarder les poids et l'architecture de notre modèle formé.
- **Classes.pkl :** Le fichier pickle peut être utilisé pour sauvegarder tous les noms de balises à classer lorsque nous prédisons le message.

- **Words.pkl :** Le fichier pickle words.pkl contient tous les mots uniques qui constituent le vocabulaire de notre modèle.
- **Json to RDF.py :** C'est dans ce fichier que nous allons convertir le fichier Intents.json en fichier intents.ttl et intentnts.rdf
- **RDF to Json :** C'est dans ce fichier que nous allons convertir le fichier Intents.rdf en fichier intents.json
- **RDF to Owl :** C'est dans ce fichier que nous allons convertir le fichier Intents.rdf en fichier intents.owl
- **Owl to RDF :** C'est dans ce fichier que nous allons convertir le fichier Intents.owl en fichier intents.rdf

Voici les étapes pour créer notre chatbot :

Étape 1. Importer des bibliothèques et charger les données :
Créer un nouveau fichier Python et le nomme « **train_Atbot** ». Ensuite, nous allons importer tous les modules requis. Puis, nous allons lire le fichier de données JSON dans notre programme Python.

Étape 2. Prétraiter les données :
Le modèle ne prend pas les données brutes. Il faut passer par beaucoup de pré-traitements pour que la machine puisse comprendre facilement. Pour les données textuelles, il y a plusieurs techniques de prétraitement. La première est la tokenisation. Cela consiste à diviser les phrases en mots. En observant le fichier d'*intents*, nous voyons que chaque balise contient une liste de modèles et de réponses. Nous *tokenisons* chaque modèle et ajoutons les mots dans une liste.

Nous créons aussi une liste de classes et de documents pour ajouter toutes les intentions associées aux modèles.

Une autre technique est la lemmatisation. Elle consiste à convertir les mots en forme de *lemme* pour réduire tous les mots canoniques. A titre d'exemple, les mots jouer, jouant, joue, joué, etc. seront tous remplacés par jouer. Ainsi, nous pouvons réduire le nombre total de mots dans notre vocabulaire. Alors maintenant, nous *lemmatisons* chaque mot et supprimons les mots en double. A la fin, les mots contiennent le vocabulaire de notre projet tandis que les classes comprennent toutes les entités à classer.

Pour enregistrer l'objet Python dans un fichier, nous avons utilisé la méthode pickle.dump().
pickle est un module de python qui permet de sauvegarder une ou plusieurs variables dans un fichier et de récupérer leurs valeurs ultérieurement. Les variables peuvent être de type quelconque. Il permet de stocker et de restaurer un objet Python tel quel sans aucune manipulation supplémentaire. Il fonctionne comme le module json mais n'est pas limité à un seul format d'objet.
Ces fichiers seront utiles une fois la formation terminée et lorsque nous prévoyons les tchats.
Étape 3. Créer des données de formation et de test :
Pour former le modèle, nous allons convertir chaque modèle d'entrée en nombres. Nous allons d'abord *lemmatiser* chaque mot du modèle. Pour cela, il faut créer une liste de zéros de la même longueur que le nombre de tous les mots. Nous allons définir la valeur 1 uniquement pour les

index qui contiennent le mot dans les modèles. De la même manière, nous créerons la sortie en définissant la valeur 1 pour la classe d'entrée pour laquelle appartient le modèle.

Étape 4. Former le modèle :
L'architecture de notre modèle sera un réseau neuronal composé de 3 couches denses. La première couche contient 128 neurones**,** la seconde a 64 et la dernière aura les mêmes neurones que le nombre de classes. Les couches de décrochage sont introduites pour réduire le sur-ajustement du modèle. Nous avons utilisé l'optimiseur L'algorithme d'optimisation Adam est utilisé pour la formation de modèles d'apprentissage profond. Il s'agit d'une extension de la descente de gradient stochastique. Dans cet algorithme d'optimisation, les moyennes courantes des gradients et des seconds moments des gradients sont utilisées. À chaque étape de la phase d'apprentissage, un mini lot d'échantillons est tiré de l'ensemble de données d'apprentissage et les poids du réseau de neurones sont ajustés en fonction des performances sur ce sous-ensemble spécifique d'exemples. La procédure d'échantillonnage par mini-lot introduit une dynamique stochastique à la descente de gradient, avec un bruit dépendant de l'état non trivial. Nous caractérisons la stochasticité d'ADAM et une variante récemment introduite, l'ADAM persistante, dans un modèle de réseau neuronal prototypique.

Lorsque la formation de **200 époques** sera terminée, nous enregistrons le modèle formé en utilisant la fonction Keras model.save (« A*tbot_model.h5* »).

Étape 5 convertir le fichier Intents.json en fichier intents.ttl et intentnts.rdf :
RDF est un format qui permet de représenter les données de manière sémantique en utilisant des triplets composés d'un sujet, d'un prédicat et d'un objet. La conversion se fait à l'aide de la bibliothèque Python `rdflib`. Le fichier JSON est chargé, un graphe RDF est créé et les données du fichier JSON sont ajoutées sous forme de triplets RDF dans ce graphe. Ensuite, le graphe RDF est enregistré dans deux fichiers différents :

- "intents.ttl" : Ce fichier est au format Turtle, une syntaxe lisible par les humains pour représenter les triplets RDF de manière concise. Il contient les triplets RDF du graphe dans une structure textuelle.

- "intents.rdf" : Ce fichier est au format RDF/XML, une représentation XML des triplets RDF. Il est souvent utilisé pour l'échange de données entre les applications.

La conversion du fichier JSON en fichiers RDF permet de profiter des avantages de la représentation sémantique des données, tels que la recherche et la réutilisation des informations dans différents contextes.

Étape 6 convertir le fichier Intents.rdf en fichier intents.json :
Cette conversion permet de représenter les données RDF dans un format plus couramment utilisé et plus facile à manipuler dans des applications et des systèmes qui prennent en charge JSON. Pour effectuer cette conversion, vous pouvez utiliser des bibliothèques ou des outils qui prennent en charge le traitement des données RDF et la conversion vers JSON. Par exemple, en utilisant RDFLib, une bibliothèque Python pour le traitement des données RDF, vous pouvez charger le fichier RDF, parcourir les triplets RDF et extraire les informations pertinentes pour

les convertir en une structure de données JSON. Il est important de noter que la conversion d'un fichier RDF en JSON peut impliquer des ajustements et des décisions spécifiques en fonction de la structure et de la sémantique des données RDF que vous souhaitez représenter dans le fichier JSON final.

Étape 7 convertir le fichier Intents.rdf en fichier intents.owl :

RDF est un modèle de données qui permet de représenter des informations sous forme de triplets sujet-prédicat-objet, tandis que OWL est un langage de représentation des ontologies utilisé pour décrire les connaissances formelles dans un domaine spécifique.

La conversion du fichier RDF en OWL implique la transformation de la structure et des informations contenues dans le fichier RDF en une représentation conforme aux spécifications OWL. Cela comprend la définition des classes, des propriétés, des individus et des relations dans l'ontologie OWL.

La conversion peut être réalisée en utilisant des outils et des bibliothèques qui prennent en charge le traitement des données RDF et la transformation vers le format OWL. Ces outils permettent de charger le fichier RDF, d'analyser sa structure et ses informations, puis de générer un fichier OWL correspondant.

Il est important de noter que la conversion du fichier RDF en OWL peut nécessiter des ajustements et des adaptations spécifiques en fonction de la structure et de la sémantique des données RDF utilisées. Cela peut inclure la gestion des types de données, des relations complexes ou des contraintes spécifiques à l'ontologie OWL cible.

Étape 8 convertir le fichier Intents.owl en fichier intents.rdf :

La conversion du fichier OWL en RDF implique la transformation de la structure et des informations contenues dans le fichier OWL en une représentation conforme aux spécifications RDF. Cela comprend la représentation des classes, des propriétés, des individus et des relations sous forme de triplets RDF.

La conversion peut être réalisée en utilisant des outils et des bibliothèques qui prennent en charge le traitement des données OWL et la transformation vers le format RDF. Ces outils permettent de charger le fichier OWL, d'analyser sa structure et ses informations, puis de générer un fichier RDF correspondant.

Il est important de noter que la conversion du fichier OWL en RDF peut nécessiter des ajustements et des adaptations spécifiques en fonction de la structure et de la sémantique des données OWL utilisées. Cela peut inclure la gestion des axiomes, des annotations ou des restrictions spécifiques à l'ontologie RDF cible.

Étape 9. Interagir avec le chatbot :

Notre modèle est prêt à chatter. Nous allons maintenant créer une belle interface graphique pour notre chatbot dans un nouveau fichier. Nous pouvons nommer le fichier comme gui_Atbot.py. Dans notre fichier GUI, nous utiliserons le module Tkinter pour construire la structure de l'application de bureau. Puis, nous allons capturer le message de l'utilisateur et refaire quelques prétraitements avant d'entrer le message dans notre modèle formé. Le modèle va ensuite prédire la balise du message de l'utilisateur. Puis, nous allons sélectionner aléatoirement la réponse dans la liste des réponses de notre fichier d'intentions ou d'« *intents* ».

3.3. Évaluation des performances du chatbot :

Avant le déploiement final, le chatbot sera soumis à des tests approfondis pour évaluer ses performances. Cela comprendra des tests fonctionnels pour vérifier si toutes les fonctionnalités fonctionnent correctement, des tests de performance pour évaluer la capacité du chatbot à gérer un volume élevé de requêtes, des tests d'interface utilisateur pour s'assurer que l'expérience utilisateur est fluide et conviviale, ainsi que des tests d'intégration pour vérifier la connexion avec les systèmes de gestion des dérangements. Les résultats de ces tests permettront d'identifier les éventuels problèmes et de les résoudre avant le déploiement final du chatbot.

Le développement et l'implémentation du chatbot de relève des dérangements nécessitent une approche méthodique et rigoureuse pour garantir un fonctionnement optimal. Cette phase est cruciale pour assurer que le chatbot répond aux besoins des utilisateurs et offre une expérience utilisateur satisfaisante.

Lors d'un test local composé de 20 questions, notre Atbot a donné des réponses logiques à 13 questions et des réponses illogiques à 7 questions.

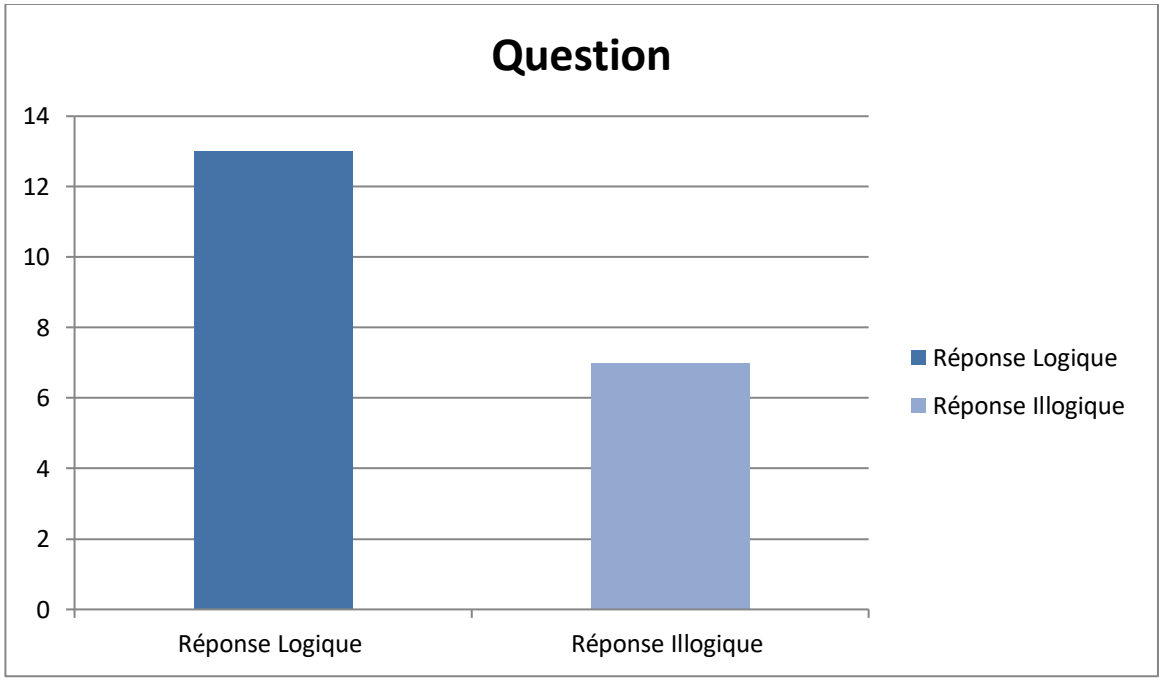


Figure N°22 : Résultats de test de performance pour Atbot

Ce test visait à évaluer la capacité du chatbot à comprendre et à répondre de manière cohérente aux différentes interrogations des utilisateurs. Les 13 réponses logiques démontrent que le chatbot a pu interpréter correctement la nature des questions posées et fournir des informations précises et pertinentes. Cependant, les 7 réponses illogiques indiquent que le chatbot peut

encore rencontrer des difficultés à comprendre certains types de questions ou à générer des réponses appropriées dans certaines situations.

Aussi nous avons réalisé un sondage auprès d'un échantillon de 100 personnes pour évaluer leur satisfaction vis-à-vis_de l'Atbot. Les participants ont été invités à donner leur opinion sur l'expérience utilisateur en attribuant une note de satisfaction allant de "plutôt satisfaisant" à "pas du tout satisfaisant", avec une catégorie supplémentaire pour les personnes sans avis.

Les résultats du sondage ont révélé que 39 personnes ont trouvé le Atbot "plutôt satisfaisant", tandis que 28 personnes l'ont jugé "tout à fait satisfaisant". De plus, 22 personnes ont indiqué qu'elles le trouvaient "plutôt pas satisfaisant", et 8 personnes l'ont évalué comme "pas du tout satisfaisant". Enfin, 3 personnes n'ont pas exprimé d'opinion claire et ont choisi la catégorie "sans avis".

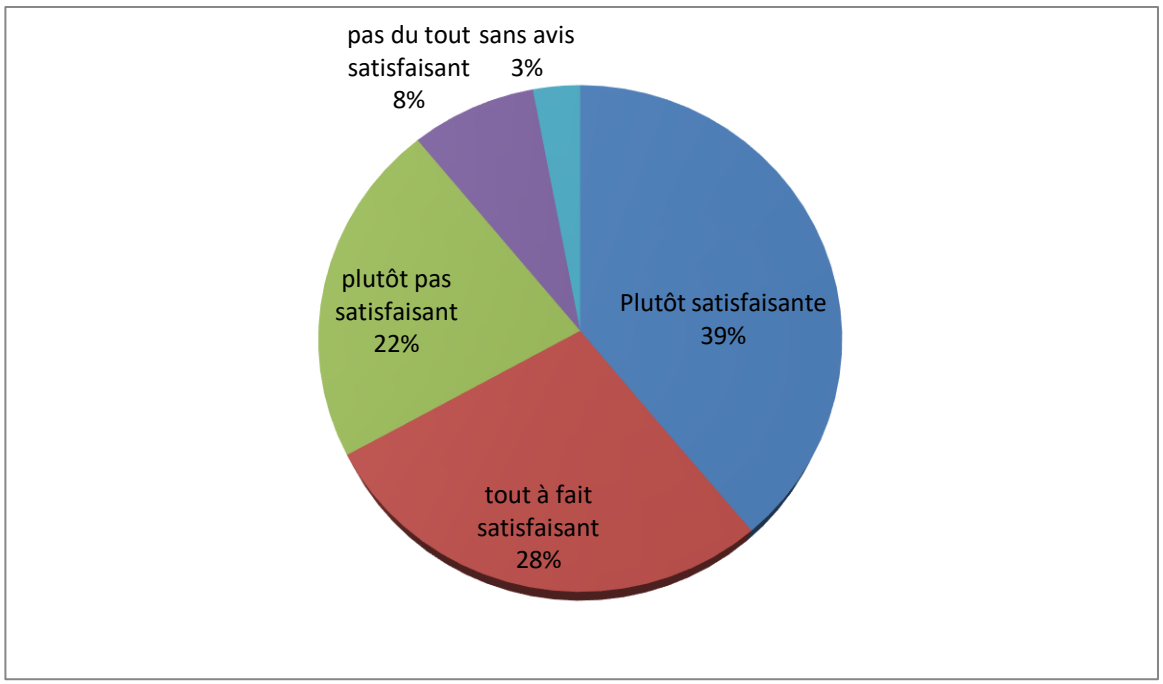


Figure N° 23 : Les Différentes Avis des Utilisateurs

Ces résultats soulignent l'importance de continuer à améliorer l'intelligence et la compréhension du Atbot en l'enrichissant avec des données supplémentaires, en affinant ses algorithmes de traitement du langage naturel et en optimisant ses modèles de réponse. Grâce à ce test, nous avons identifié des zones d'amélioration spécifiques pour renforcer la précision et la cohérence des réponses fournies par notre Atbot.

3.4. Code source :

- Train_atbot :

1. Importation des modules : Les modules nécessaires pour créer et entraîner le modèle de chatbot sont importés. Cela inclut des modules tels que `numpy`, `keras`, `tensorflow`, `random`, `string`, `nltk`, `json`, `pickle`, etc.
2. Lecture du fichier `intents.json` : Le fichier JSON contenant les intentions du chatbot est lu à l'aide de la fonction `open()` et `json.loads()`. Les intentions sont stockées dans la variable `intents`.
3. Prétraitement des données :
   - Les mots et les classes sont initialisés comme des listes vides.
   - Les phrases et les tags des intentions sont parcourus pour extraire les mots, les classes et les documents (phrases) correspondants.
   - Les mots sont lemmatisés, mis en minuscules et les ponctuations sont ignorées.
   - Les mots et les classes sont convertis en ensembles (pour supprimer les doublons) et triés.
4. Sauvegarde des mots et des classes : Les listes de mots et de classes obtenues après le prétraitement sont sauvegardées dans des fichiers pickle (`words.pkl` et `classes.pkl`).
5. Création des données d'entraînement :
   - Un tableau vide `training` est créé pour stocker les données d'entraînement.
   - Un tableau vide `output_empty` est créé pour représenter les sorties vides (0) correspondant à chaque classe.
   - Pour chaque document (phrase) et son tag, on construit un sac de mots (BoW) en utilisant un encodage one-hot.
   - Les sorties vides sont modifiées pour représenter la classe correspondante à l'aide de l'encodage one-hot.
   - Le BoW et les sorties correspondantes sont ajoutés à la liste `training`.
6. Entraînement du modèle de chatbot :
   - Les données d'entraînement sont mélangées et converties en un tableau numpy.
   - Les features et les labels sont extraits des données d'entraînement.
   - Le modèle de chatbot est créé à l'aide de la classe `Sequential` de Keras.
   - Les couches du modèle sont ajoutées : une couche d'entrée, des couches cachées avec des fonctions d'activation ReLU et des couches de dropout, et une couche de sortie avec une fonction d'activation softmax.
   - Le modèle est compilé avec une fonction de perte de `categorical_crossentropy` et l'optimiseur Adam.
   - L'entraînement du modèle est effectué avec la méthode `fit()`, en utilisant les features et les labels d'entraînement, un nombre d'époques et une taille de batch spécifiés.
7. Sauvegarde du modèle entraîné : Le modèle entraîné est sauvegardé dans un fichier `Atbot_model.h5`.
8. Fonctions d'aide pour la prédiction et la génération de réponses :
   - La fonction `clean_text()` tokenise et lemmatize le texte d'entrée.
   - La fonction `bag_of_words()` crée un sac de mots (BoW) à partir du texte d'entrée et du vocabulaire.
   - La fonction `pred_class()` prédit la classe à laquelle appartient le texte d'entrée en utilisant le modèle de chatbot.

- La fonction `get_response()` génère une réponse à partir de la classe prédite en choisissant une réponse aléatoire parmi les réponses disponibles dans les intentions.

9. Conversion du fichier JSON en fichier CSV : Le fichier `intents.json` est converti en un fichier CSV `intents.csv` en utilisant la bibliothèque pandas.

10. Conversion du fichier JSON en fichier RDF : Le fichier `intents.json` est converti en un fichier RDF `intents.rdf` en utilisant la bibliothèque RDFLib.

11. Conversion du fichier RDF en fichier JSON: Le fichier `intents.rdf' est converti en un fichier JSON `intents.json` en utilisant la bibliothèque RDFLib.

12. Importation du module GUI : Le fichier `gui_Atbot.py` est importé pour exécuter l'interface graphique du chatbot.

```
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
from keras.optimizers import adam_legacy
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout
import tensorflow as tf
import random
import string
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import json
import pickle
nltk.download("punkt")
nltk.download("wordnet")

words=[]
classes = []
doc_x = []
doc_y = []
ignore_letters = ['!', '?', ',', '.']
intents_file = open('intents.json').read()
intents = json.loads(intents_file)

for intent in intents['intents']:
    for pattern in intent['patterns']:
        #tokéniser chaque mot
        tokens = nltk.word_tokenize(pattern)
        words.extend(tokens)
        #ajouter des documents dans le corpus
        doc_x.append(pattern)
        doc_y.append(intent["tag"])
        # ajouter à notre liste de classes
        if intent['tag'] not in classes:
            classes.append(intent['tag'])
print(doc_x)
print(doc_y)
```

```
# lemmaziser et abaisser chaque mot et supprimer les doublons
words = [lemmatizer.lemmatize(word.lower()) for word in words if word not in
string.punctuation]
words = sorted(list(set(words)))
# trier les classes
classes = sorted(list(set(classes)))
# documents x et documents y = modèles et intentions
print (len(doc_x), "documents_x")
print (len(doc_y), "documents_y")
# classes = intents
print (len(classes), "classes", classes)
# words = tous les mots, vocabulaire
print (len(words), "unique lemmatized words", words)

pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))

# créer nos données d'entraînement
training = []
# créer un tableau vide pour notre sortie
output_empty = [0] * len(classes)
# ensemble d'entraînement, sac de mots pour chaque phrase
# liste pour les données d'entraînement
training = []
out_empty = [0] * len(classes)

# création du modèle d'ensemble de mots
for idx, doc in enumerate(doc_x):
    bow = []
    text = lemmatizer.lemmatize(doc.lower())
    for word in words:
        bow.append(1) if word in text else bow.append(0)

    # marque l'index de la classe à laquelle le pattern atguel est associé à
    output_row = list(out_empty)
    output_row[classes.index(doc_y[idx])] = 1

    # ajoute le one hot encoded BoW et les classes associées à la liste
training
    training.append([bow, output_row])

# mélanger les données et les convertir en array
random.shuffle(training)
training = np.array(training, dtype=object)

# séparer les features et les labels target
train_X = np.array(list(training[:, 0]))
train_y = np.array(list(training[:, 1]))
# définition de quelques paramètres
input_shape = (len(train_X[0]),)
output_shape = len(train_y[0])
epochs = 200
```

```
# modèle Deep Learning
model = Sequential()
model.add(Dense(128, input_shape=input_shape, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="relu"))
model.add(Dropout(0.3))
model.add(Dense(output_shape, activation = "softmax"))

adam = tf.keras.optimizers.Adam(
    learning_rate=0.001,
    beta_1=0.9,
    beta_2=0.999,
    epsilon=1e-07,
    amsgrad=False,
    weight_decay=None,
    clipnorm=None,
    clipvalue=None,
    global_clipnorm=None,
    use_ema=False,
    ema_momentum=0.99,
    ema_overwrite_frequency=None,
    jit_compile=True,
    name="Adam",
)
model.compile(loss='categorical_crossentropy', optimizer=adam,
metrics=["accuracy"])
print(model.summary())
#ajustement et sauvegarde du modèle
hist = model.fit(np.array(train_X), np.array(train_y), epochs=200,
batch_size=5, verbose=1)
model.save('Atbot_model.h5', hist)

print("model created")

def clean_text(text):
  tokens = nltk.word_tokenize(text)
  tokens = [lemmatizer.lemmatize(word) for word in tokens]
  return tokens

def bag_of_words(text, vocab):
  tokens = clean_text(text)
  bow = [0] * len(vocab)
  for w in tokens:
    for idx, word in enumerate(vocab):
      if word == w:
        bow[idx] = 1
  return np.array(bow)
```

```
def pred_class(text, vocab, labels):
  bow = bag_of_words(text, vocab)
  result = model.predict(np.array([bow]))[0]
  thresh = 0.2
  y_pred = [[idx, res] for idx, res in enumerate(result) if res > thresh]

  y_pred.sort(key=lambda x: x[1], reverse=True)
  return_list = []
  for r in y_pred:
    return_list.append(labels[r[0]])
  return return_list

def get_response(intents_list, intents_json):
  tag = intents_list[0]
  list_of_intents = intents_json["intents"]
  for i in list_of_intents:
    if i["tag"] == tag:
      result = random.choice(i["responses"])
      break
  return result
import pandas as pd

df = pd.read_json('intents.json')
df.to_csv('intents.csv')
import Json_to_RDF.py
import RDF_to_Json
import gui_Atbot.py
```

- **Gui atbot :**

Certaines bibliothèques ont été importées dans le programme pour des fonctionnalités spécifiques :

- `**nltk**`: C'est la bibliothèque Natural Language Toolkit qui est utilisée pour le traitement du langage naturel. Elle est utilisée ici pour la lemmatisation des mots (WordNetLemmatizer), qui permet de réduire les mots à leur forme de base.

- `**pickle**`: C'est une bibliothèque Python pour la sérialisation et la désérialisation d'objets Python. Elle est utilisée ici pour charger les fichiers pickle contenant les données prétraitées (words.pkl, classes.pkl) qui sont utilisées pour la classification des intentions.

- `**numpy**`: C'est une bibliothèque Python pour le calcul scientifique. Elle est utilisée ici pour créer des tableaux numpy qui représentent les sacs de mots (bag_of_words).

- `**keras.models**`: C'est un module de la bibliothèque Keras, qui est un framework d'apprentissage automatique et de deep learning. Il est utilisé ici pour charger le modèle de réseau neuronal (Atbot_model.h5) qui est utilisé pour la prédiction de l'intention.

- `**json**`: C'est une bibliothèque Python pour le travail avec des fichiers JSON (JavaScript Object Notation). Elle est utilisée ici pour charger le fichier intents.json qui contient les données des intentions et des réponses.

- `**random**`: C'est un module Python pour la génération de nombres aléatoires. Il est utilisé ici pour choisir une réponse aléatoire parmi les réponses possibles.

- `**PIL**`: C'est la bibliothèque Python Imaging Library, qui est utilisée pour le traitement des images. Elle est utilisée ici pour charger et afficher l'avatar du chatbot.

- `**tkinter**`: C'est une bibliothèque Python pour créer des interfaces graphiques. Elle est utilisée ici pour créer la fenêtre de chat et les différents composants graphiques (Text, Button, etc.).

Ces bibliothèques sont importées pour fournir les fonctionnalités nécessaires à l'exécution du programme de chatbot, telles que le traitement du langage naturel, la classification des intentions, l'affichage de l'interface graphique et l'affichage de l'avatar du chatbot.

2. Chargement du modèle de chatbot et des données :

- Le modèle de chatbot est chargé à partir du fichier 'Atbot_model.h5' à l'aide de la fonction `load_model` de Keras.

- Les données d'intention (intents) sont chargées à partir du fichier 'intents.json' à l'aide de la fonction `json.loads`.

- Les mots et les classes sont chargés à partir des fichiers 'words.pkl' et 'classes.pkl' respectivement à l'aide de la fonction `pickle.load`.

3. Définition de fonctions pour le traitement du texte :

- `clean_up_sentence(sentence)` : Cette fonction prend une phrase en entrée, la tokenise en mots individuels à l'aide de la bibliothèque NLTK et applique la lemmatisation à chaque mot à l'aide de `WordNetLemmatizer`.

- `bag_of_words(sentence, words, show_details=True)` : Cette fonction crée un sac de mots pour une phrase donnée en utilisant les mots fournis et la fonction `clean_up_sentence`. Chaque mot est représenté par un 1 s'il est présent dans la phrase et un 0 sinon.

- `predict_class(sentence)` : Cette fonction prédit l'intention d'une phrase en utilisant le modèle de chatbot. Elle utilise la fonction `bag_of_words` pour obtenir le sac de mots correspondant à la phrase, puis utilise le modèle pour prédire l'intention. Les prédictions avec une probabilité supérieure à un seuil prédéfini sont renvoyées.

- `getResponse(ints, intents_json)` : Cette fonction récupère une réponse aléatoire correspondant à l'intention prédite à partir des données d'intention (intents).

4. Définition de la fonction `send()` :

- Cette fonction est appelée lorsque l'utilisateur envoie un message.

- Elle récupère le message saisi par l'utilisateur à partir de la boîte de saisie.

- Elle utilise les fonctions `predict_class` et `getResponse` pour obtenir la réponse du chatbot.

- Elle affiche le message de l'utilisateur et la réponse du chatbot dans la fenêtre de discussion.

5. Création de l'interface graphique à l'aide de la bibliothèque Tkinter :

- Une fenêtre principale est créée avec une taille fixe.

- Une fenêtre de discussion est créée pour afficher les messages de l'utilisateur et du chatbot.

- Une barre de défilement est ajoutée à la fenêtre de discussion pour faciliter la navigation dans les messages.

- Une boîte de saisie est

ajoutée pour permettre à l'utilisateur d'entrer des messages.

- Un bouton "envoyé" est ajouté pour envoyer le message de l'utilisateur.

- Les composants sont placés à l'aide de coordonnées x et y.

6. Lancement de la boucle principale de l'interface graphique :

- La boucle `mainloop()` est exécutée pour afficher la fenêtre de l'interface graphique et attendre les interactions de l'utilisateur.

L'interface graphique permet à l'utilisateur d'envoyer des messages au chatbot et d'afficher les réponses dans la fenêtre de discussion. Le chatbot utilise le modèle de Deep Learning chargé pour prédire l'intention des messages de l'utilisateur et générer des réponses appropriées à partir des données d'intention fournies.

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
import pickle
import numpy as np

from keras.models import load_model
model = load_model('Atbot_model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
```

```
def clean_up_sentence(sentence):
    # tokeniser le modèle - diviser les mots en tableau
    sentence_words = nltk.word_tokenize(sentence)
    # radicaliser chaque mot - réduire à la forme de base
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words
# retour sac de mots tableau : 0 ou 1 pour les mots qui existent dans la phrase
def bag_of_words(sentence, words, show_details=True):
    # modèles de tokenisation
    sentence_words = clean_up_sentence(sentence)
    #sac de mots - matrice de vocabulaire
    bag = [0]*len(words)
    for s in sentence_words:
        for i,word in enumerate(words):
            if word == s:
                # attribuer 1 si le mot actuel est dans la position du vocabulaire
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % word)
    return(np.array(bag))
```

```python
def predict_class(sentence):
    # filtrer les prédictions en dessous du seuil
    p = bag_of_words(sentence, words,show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i,r] for i,r in enumerate(res) if r>ERROR_THRESHOLD]
    # probabilité de force de tri
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list

def getResponse(ints, intents_json):
    tag = ints[0]['intent']
    list_of_intents = intents_json['intents']
    for i in list_of_intents:
        if(i['tag']== tag):
            result = random.choice(i['responses'])
            break
    return result
```

```python
#Création de l'interface graphique de tkinter
import tkinter
from tkinter import *

def send():
    msg = EntryBox.get("1.0",'end-1c').strip()
    EntryBox.delete("0.0",END)

    if msg != '':
        ChatBox.config(state=NORMAL)
        ChatBox.insert(END, "You: " + msg + '\n\n')
        ChatBox.config(foreground="#446665", font=("Verdana", 12 ))

        ints = predict_class(msg)
        res = getResponse(ints, intents)

        ChatBox.insert(END, "Bot: " + res + '\n\n')

        ChatBox.config(state=DISABLED)
        ChatBox.yview(END)


root = Tk()
root.title("Atbot")
root.geometry("400x500")
root.resizable(width=FALSE, height=FALSE)
```

```
#Créer une fenêtre de chat
ChatBox = Text(root, bd=0, bg="light green", height="8", width="50",
font="Cambria",)

ChatBox.config(state=DISABLED)

#Lier la barre de défilement à la fenêtre de discussion
scrollbar = Scrollbar(root, command=ChatBox.yview, cursor="heart")
ChatBox['yscrollcommand'] = scrollbar.set

#Créer un bouton pour envoyer un message
SendButton = Button(root, font=("green",12,'bold'), text="envoyé", width="12",
height=5,
                    bd=0, bg="#f9a602",
activebackground="#3c9d9b",fg='#000000',
                    command= send )

#Créer la boîte pour saisir le message
EntryBox = Text(root, bd=0, bg="light blue",width="29", height="5",
font="Cambria")
#EntryBox.bind("<Return>", send)
```

```
#Placer tous les composants sur l'écran
scrollbar.place(x=376,y=10, height=386)
ChatBox.place(x=10,y=10, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=250)
SendButton.place(x=14, y=401, height=90)

root.mainloop()
```

- **Intents Json**

Ce fichier JSON contient des données d'intention (intents) pour un chatbot. Chaque intention est définie avec les champs suivants :

- "tags" : Une étiquette représentant l'intention.

- "patterns" : Une liste de motifs ou de phrases qui peuvent être associées à cette intention.

- "responses" : Une liste de réponses possibles pour cette intention.

Voici une explication des intentions présentes dans le fichier :

1. "no tone" : Cette intention est associée à des motifs tels que "no dial tone" et "there is noise in the line". La réponse suggère de changer le cordon reliant le téléphone à la prise murale.

2. "derangement de telephone" : Cette intention est associée à des motifs tels que "Telephone coupe" et "Derangement de telephone". La réponse suggère de brancher l'appareil téléphonique à la prise murale et de vérifier s'il y a une tonalité.

3. "option" : Cette intention est associée à des motifs tels que "How you could help me?" et "What you can do?". La réponse indique que le chatbot peut aider à résoudre les problèmes ou les perturbations liés aux téléphones ou à la connexion Internet.

4. "slow connection" : Cette intention est associée à des motifs tels que "slow connection" et "drop in flow". La réponse suggère de connecter le modem directement à la prise murale sans filtre, de désactiver le Wi-Fi du modem et de faire un test de débit avec un ordinateur connecté par câble réseau.

5. "voyant adsl clignotant" : Cette intention est associée à des motifs tels que "clignotant" et "le voyant adsl du modem est clignotant". La réponse suggère de fournir le numéro de téléphone afin d'envoyer une équipe d'intervention pour résoudre le problème d'Internet dans les deux jours suivants.

6. "lenteur de connexion" : Cette intention est associée à des motifs tels que "internet lente" et "chute de debit". La réponse suggère de brancher le modem directement à la prise murale sans filtre, de désactiver le Wi-Fi du modem et de faire un test de débit avec un ordinateur connecté par câble réseau.

7. "options" : Cette intention est associée à des motifs tels que "Comment pourriez-vous m'aider ?" et "que pouvez-vous faire pour moi ?". La réponse indique que le chatbot peut aider à résoudre les problèmes ou les perturbations liés aux téléphones ou à la connexion Internet.

8. "internet disruption" : Cette intention est associée à des motifs tels que "perturbation internet" et "Internet cut". La réponse suggère de vérifier les voyants Internet et ADSL du modem pour déterminer s'ils sont allumés en vert ou en rouge.

9. "Salutations" : Cette intention est associée à des motifs tels que "Salem" et "Salut". La réponse de salutation du chatbot est "Bonjour, Je suis ATBot, comment puis-je vous aider aujourd'hui ?"

10. "pas de question" : Cette intention n'a pas de motifs définis, mais elle est associée à des réponses telles que "Désolé, je ne peux pas vous comprendre" et "Merci de me donner plus d'information" lorsque

le message de l'utilisateur ne correspond à aucune autre intention.

11. "goodbye" : Cette intention est associée à des motifs tels que "OK, bye" et "bye". Les réponses du chatbot sont "See you later" et "Goodbye".

12. "fixed adsl light" : Cette intention est associée à des motifs tels que "fixed" et "the adsl light on the modem is fixed". La réponse suggère de recharger le compte ADSL ou de configurer le modem en contactant l'une des agences les plus proches.

13. "telephone trouble" : Cette intention est associée à des motifs tels que "Telephone disconnected" et "Telephone fault". La réponse suggère de connecter l'appareil téléphonique à la prise murale et de vérifier s'il y a une tonalité.

14. "derangement Internet" : Cette intention est associée à des motifs tels que "pb internet" et "Internet coupe". La réponse suggère de vérifier les voyants Internet et ADSL du modem pour déterminer s'ils sont allumés en vert ou en rouge.

15. "green internet light" : Cette intention est associée à des motifs tels que "green" et "the internet light on the modem is green". La réponse suggère de vérifier le mot de passe Wi-Fi ou le câble réseau si le voyant Internet est vert.

16. "greeting" : Cette intention est associée à des motifs tels que "what is your name?" et "Hello". La réponse de salutation du chatbot est "Hello, I am ATBot, How can i help you today?"

17. "au revoir" : Cette intention est associée au motif "au revoir". La réponse du chatbot est également "au revoir".

18. "derangement telephonique necessite AT" : Cette intention est associée à des motifs tels que "toujours rien, pas de tonalite" et "le problème de tonalite persiste". La réponse suggère de fournir le numéro de téléphone afin d'envoyer une équipe d'intervention pour résoudre le problème de ligne.

19. "voyant internet rouge" : Cette intention est associée à des motifs tels que "rouge" et "le voyant internet du modem est rouge". La réponse suggère de recharger le compte ADSL ou de configurer le modem en contactant l'une des agences les plus proches.
20. "pas de tonalite" : Cette intention est associée à des motifs tels que "pas de tonalite" et "il y a du bruit dans la ligne". La réponse suggère de changer le cordon reliant l'appareil téléphonique à la prise murale.

21. "voyant internet vert" : Cette intention est associée à des motifs tels que "vert" et "le voyant internet du modem est vert". La réponse suggère de vérifier le mot de passe Wi-Fi ou le câble réseau si le voyant Internet est vert.

22. "telephone fault requires AT" : Cette intention est associée à des motifs tels que "still nothing, no dial tone" et "the tone problem persists". La réponse suggère de fournir le numéro de téléphone afin d'envoyer une équipe d'intervention pour résoudre le problème de ligne.

23. "red internet light" : Cette intention est associée à des motifs tels que "red" et "the internet light on the modem is red". La réponse suggère de recharger le compte ADSL ou de configurer le modem en contactant l'une des agences les plus proches.

24. "blinking adsl light" : Cette intention est associée à des motifs tels que "turn signal" et "the modem's adsl light is flashing". La réponse suggère de fournir le numéro de téléphone afin d'envoyer une équipe d'intervention pour résoudre le problème de ligne.

25. "thanks" : Cette intention est associée à des motifs tels que "Thanks" et "Thank you". Les réponses du chatbot sont "Happy to help!" et "you are welcome".

26. "toujours pas de tonalite" : Cette intention est associée au motif "j'ai changé le cordon, toujours pas de tonalite". La réponse suggère de changer l'appareil téléphonique et de vérifier à nouveau la tonalité.

27. "merci" : Cette intention est associée à des motifs tels que "merci" et "Merci a vous". Les réponses du chatbot sont "heureux de vous aider" et "je vous en prie".

28. "still no dial tone" : Cette intention est associée au motif "I changed the cord, still no tone". La réponse suggère de changer l'appareil téléphonique et de vérifier à nouveau la tonalité.29. "vouyant adsl fixe" : Cette intention est associée à des motifs tels que "fixe" et "le voyant adsl du

modem est fixe". La réponse suggère de recharger le compte ADSL ou de configurer le modem en contactant l'une des agences les plus proches.

Si l'utilisateur envoie un message qui ne correspond à aucune des intentions répertoriées ci-dessus, le chatbot répondra par défaut avec l'une des réponses de l'intention "pas de question".

```
{"intents":
    [
        {"tag": "Salutations",
         "patterns": [" Salem", "cest quoi ton nom?", "Salut", "jai un probleme de
connexion", "jai un problemme internet"],
         "responses": ["Bonjour, Je suis ATBot, comment puis je vous aider
aujourd'hui?"],
         "context": [""]
        },

        {"tag": "greeting",
         "patterns": ["what is your name?", "Hello", "Hi", "jai un probleme de
connexion", "jai un problemme internet"],
         "responses": ["Hello, I am ATBot, How can i help you to day?"],
         "context": [""]
        },

        {"tag": "au revoir",
         "patterns": ["au revoir"],
         "responses": ["au revoir"],
         "context": [""]
        },

        {"tag": "goodbye",
         "patterns": ["OK, bye","bye"],
         "responses": ["See you later", "Goodbye"],
         "context": [""]
        },

        {"tag": "thanks",
         "patterns": ["Thanks", "Thank you"],
         "responses": ["Happy to help!", "you are welcome"],
         "context": [""]
        },

        {"tag": "merci",
        "patterns": [ "merci", "Merci a vous"],
        "responses": ["heureux de vous aider", "je vous en prie"],
        "context": [""]
        },

        {"tag": "pas de question",
         "patterns": [],
         "responses": ["Sorry, can't understand you", "Désolé, je ne peux pas vous
comprendre", "Please give me more information", "Merci de me donner plus
d'information"],
```

```
 {"tag": "options",
         "patterns": ["Comment pourriez-vous m'aider ?", "que peux tu faire pour
moi?", ""],
         "responses": ["je peux vous aider a relevé votre probleme ou derrangement
de telephone ou de connexion internet "],
         "context": [""]
        },

        {"tag": "option",
        "patterns": ["How you could help me?", "What you can do?"],
        "responses": ["I can help you to solve your problem or trouble with your
phone or internet connection"],
        "context": [""]
       },

        {"tag": "signalisation",
        "patterns": ["048", "que peux tu faire pour moi?"],
        "responses": ["c'est pris en charge,votre derangement est signalé aux
service concernés "],
        "context": [""]
       },

       {"tag": "signage",
        "patterns": ["048", "what can you do for me?"],
        "responses": ["it is taken care of, your disturbance is reported to the
departments concerned "],
        "context": [""]
       },

        {"tag": "derangement",
         "patterns": [" derangement", "j'ai un derangement", "jai un probleme", "je
ne suis pas content"],
         "responses": ["vous avez un derangement internet ou telephone", "vous avez
un probleme internet ou telephone"],
         "context": [""]
        },

        {"tag": "disturbance",
         "patterns": [" disturbance", "I have a problem", "I'm not happy"],
         "responses": ["you have an internet or telephone problem"],
         "context": [""]
        },


        {"tag": "derangement de telephone",
         "patterns": ["Telephone coupe", "Derangement de telephone", "ligne
occupe", "telephone", "j'ai un probleme de telephone"],
         "responses": ["veuillez brancher votre appareil telephonique  avec la
prise téléphonique  murale et verifiez s'il a de tonalité "],
         "context": [""]
        },
```

```
{"tag": "telephone trouble",
        "patterns": ["Telephone disconnected", "Telephone fault", "no dial tone",
"line busy", "phone", "I have a phone problem" ],
        "responses": ["please connect your telephone device with the telephone wall
socket and check if it has a dial tone"],
        "context": [""]
        },

        {"tag": "pas de tonalite",
         "patterns": ["pas de tonalite", "il y a du bruit dans la ligne"],
         "responses": ["veuillez changer le cordon reliant l'appareil telephonique
a la prise murale"],
         "context": [""]
        },

        {"tag": "no tone",
        "patterns": ["no dial tone", "there is noise in the line"],
        "responses": ["please change the cord connecting the telephone set to the
wall socket"],
        "context": [""]
        },

        {"tag": "toujours pas de tonalite",
         "patterns": ["j'ai change le cordon, toujours pas de tonalite","je l'ai
fait" ],
         "responses": ["vueillez changer votre appareil telephonique et reverifier
la tonalite"],
         "context": [""]
        },

        {"tag": "still no dial tone",
         "patterns": ["I changed the cord, still no tone" ],
         "responses": ["please change your telephone device and recheck the dial
tone"],
         "context": [""]
        },

        {"tag": "derangement telephonique necessite AT",
         "patterns": ["toujours rien, pas de tonalite", "le probleme de tonalite
persiste"],
         "responses": ["donnez moi votre numero de telphone je vais vous envoyé une
equipe d'intervention durant les deux jours qui suis pour regler votre probleme de
ligne"],
         "context": [""]
        },
 {"tag": "telephone fault requires AT",
        "patterns": ["still nothing, no dial tone", "the tone problem persists"],
        "responses": ["give me your phone number I will send you an intervention
team during the two days that follow to solve your line problem"],
        "context": [""]
        },
```

```
 {"tag": "lenteur de connexion",

         "patterns": [ "internet lente", "chute de debit"],
         "responses": ["Essayez de brancher le modem directement avec la prise murale
sans filtre et desactiver le wifi du modem puis faire un test de debit directement
avec un ordinateur branche avec un cable reseau"],
         "context": [""]
        },

        {"tag": "slow connection",
        "patterns": [ "slow connection", "drop in flow"],
        "responses": ["Try to connect the modem directly with the wall socket without
filter and deactivate the wifi of the modem then do a speed test directly with a
computer connected with a network cable" ],
        "context": [""]
        },

        {"tag": "derangement Internet",
         "patterns": ["pb internet", "Internet coupe", "Derangement internet", "pas de
connexion", "perturbation internet","internet"],
         "responses": ["veuillez verifier votre voyant internet du Modem s'il est
allumé en vert ou en rouge sinon veuillez verifier votre voyant adsl du modem s'il est
fixe ou clignotant"],
         "context": [""]
        },

        {"tag": "internet disruption",
         "patterns": ["perturbation internet", "Internet cut", "Internet disturbance",
"no connection", "internet disturbance"],
         "responses": ["please check your modem internet light if it is green or red
if not please check your modem adsl light if it is steady or flashing"],
         "context": [""]
        },

        {"tag": "voyant intrenet rouge",
         "patterns": ["rouge", "le voyant internet du modem est rouge" ],
         "responses": ["veuillez recharger votre compte ADSL ou configuer votre modem
, pour cela veuillez vous rapprocher de la plus proche de nos agence pour vous prendre
en charge"],
         "context": [""]
        },

        {"tag": "red internet light",
        "patterns": ["red", "the internet light on the modem is red" ],
        "responses": ["please recharge your ADSL account or configure your modem, for
this please contact the nearest of our agencies to take care of you"],
        "context": [""]
        },

        {"tag": "voyant intrenet vert",
         "patterns": ["vert", "le voyant internet du modem est vert" ],
         "responses": ["Donc vous n'avez pas de probleme avec algerie telecom,
veuillez verifier votre mot de passe wifi ou votre cable reseau"],
         "context": [""]
```

```
 {"tag": "green internet light",
         "patterns": ["green", "the internet light on the modem is green" ],
         "responses": ["So you have no problem with algerie telecom, please check
your wifi password or your network cable"],
         "context": [""]
        },

        {"tag": "vouyant adsl fixe",
         "patterns": ["fixe", "le voyant adsl du modem est fixe","il est fixe"],
         "responses": ["veuillez recharger votre compte ADSL ou configuer votre
modem , pour cela veuillez vous rapprocher de la plus proche de nos agence pour
vous prendre en charge"],
         "context": [""]
        },

        {"tag": "fixed adsl light",
         "patterns": ["fixed", "the adsl light on the modem is fixed"],
         "responses": ["please recharge your ADSL account or configure your modem,
for this please contact the nearest of our agencies to take care of you"],
         "context": [""]
        },

        {"tag": "voyant adsl clignotant",
         "patterns": ["clignotant", "le voyant adsl du modem est clignotant"],
         "responses": ["donnez moi votre numero de telphone je vais vous envoyé
une equipe d'intervention durant les deux jours qui suis pour regler votre
probleme d'internet"],
         "context": [""]
        },

        {"tag": "blinking adsl light",
        "patterns": ["turn signal", "the modem's adsl light is flashing"],
        "responses": ["give me your phone number I will send you an intervention
team during the two days that follow to solve your line problem"],
        "context": [""]
       }
   ]
}
```

- **Json to RDF :**

Pour créer un graphe RDF à partir d'un fichier JSON contenant des intentions. Le programme suit les étapes suivantes :

1. Importation des bibliothèques nécessaires :

  - `json` pour la manipulation des fichiers JSON.

- `rdflib` pour la création et la manipulation de graphes RDF.

- `URIRef`, `Literal`, et `Namespace` de `rdflib` pour créer des URI, des littéraux et des espaces de noms RDF.

2. Chargement du fichier `intents.json` contenant les intentions dans une variable `intents`.

3. Création d'un graphe RDF vide avec `g = Graph()`.

4. Définition des préfixes RDF à utiliser dans le graphe avec les espaces de noms RDF : `rdf`, `rdfs`, `owl`, et `codeway`.

5. Ajout des préfixes RDF au graphe avec `g.bind()`.

6. Parcours de chaque intention dans la liste des intentions :

- Création d'un nœud URI unique pour l'intention en utilisant `quote(intent['tag'])` comme identifiant.

- Ajout du type d'intention au graphe avec la relation `rdf:type`.

- Parcours de chaque motif dans l'intention :

- Création d'un nœud URI unique pour le motif en utilisant `quote(pattern)` comme identifiant.

- Ajout du type de motif au graphe.

- Ajout de la relation entre l'intention et le motif avec `codeway:hasPattern`.

- Parcours de chaque réponse dans l'intention :

- Création d'un nœud URI unique pour la réponse en utilisant `quote(response)` comme identifiant.

- Ajout du type de réponse au graphe.

- Ajout de la relation entre l'intention et la réponse avec `codeway:hasResponse`.

7. Enregistrement du graphe RDF dans deux fichiers différents :

- `intents.ttl` au format Turtle.

- `intents.rdf` au format XML.

Le programme utilise le module `quote` de `urllib.parse` pour encoder les identifiants des nœuds URI en remplaçant les caractères spéciaux par leur représentation URL-safe.

Cela génère un graphe RDF à partir des données structurées dans notre fichier JSON, en utilisant des nœuds URI et des relations pour représenter les intentions, les motifs et les réponses.

```python
import json
from rdflib import Graph, URIRef, Literal, Namespace
from urllib.parse import quote

# Charger le fichier intents.json
with open('intents.json') as f:
    intents = json.load(f)

# Créer un graphe RDF vide
g = Graph()

# Définir les préfixes RDF
rdf = Namespace('http://www.w3.org/1999/02/22-rdf-syntax-ns#')
rdfs = Namespace('http://www.w3.org/2000/01/rdf-schema#')
owl = Namespace('http://www.w3.org/2002/07/owl#')
codeway = Namespace('http://www.codeway.com/')

# Ajouter les préfixes RDF au graphe
g.bind('rdf', rdf)
g.bind('rdfs', rdfs)
g.bind('owl', owl)
g.bind('codeway', codeway)

# Pour chaque intention dans le fichier intents.json
for intent in intents["intents"]:

    # Créer un nœud URI unique pour cette intention
    intent_uri = codeway[quote(intent['tag'])]

    # Ajouter le type d'intention au graphe
    g.add((intent_uri, rdf.type, codeway.Intent))

    # Pour chaque motif dans l'intention
    for pattern in intent['patterns']:

        # Créer un nœud URI unique pour ce motif
        pattern_uri = codeway[quote(pattern)]

        # Ajouter le type de motif au graphe
        g.add((pattern_uri, rdf.type, codeway.Pattern))

        # Ajouter la relation entre l'intention et le motif
        g.add((intent_uri, codeway.hasPattern, pattern_uri))

    # Pour chaque réponse dans l'intention
    for response in intent['responses']:
```

```
 # Créer un nœud URI unique pour cette réponse
        response_uri = codeway[quote(response)]

        # Ajouter le type de réponse au graphe
        g.add((response_uri, rdf.type, codeway.Response))

        # Ajouter la relation entre l'intention et la réponse
        g.add((intent_uri, codeway.hasResponse, response_uri))

# Enregistrer le graphe RDF dans un fichier turtle
g.serialize(destination='intents.ttl', format='turtle')
g.serialize(destination='intents.rdf', format='xml')
```

- **RDF_to_Json**

Ce code réalise les étapes inverses du précédent. Il extrait les données du graphe RDF et les enregistre dans un fichier JSON.

Voici comment le code fonctionne :

1. Importation des bibliothèques :

- `json` : Cette bibliothèque permet de manipuler des fichiers JSON.

- `rdflib` : C'est une bibliothèque Python pour la manipulation de graphes RDF.

- `urllib.parse` : Cette bibliothèque fournit des fonctions pour manipuler les URL.

2. Chargement du fichier RDF :

Le code utilise la fonction `parse()` de `rdflib.Graph` pour charger le contenu du fichier RDF `intents.rdf` dans le graphe `g`. Le format `"xml"` est spécifié pour indiquer que le fichier est au format XML RDF.

3. Définition des préfixes RDF :

Ce code définisse les mêmes préfixes RDF que dans le code précédent.

4. Chargement des préfixes RDF dans le graphe :

Les lignes `g.bind()` ajoutent les préfixes RDF définis précédemment au graphe `g`, afin qu'ils puissent être utilisés pour lire les triplets RDF.

5. Définition de l'URI de la classe Intent :

La variable `intent_class_uri` est définie pour représenter l'URI de la classe `codeway.Intent`.

6. Récupération des intentions du graphe :

Le code utilise une boucle `for` pour parcourir tous les sujets du graphe qui ont comme prédicat le type `codeway.Intent`. Cela correspond aux nœuds représentant les intentions dans le graphe RDF.

- Création d'un dictionnaire pour chaque intention :

Pour chaque intention, un dictionnaire est créé avec la clé `"tags"` contenant l'identifiant de l'intention (décodé en utilisant `unquote()` et en supprimant le préfixe `codeway`), et les listes `"patterns"` et `"responses"` initialisées vides.

- Récupération des motifs associés à l'intention :

Le code utilise une boucle `for` pour parcourir tous les objets du graphe qui ont comme sujet l'intention en cours et comme prédicat `codeway.hasPattern`. Cela correspond aux motifs associés à l'intention.

- Récupération des réponses associées à l'intention :

Le code utilise une boucle `for` pour parcourir tous les objets du graphe qui ont comme sujet l'intention en cours et comme prédicat `codeway.hasResponse`. Cela correspond aux réponses associées à l'intention.

- Ajout de l'intention au dictionnaire des intentions :

Une fois les motifs et les réponses récupérés, le dictionnaire de l'intention est ajouté à la liste `intents`.

7. Création d'un dictionnaire contenant les intentions :

Un dictionnaire est créé avec la clé `"intents"` contenant la liste des intentions récupérées.

8. Enregistrement des intentions dans un fichier JSON :

Le code utilise la fonction `json.dump()` pour enregistrer le dictionnaire des intentions dans le fichier `intents.json`, en utilisant une indentation de 4 espaces pour une meilleure lisibilité.

Cela permet de convertir les données RDF extraites du fichier `intents.rdf` en un fichier JSON `intents.json` contenant les mêmes informations structurées.

```
import json
from rdflib import Graph, Namespace, URIRef
from urllib.parse import unquote

# Charger le fichier RDF
g = Graph()
g.parse("intents.rdf", format="xml")

# Définir les préfixes RDF
rdf = Namespace('http://www.w3.org/1999/02/22-rdf-syntax-ns#')
rdfs = Namespace('http://www.w3.org/2000/01/rdf-schema#')
owl = Namespace('http://www.w3.org/2002/07/owl#')
codeway = Namespace('http://www.codeway.com/')

# Charger les préfixes RDF dans le graphe
g.bind('rdf', rdf)
g.bind('rdfs', rdfs)
g.bind('owl', owl)
g.bind('codeway', codeway)

# Définir l'URI de la classe Intent
intent_class_uri = codeway.Intent

# Récupérer toutes les intentions du graphe
intents = []

for intent_uri in g.subjects(predicate=rdf.type, object=intent_class_uri):
    intent = {"tags": unquote(intent_uri.replace(str(codeway), "")), "patterns":
[], "responses": []}

    # Récupérer les motifs associés à l'intention
    for pattern_uri in g.objects(subject=intent_uri,
predicate=codeway.hasPattern):
        pattern = unquote(pattern_uri.replace(str(codeway), ""))
        intent["patterns"].append(pattern)

    # Récupérer les réponses associées à l'intention
    for response_uri in g.objects(subject=intent_uri,
predicate=codeway.hasResponse):
        response = unquote(response_uri.replace(str(codeway), ""))
        intent["responses"].append(response)

    intents.append(intent)

# Créer un dictionnaire contenant les intentions
data = {"intents": intents}

# Enregistrer les intentions dans un fichier intents.json
with open("intents.json", "w") as file:
    json.dump(data, file, indent=4)
```

- **RDF_to_Owl :**

Ce programme Python utilise la bibliothèque RDFLib pour convertir un fichier RDF en un fichier OWL. Voici une explication étape par étape du programme :

1. Importation des modules nécessaires :

- `Graph` de RDFLib pour représenter le graphe RDF et OWL.

- `Namespace` de RDFLib pour définir les préfixes RDF et OWL.

- `RDF` et `OWL` de RDFLib pour accéder aux constantes RDF et OWL.

2. Chargement du fichier RDF :

- La fonction `parse` est utilisée pour charger le fichier RDF dans le graphe `g`. Vous devez spécifier le nom du fichier RDF et le format du fichier (dans cet exemple, "xml").

3. Création d'un nouveau graphe OWL :

- La variable `g_owl` est créée comme un nouveau graphe vide pour contenir les triplets OWL.

4. Définition des préfixes RDF et OWL :

- Les préfixes RDF et OWL sont définis en utilisant la fonction `bind` sur le graphe `g_owl`. Cela permet d'associer les préfixes aux espaces de noms correspondants.

5. Copie des triplets RDF dans le graphe OWL :

- Une boucle parcourt tous les triplets du graphe RDF `g` et les ajoute au graphe OWL `g_owl` en utilisant la fonction `add`.

6. Ajout des assertions de type OWL.Class :

- Une boucle parcourt toutes les ressources ayant le prédicat RDF.type et ajoute une assertion de type OWL.Class correspondante dans le graphe OWL.

7. Enregistrement du graphe OWL dans un fichier OWL :

- La fonction `serialize` est utilisée pour enregistrer le graphe OWL `g_owl` dans un fichier OWL. Vous devez spécifier le nom du fichier de sortie et le format de sortie souhaité (dans cet exemple, "xml").

```python
from rdflib import Graph, Namespace, RDF, OWL

# Charger le fichier RDF
g = Graph()
g.parse("intents.rdf", format="xml")

# Créer un nouveau graphe OWL
g_owl = Graph()

# Définir les préfixes RDF et OWL
rdf = Namespace('http://www.w3.org/1999/02/22-rdf-syntax-ns#')
owl = Namespace('http://www.w3.org/2002/07/owl#')
g_owl.bind('rdf', rdf)
g_owl.bind('owl', owl)

# Copier les triplets RDF dans le graphe OWL
for subj, pred, obj in g:
    g_owl.add((subj, pred, obj))

# Ajouter les assertions de type owl:Class pour les ressources de type RDF
for subj in g.subjects(RDF.type, None):
    g_owl.add((subj, RDF.type, OWL.Class))

# Enregistrer le graphe OWL dans un fichier OWL
g_owl.serialize(destination="intents.owl", format="xml")
```

- **Owl_to_RDF :**

Ce programme utilise la bibliothèque RDFLib pour charger un fichier OWL, puis le convertir en différents formats RDF tels que RDF/XML, Turtle et N-Triples. Voici comment il fonctionne :

- Tout d'abord, le programme importe la classe `Graph` de la bibliothèque RDFLib.

- Ensuite, un nouvel objet `Graph` est créé, appelé `g`.

- Le fichier OWL est chargé dans le graphe `g` en utilisant la méthode `parse` de `Graph`, en spécifiant le nom du fichier et le format (dans ce cas, XML).

- Ensuite, le graphe `g` est sérialisé et enregistré dans trois fichiers différents : RDF/XML, Turtle et N-Triples.

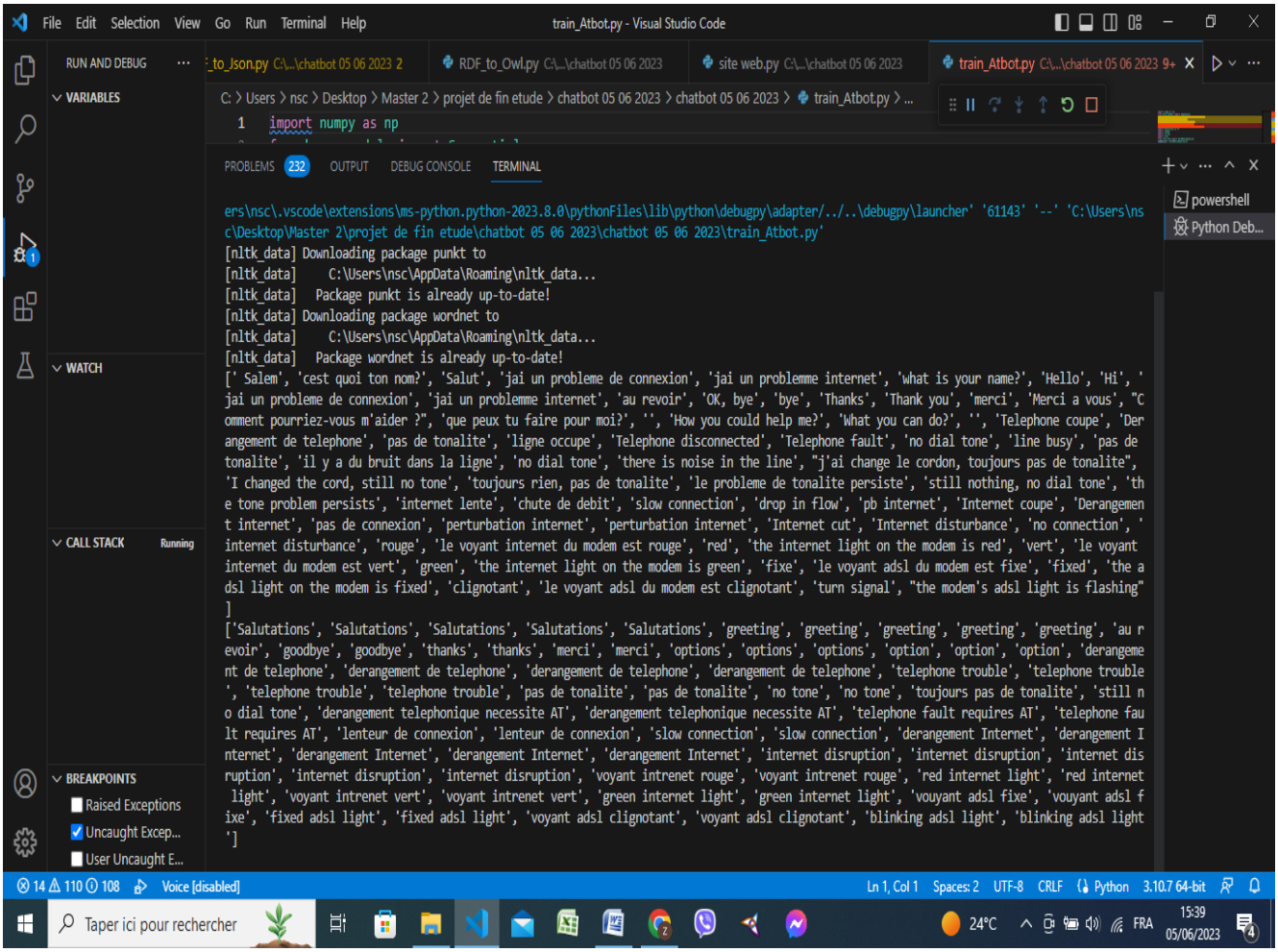
- Pour chaque enregistrement, la méthode `serialize` de `Graph` est utilisée, en spécifiant le format de sérialisation (XML, Turtle ou NT) et le nom du fichier de destination.

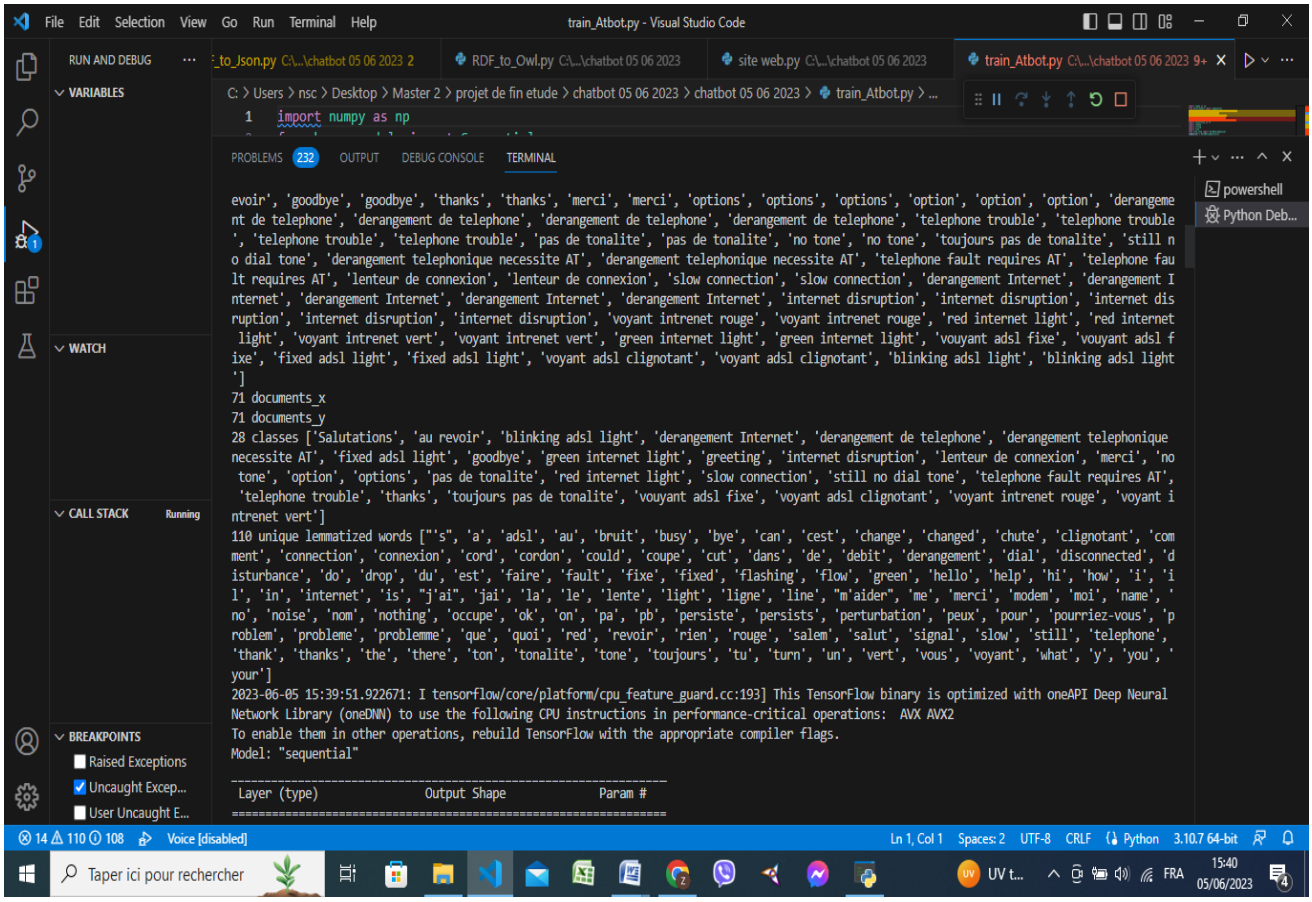
Ainsi, ce programme permet de convertir un fichier OWL en différents formats RDF, ce qui peut être utile pour utiliser ces données avec d'autres outils et bibliothèques compatibles avec RDF.

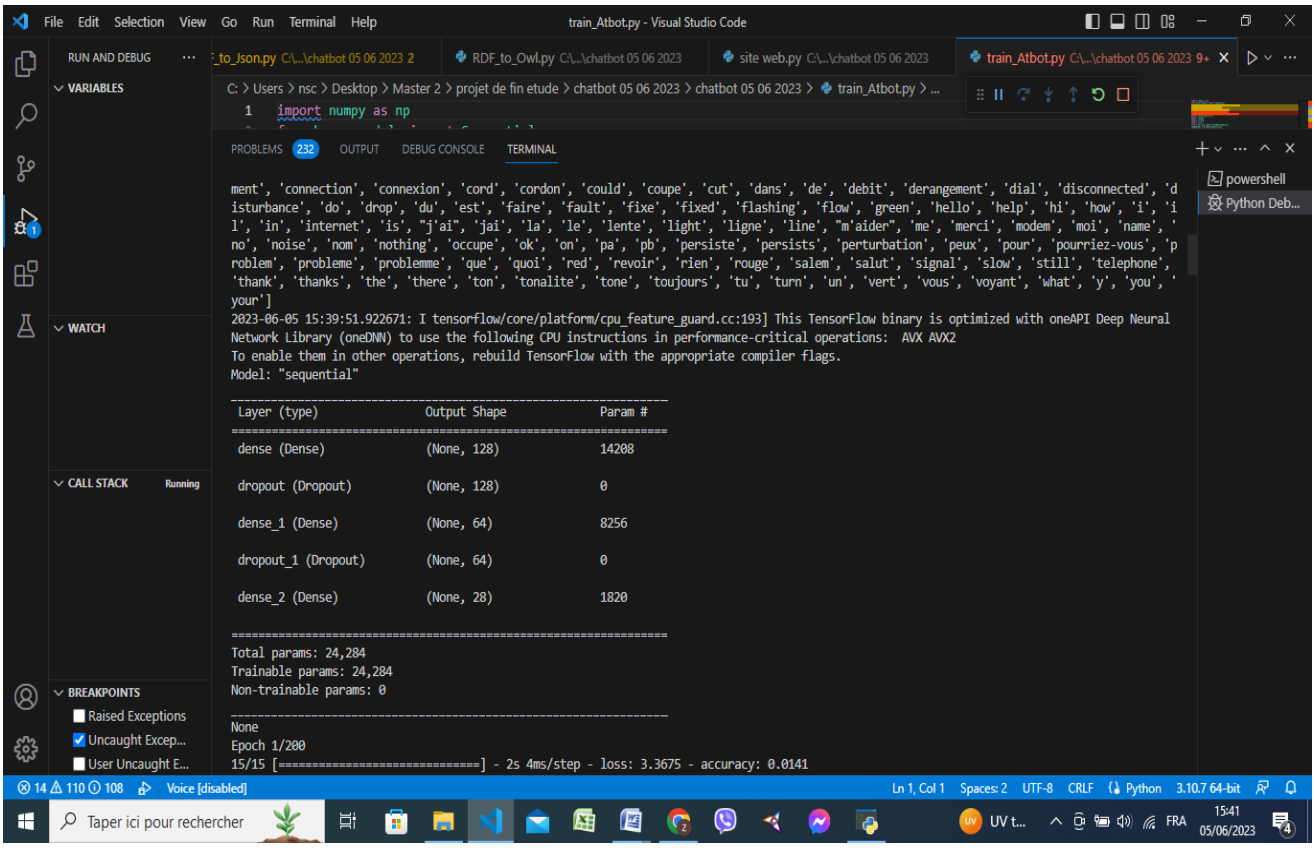
```python
from rdflib import Graph

# Charger le fichier OWL
g = Graph()
g.parse("intents.owl", format="xml")

# Enregistrer le graphe RDF dans un fichier RDF/XML
g.serialize(destination="intents.rdf", format="xml")

# Enregistrer le graphe RDF dans un fichier Turtle
g.serialize(destination="intents.ttl", format="turtle")

# Enregistrer le graphe RDF dans un fichier N-Triples
g.serialize(destination="intents.nt", format="nt")
```

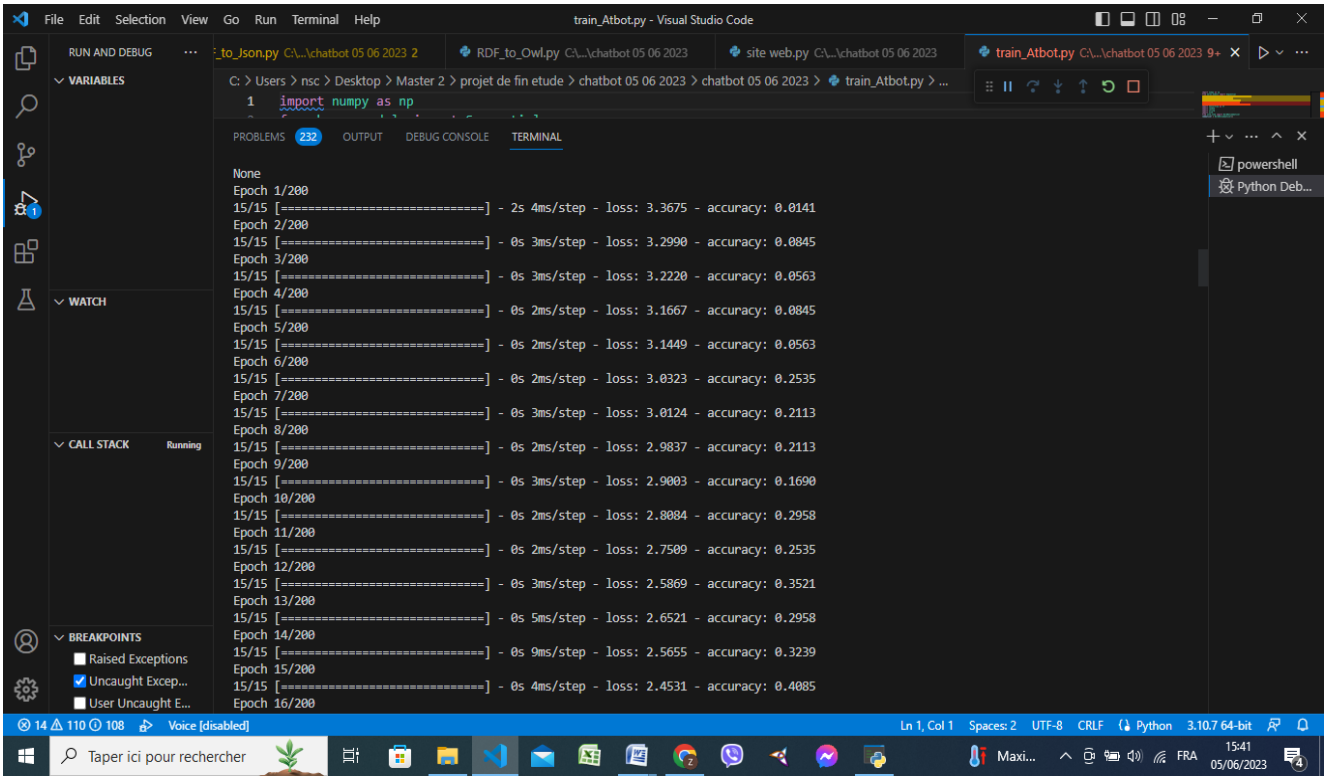
3.5. Démonstration Atbot :

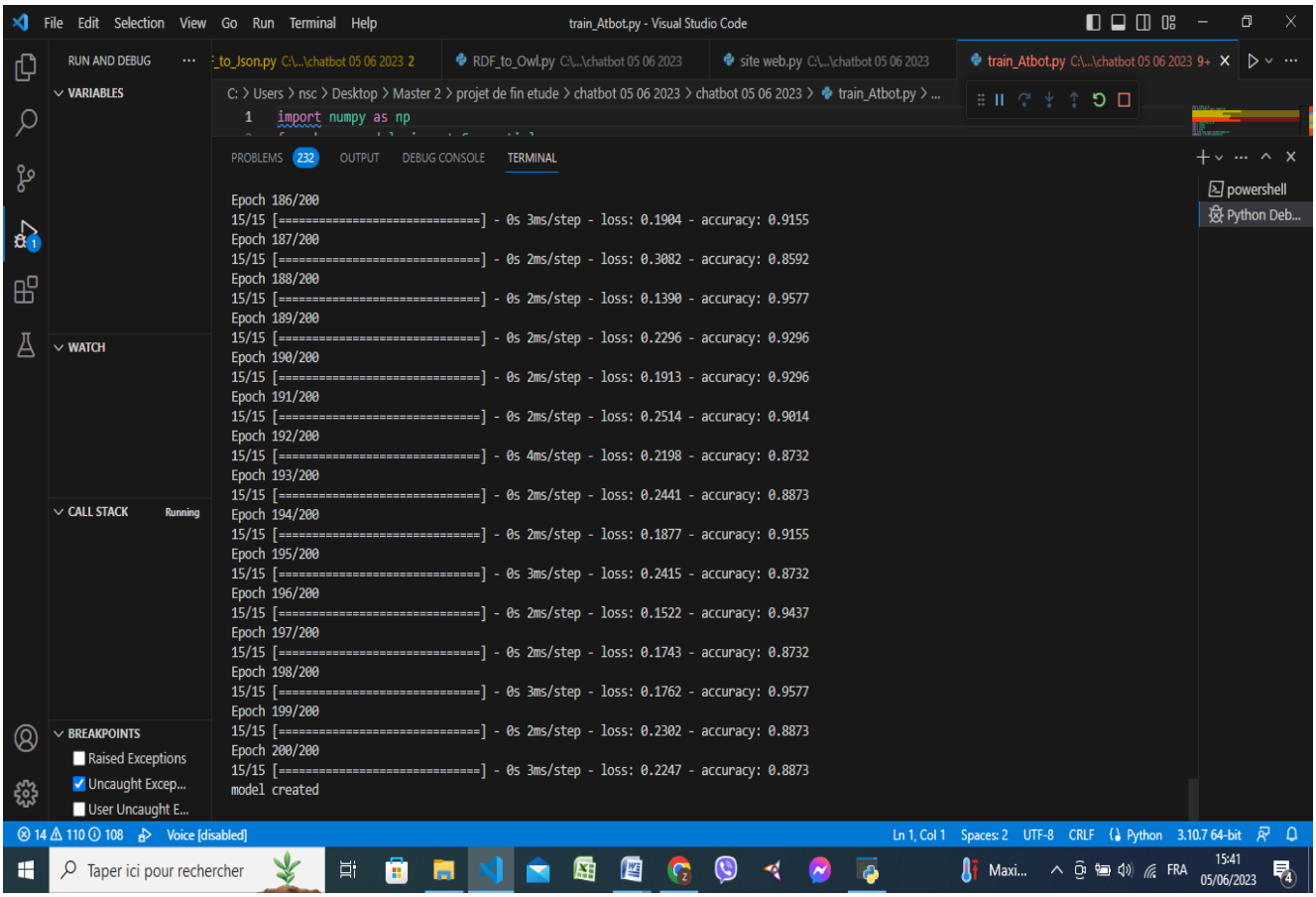
3.5.1 Apprentissage :



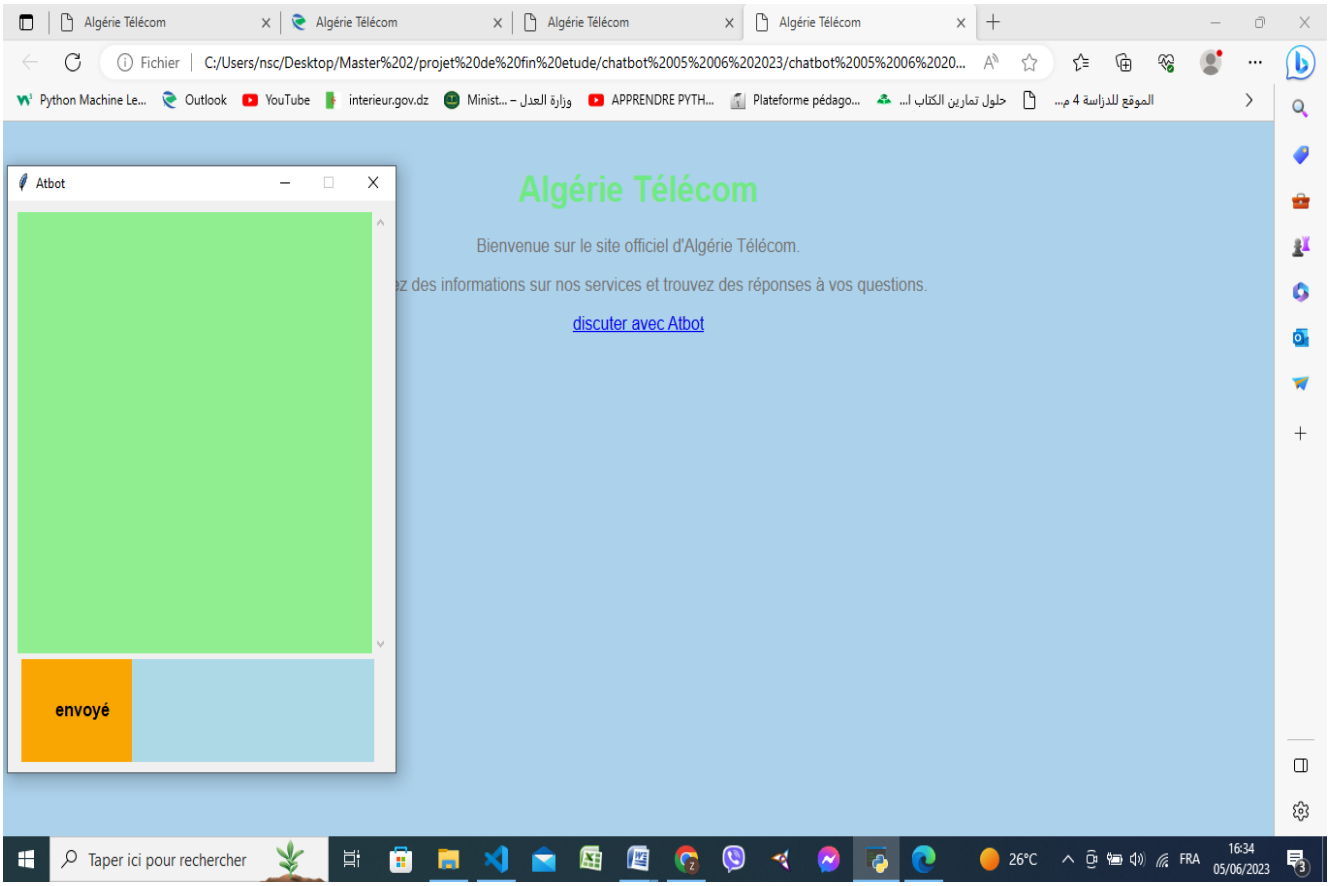

File Edit Selection View Go Run Terminal Help
train_Atbot.py - Visual Studio Code
RUN AND DEBUG
VARIABLES
WATCH
CALL STACK
Running
BREAKPOINTS
Raised Exceptions
Uncaught Excep...
User Uncaught E...
RDF_to_Owl.py C:\...\chatbot 05 06 2023
site web.py C:\...\chatbot 05 06 2023
train_Atbot.py C:\...\chatbot 05 06 2023 9+
C: > Users > nsc > Desktop > Master 2 > projet de fin etude > chatbot 05 06 2023 > chatbot 05 06 2023 > train_Atbot.py > ...
1 import numpy as np
PROBLEMS 232
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
Python Deb...
evoir', 'goodbye', 'goodbye', 'thanks', 'thanks', 'merci', 'merci', 'options', 'options', 'options', 'option', 'option', 'option', 'derangement de telephone', 'derangement de telephone', 'derangement de telephone', 'derangement de telephone', 'telephone trouble', 'telephone trouble', 'telephone trouble', 'telephone trouble', 'pas de tonalite', 'pas de tonalite', 'no tone', 'no tone', 'toujours pas de tonalite', 'still no dial tone', 'derangement telephonique necessite AT', 'derangement telephonique necessite AT', 'telephone fault requires AT', 'telephone fault requires AT', 'lenteur de connexion', 'lenteur de connexion', 'slow connection', 'slow connection', 'derangement Internet', 'derangement Internet', 'derangement Internet', 'derangement Internet', 'derangement Internet', 'internet disruption', 'internet disruption', 'internet disruption', 'internet disruption', 'voyant intrenet rouge', 'voyant intrenet rouge', 'red internet light', 'red internet light', 'voyant intrenet vert', 'voyant intrenet vert', 'green internet light', 'green internet light', 'vouyant adsl fixe', 'vouyant adsl fixe', 'fixed adsl light', 'fixed adsl light', 'voyant adsl clignotant', 'voyant adsl clignotant', 'blinking adsl light', 'blinking adsl light']
71 documents_x
71 documents_y
28 classes ['Salutations', 'au revoir', 'blinking adsl light', 'derangement Internet', 'derangement de telephone', 'derangement telephonique necessite AT', 'fixed adsl light', 'goodbye', 'green internet light', 'greeting', 'internet disruption', 'lenteur de connexion', 'merci', 'no tone', 'option', 'options', 'pas de tonalite', 'red internet light', 'slow connection', 'still no dial tone', 'telephone fault requires AT', 'telephone trouble', 'thanks', 'toujours pas de tonalite', 'vouyant adsl fixe', 'voyant adsl clignotant', 'voyant intrenet rouge', 'voyant intrenet vert']
110 unique lemmatized words ["'s", 'a', 'adsl', 'au', 'bruit', 'busy', 'bye', 'can', 'cest', 'change', 'changed', 'chute', 'clignotant', 'comment', 'connection', 'connexion', 'cord', 'cordon', 'could', 'coupe', 'cut', 'dans', 'de', 'debit', 'derangement', 'dial', 'disconnected', 'disturbance', 'do', 'drop', 'du', 'est', 'faire', 'fault', 'fixe', 'fixed', 'flashing', 'flow', 'green', 'hello', 'help', 'hi', 'how', 'i', 'il', 'in', 'internet', 'is', "j'ai", 'jai', 'la', 'le', 'lente', 'light', 'ligne', 'line', "m'aider", 'me', 'merci', 'modem', 'moi', 'name', 'no', 'noise', 'nom', 'nothing', 'occupe', 'ok', 'on', 'pa', 'pb', 'persiste', 'persists', 'perturbation', 'peux', 'pour', 'pourriez-vous', 'problem', 'probleme', 'problemme', 'que', 'quoi', 'red', 'revoir', 'rien', 'rouge', 'salem', 'salut', 'signal', 'slow', 'still', 'telephone', 'thank', 'thanks', 'the', 'there', 'ton', 'tonalite', 'tone', 'toujours', 'tu', 'turn', 'un', 'vert', 'vous', 'voyant', 'what', 'y', 'you', 'your']
2023-06-05 15:39:51.922671: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Model: "sequential"
Layer (type) Output Shape Param #
Ln 1, Col 1 Spaces: 2 UTF-8 CRLF Python 3.10.7 64-bit
Taper ici pour rechercher
15:40
05/06/2023


File Edit Selection View Go Run Terminal Help
train_Atbot.py - Visual Studio Code
RUN AND DEBUG
VARIABLES
WATCH
CALL STACK
Running
BREAKPOINTS
Raised Exceptions
Uncaught Excep...
User Uncaught E...
RDF_to_Owl.py C:\...\chatbot 05 06 2023
site web.py C:\...\chatbot 05 06 2023
train_Atbot.py C:\...\chatbot 05 06 2023 9+
C: > Users > nsc > Desktop > Master 2 > projet de fin etude > chatbot 05 06 2023 > chatbot 05 06 2023 > train_Atbot.py > ...
1 import numpy as np
PROBLEMS 232
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
Python Deb...
ment', 'connection', 'connexion', 'cord', 'cordon', 'could', 'coupe', 'cut', 'dans', 'de', 'debit', 'derangement', 'dial', 'disconnected', 'disturbance', 'do', 'drop', 'du', 'est', 'faire', 'fault', 'fixe', 'fixed', 'flashing', 'flow', 'green', 'hello', 'help', 'hi', 'how', 'i', 'il', 'in', 'internet', 'is', "j'ai", 'jai', 'la', 'le', 'lente', 'light', 'ligne', 'line', "m'aider", 'me', 'merci', 'modem', 'moi', 'name', 'no', 'noise', 'nom', 'nothing', 'occupe', 'ok', 'on', 'pa', 'pb', 'persiste', 'persists', 'perturbation', 'peux', 'pour', 'pourriez-vous', 'problem', 'probleme', 'problemme', 'que', 'quoi', 'red', 'revoir', 'rien', 'rouge', 'salem', 'salut', 'signal', 'slow', 'still', 'telephone', 'thank', 'thanks', 'the', 'there', 'ton', 'tonalite', 'tone', 'toujours', 'tu', 'turn', 'un', 'vert', 'vous', 'voyant', 'what', 'y', 'you', 'your']
2023-06-05 15:39:51.922671: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Model: "sequential"
Layer (type) Output Shape Param #
dense (Dense) (None, 128) 14208
dropout (Dropout) (None, 128) 0
dense_1 (Dense) (None, 64) 8256
dropout_1 (Dropout) (None, 64) 0
dense_2 (Dense) (None, 28) 1820
Total params: 24,284
Trainable params: 24,284
Non-trainable params: 0
None
Epoch 1/200
15/15 [==========] - 2s 4ms/step - loss: 3.3675 - accuracy: 0.0141
Ln 1, Col 1 Spaces: 2 UTF-8 CRLF Python 3.10.7 64-bit
Taper ici pour rechercher
15:41
05/06/2023

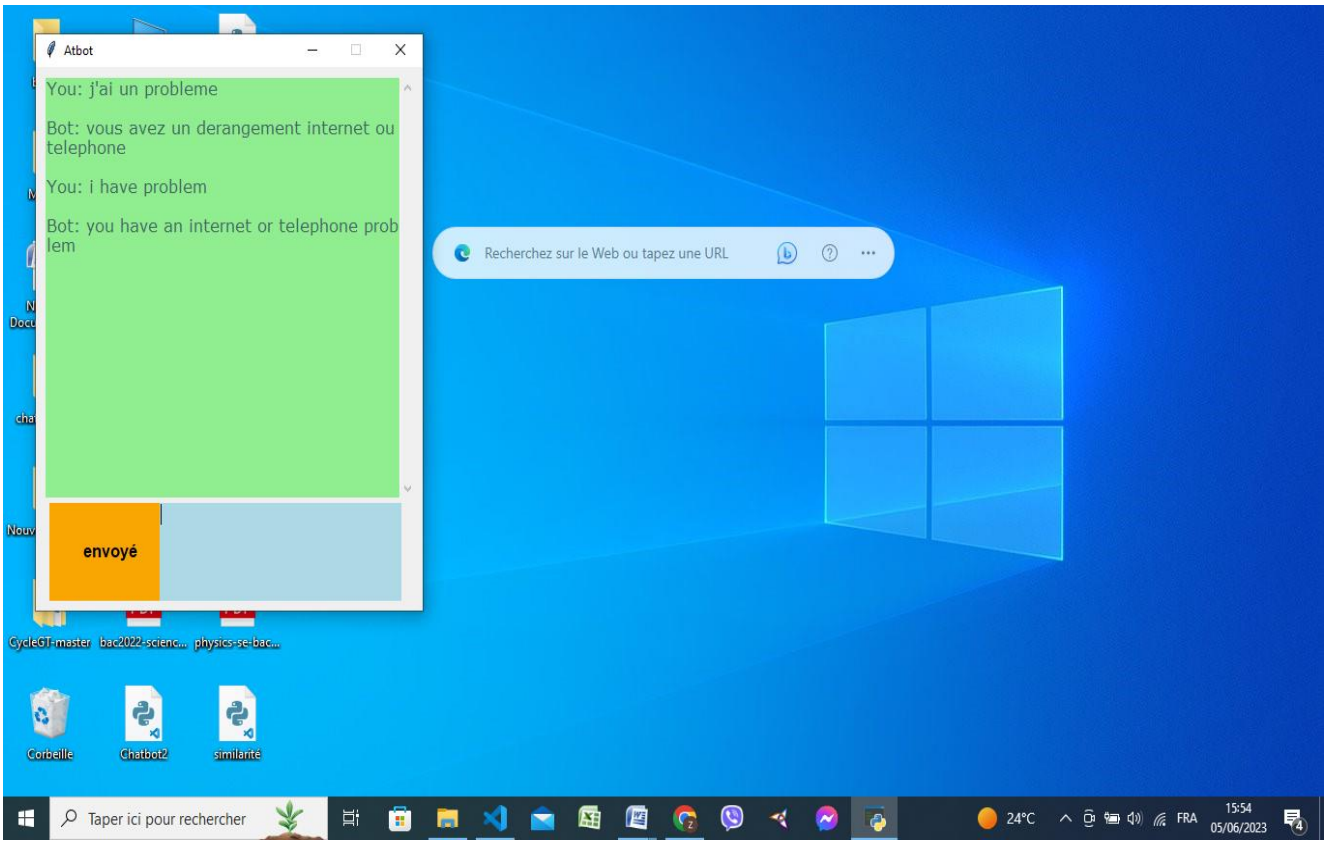

File Edit Selection View Go Run Terminal Help
train_Atbot.py - Visual Studio Code
RUN AND DEBUG
VARIABLES
WATCH
CALL STACK
Running
BREAKPOINTS
Raised Exceptions
Uncaught Excep...
User Uncaught E...
to_Json.py C:\...\chatbot 05 06 2023 2
RDF_to_Owl.py C:\...\chatbot 05 06 2023
site web.py C:\...\chatbot 05 06 2023
train_Atbot.py C:\...\chatbot 05 06 2023 9+
C: > Users > nsc > Desktop > Master 2 > projet de fin etude > chatbot 05 06 2023 > chatbot 05 06 2023 > train_Atbot.py > ...
1 import numpy as np
PROBLEMS 232
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
Python Deb...
None
Epoch 1/200
15/15 [==============================] - 2s 4ms/step - loss: 3.3675 - accuracy: 0.0141
Epoch 2/200
15/15 [==============================] - 0s 3ms/step - loss: 3.2990 - accuracy: 0.0845
Epoch 3/200
15/15 [==============================] - 0s 3ms/step - loss: 3.2220 - accuracy: 0.0563
Epoch 4/200
15/15 [==============================] - 0s 2ms/step - loss: 3.1667 - accuracy: 0.0845
Epoch 5/200
15/15 [==============================] - 0s 2ms/step - loss: 3.1449 - accuracy: 0.0563
Epoch 6/200
15/15 [==============================] - 0s 2ms/step - loss: 3.0323 - accuracy: 0.2535
Epoch 7/200
15/15 [==============================] - 0s 3ms/step - loss: 3.0124 - accuracy: 0.2113
Epoch 8/200
15/15 [==============================] - 0s 2ms/step - loss: 2.9837 - accuracy: 0.2113
Epoch 9/200
15/15 [==============================] - 0s 3ms/step - loss: 2.9003 - accuracy: 0.1690
Epoch 10/200
15/15 [==============================] - 0s 2ms/step - loss: 2.8084 - accuracy: 0.2958
Epoch 11/200
15/15 [==============================] - 0s 2ms/step - loss: 2.7509 - accuracy: 0.2535
Epoch 12/200
15/15 [==============================] - 0s 3ms/step - loss: 2.5869 - accuracy: 0.3521
Epoch 13/200
15/15 [==============================] - 0s 5ms/step - loss: 2.6521 - accuracy: 0.2958
Epoch 14/200
15/15 [==============================] - 0s 9ms/step - loss: 2.5655 - accuracy: 0.3239
Epoch 15/200
15/15 [==============================] - 0s 4ms/step - loss: 2.4531 - accuracy: 0.4085
Epoch 16/200
14 110 108
Voice [disabled]
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
Python
3.10.7 64-bit
Taper ici pour rechercher
Maxi...
FRA
15:41
05/06/2023

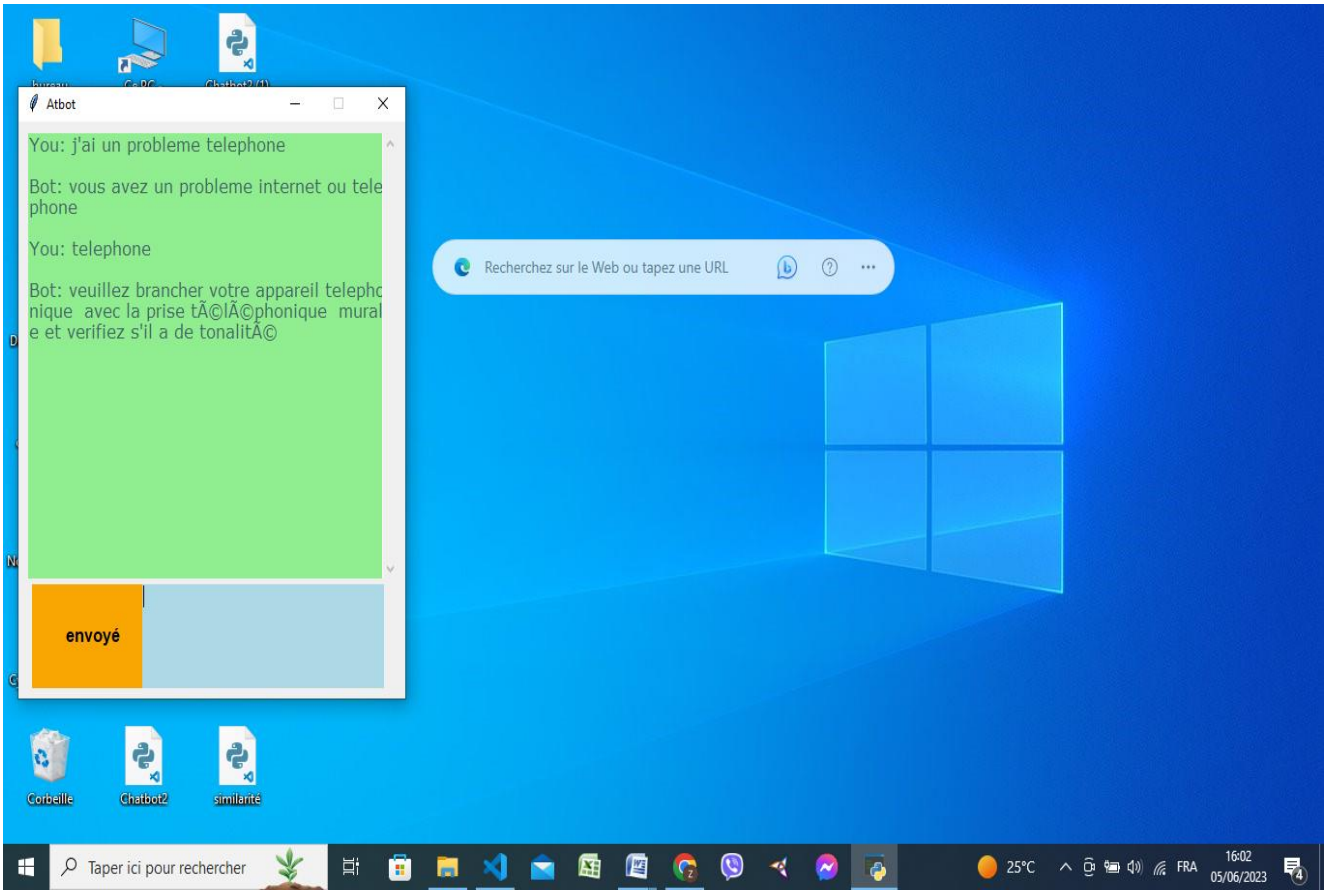

File Edit Selection View Go Run Terminal Help
train_Atbot.py - Visual Studio Code
RUN AND DEBUG
VARIABLES
WATCH
CALL STACK
Running
BREAKPOINTS
Raised Exceptions
Uncaught Excep...
User Uncaught E...
to_Json.py C:\...\chatbot 05 06 2023 2
RDF_to_Owl.py C:\...\chatbot 05 06 2023
site web.py C:\...\chatbot 05 06 2023
train_Atbot.py C:\...\chatbot 05 06 2023 9+
C: > Users > nsc > Desktop > Master 2 > projet de fin etude > chatbot 05 06 2023 > chatbot 05 06 2023 > train_Atbot.py > ...
1 import numpy as np
PROBLEMS 232
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
Python Deb...
Epoch 186/200
15/15 [==============================] - 0s 3ms/step - loss: 0.1904 - accuracy: 0.9155
Epoch 187/200
15/15 [==============================] - 0s 2ms/step - loss: 0.3082 - accuracy: 0.8592
Epoch 188/200
15/15 [==============================] - 0s 2ms/step - loss: 0.1390 - accuracy: 0.9577
Epoch 189/200
15/15 [==============================] - 0s 2ms/step - loss: 0.2296 - accuracy: 0.9296
Epoch 190/200
15/15 [==============================] - 0s 2ms/step - loss: 0.1913 - accuracy: 0.9296
Epoch 191/200
15/15 [==============================] - 0s 2ms/step - loss: 0.2514 - accuracy: 0.9014
Epoch 192/200
15/15 [==============================] - 0s 4ms/step - loss: 0.2198 - accuracy: 0.8732
Epoch 193/200
15/15 [==============================] - 0s 2ms/step - loss: 0.2441 - accuracy: 0.8873
Epoch 194/200
15/15 [==============================] - 0s 2ms/step - loss: 0.1877 - accuracy: 0.9155
Epoch 195/200
15/15 [==============================] - 0s 3ms/step - loss: 0.2415 - accuracy: 0.8732
Epoch 196/200
15/15 [==============================] - 0s 2ms/step - loss: 0.1522 - accuracy: 0.9437
Epoch 197/200
15/15 [==============================] - 0s 2ms/step - loss: 0.1743 - accuracy: 0.8732
Epoch 198/200
15/15 [==============================] - 0s 3ms/step - loss: 0.1762 - accuracy: 0.9577
Epoch 199/200
15/15 [==============================] - 0s 2ms/step - loss: 0.2302 - accuracy: 0.8873
Epoch 200/200
15/15 [==============================] - 0s 3ms/step - loss: 0.2247 - accuracy: 0.8873
model created
14 110 108
Voice [disabled]
Ln 1, Col 1
Spaces: 2
UTF-8
CRLF
Python
3.10.7 64-bit
Taper ici pour rechercher
Maxi...
FRA
15:41
05/06/2023

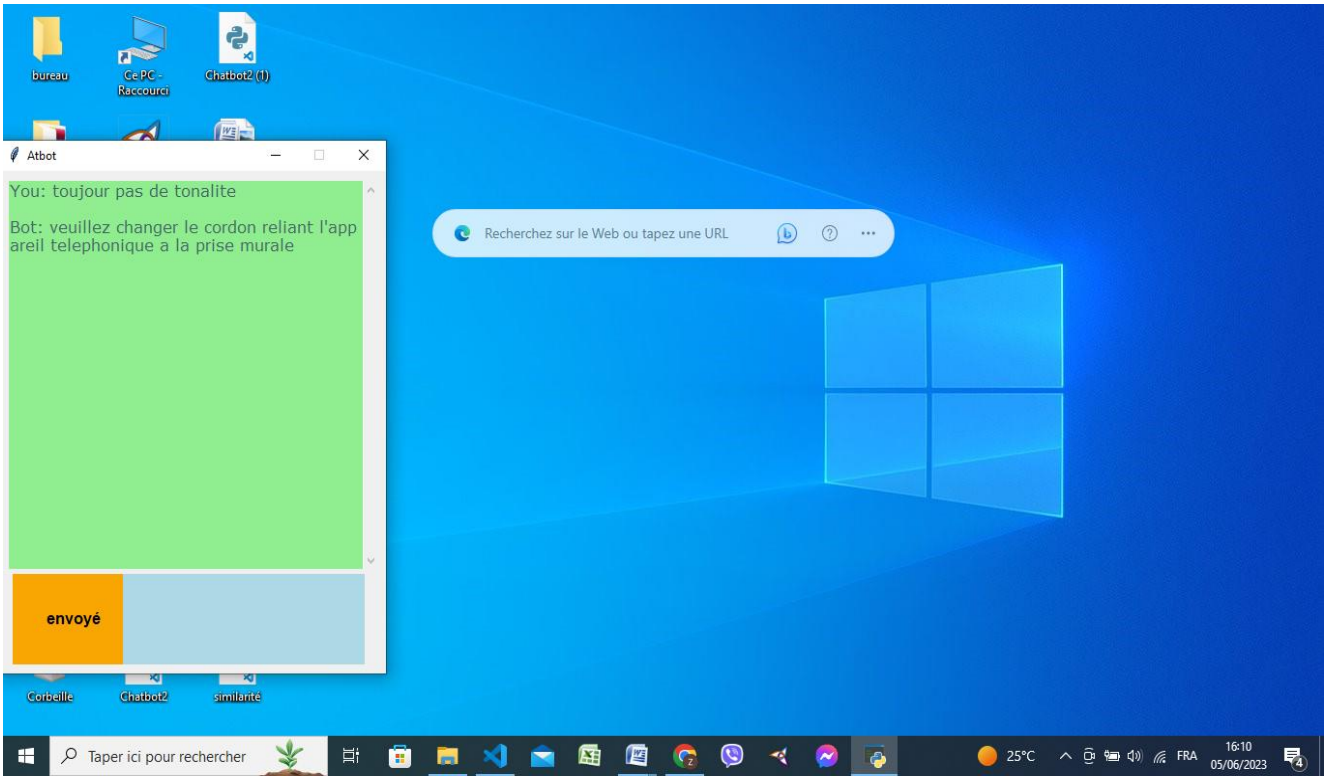
3.5.2. Dialogue avec Atbot :

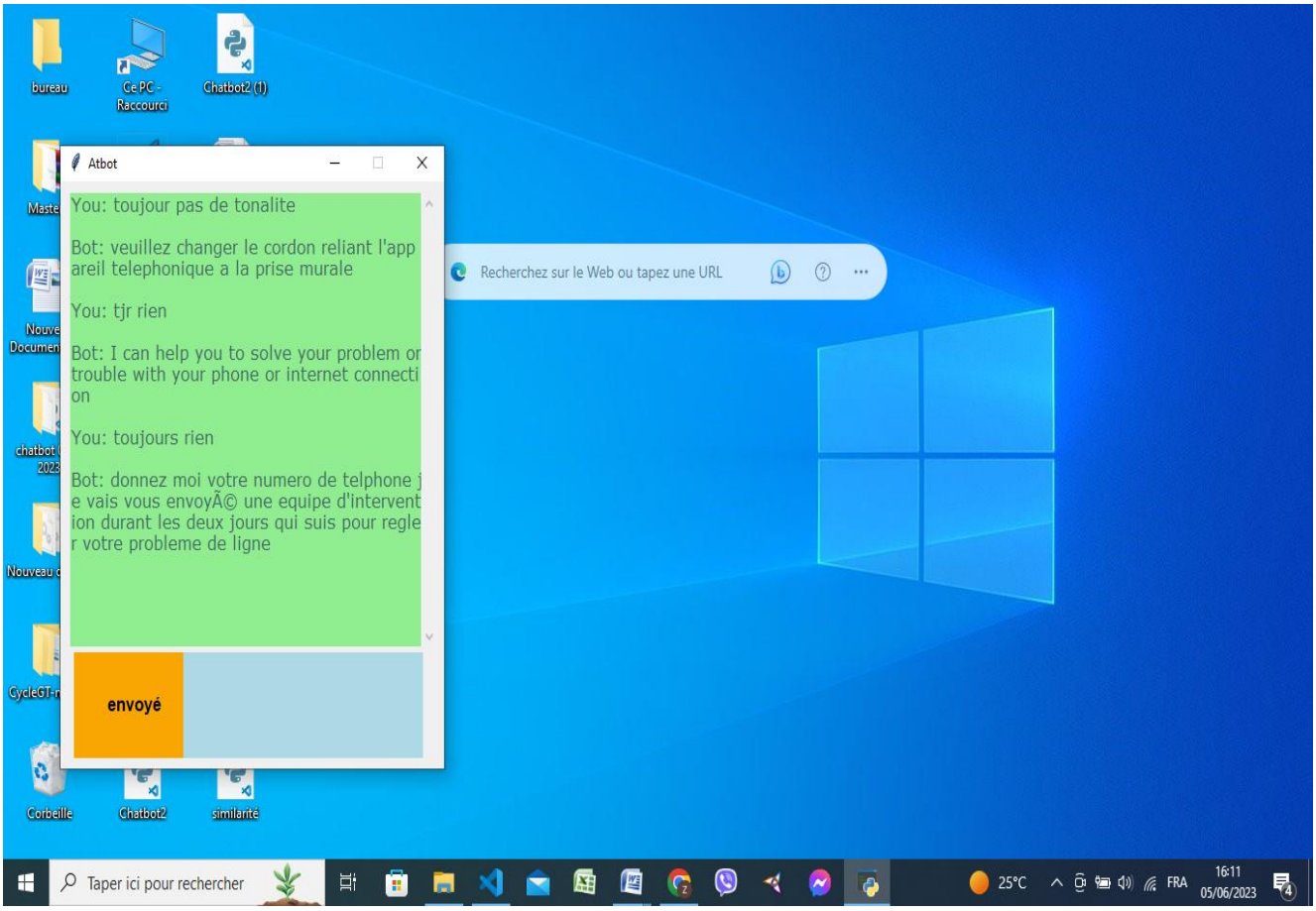


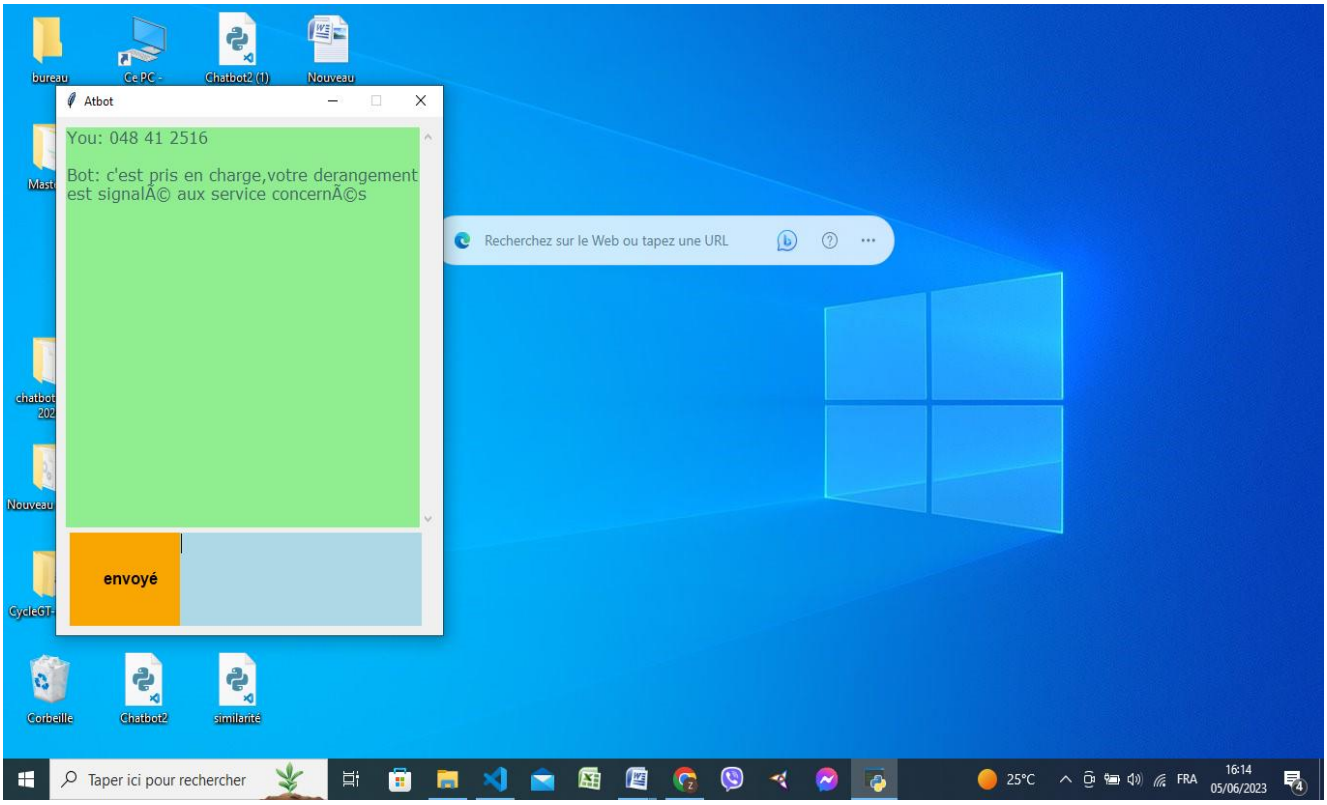

Algérie Télécom
Fichier | C:/Users/nsc/Desktop/Master%202/projet%20de%20fin%20etude/chatbot%2005%2006%202023/chatbot%2005%2006%2020...
Python Machine Le...
Outlook
YouTube
interieur.gov.dz
Minist... – وزارة العدل
APPRENDRE PYTH...
Plateforme pédago...
حلول تمارين الكتاب ا...
الموقع للدراسة 4 م...
Atbot
You: salem
Bot: Bonjour, Je suis ATBot, comment puis j e vous aider aujourd'hui?
You: bonjour
Bot: je peux vous aider a relevÃ© votre pro bleme ou derrangement de telephone ou de connexion internet
You: hello
Bot: Hello, I am ATBot, How can i help you to day?
You: hi
Bot: Hello, I am ATBot, How can i help you to day?
envoyé
Algérie Télécom
nue sur le site officiel d'Algérie Télécom.
sur nos services et trouvez des réponses à vos questions.
discuter avec Atbot
Taper ici pour rechercher
26°C
FRA
16:46
05/06/2023

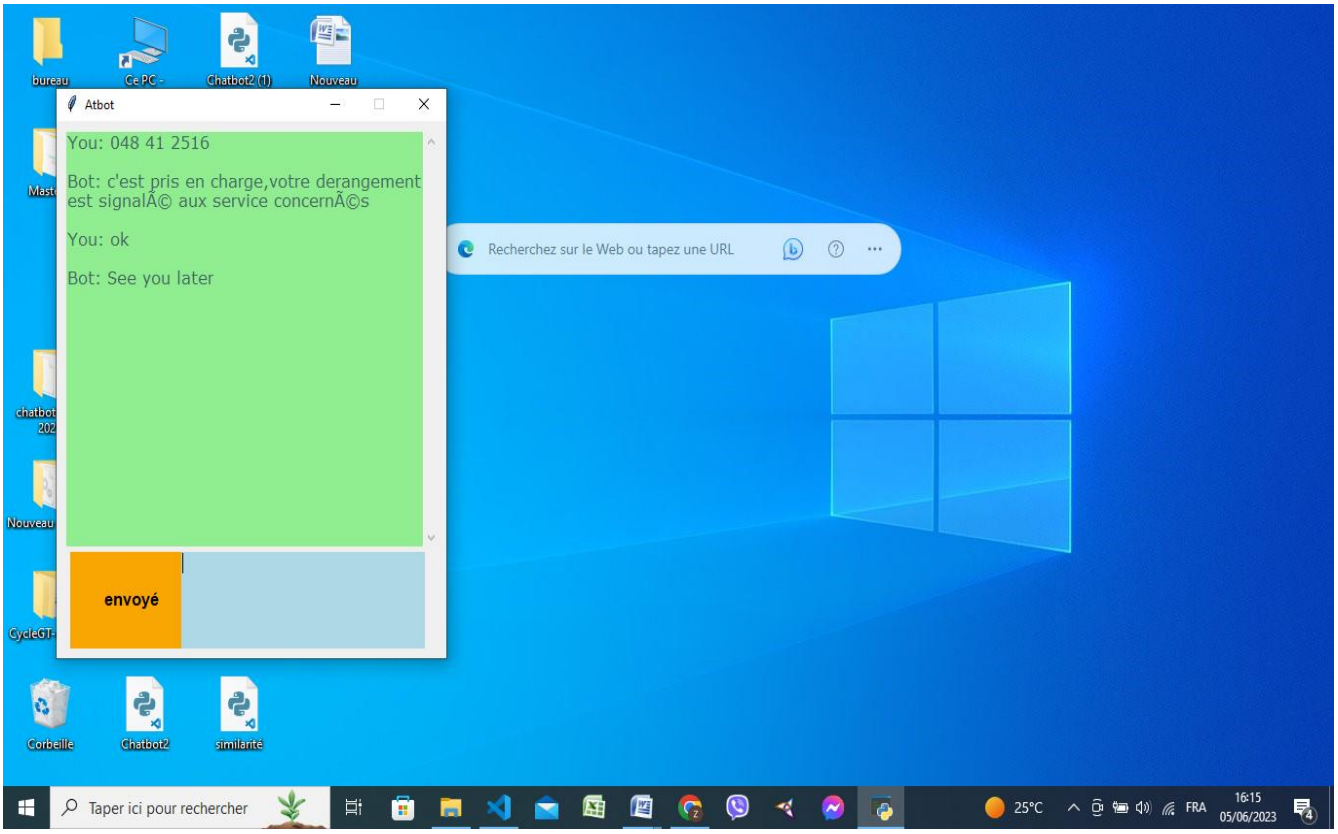

Atbot
You: j'ai un probleme
Bot: vous avez un derangement internet ou telephone
You: i have problem
Bot: you have an internet or telephone prob lem
envoyé
Recherchez sur le Web ou tapez une URL
CycleGT-master
bac2022-scienc...
physics-se-bac...
Corbeille
Chatbot2
similarité
Taper ici pour rechercher
24°C
FRA
15:54
05/06/2023

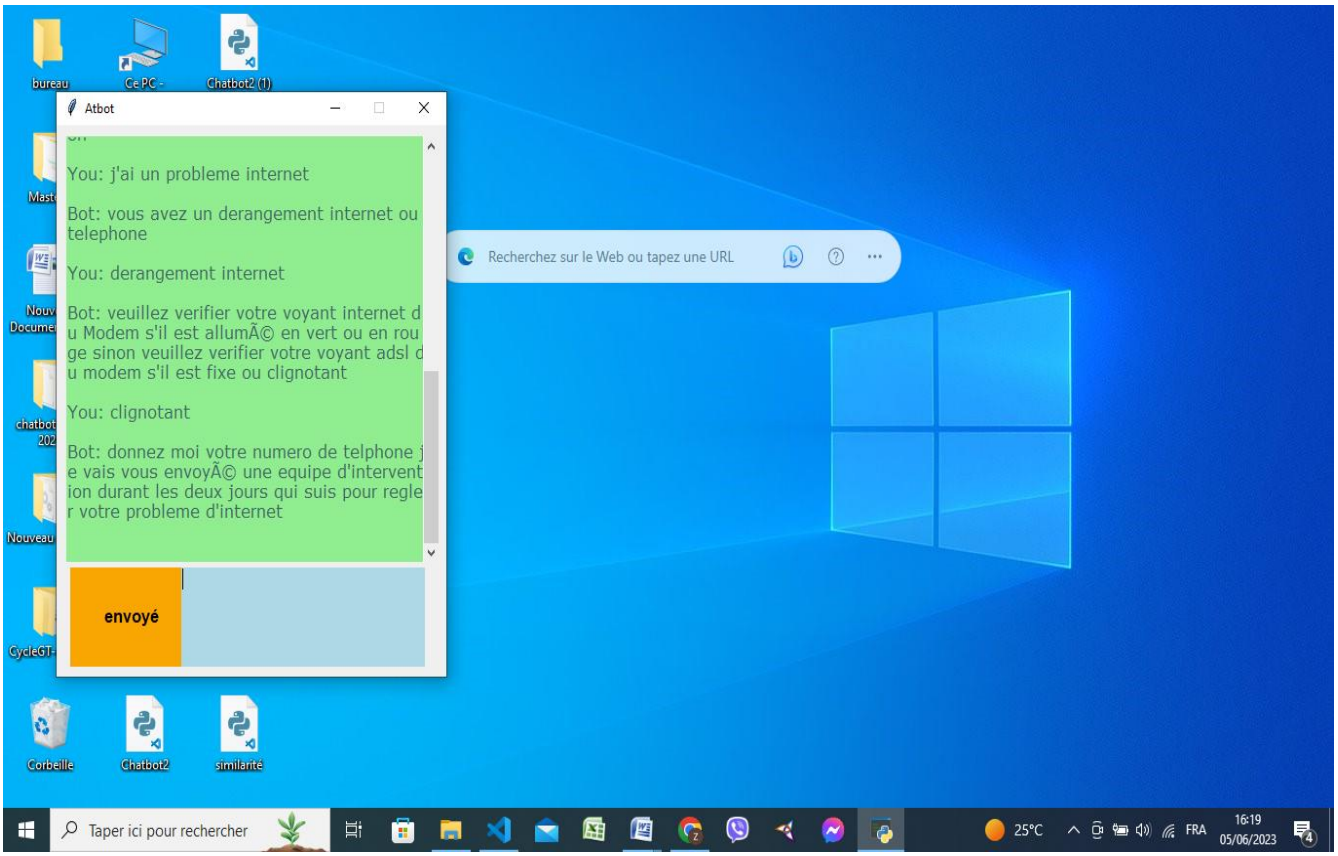

Atbot
You: j'ai un probleme telephone
Bot: vous avez un probleme internet ou tele phone
You: telephone
Bot: veuillez brancher votre appareil telepho nique avec la prise tÃ©lÃ©phonique mural e et verifiez s'il a de tonalitÃ©
envoyé
Recherchez sur le Web ou tapez une URL
Corbeille
Chatbot2
similarité
Taper ici pour rechercher
25°C
FRA
16:02
05/06/2023

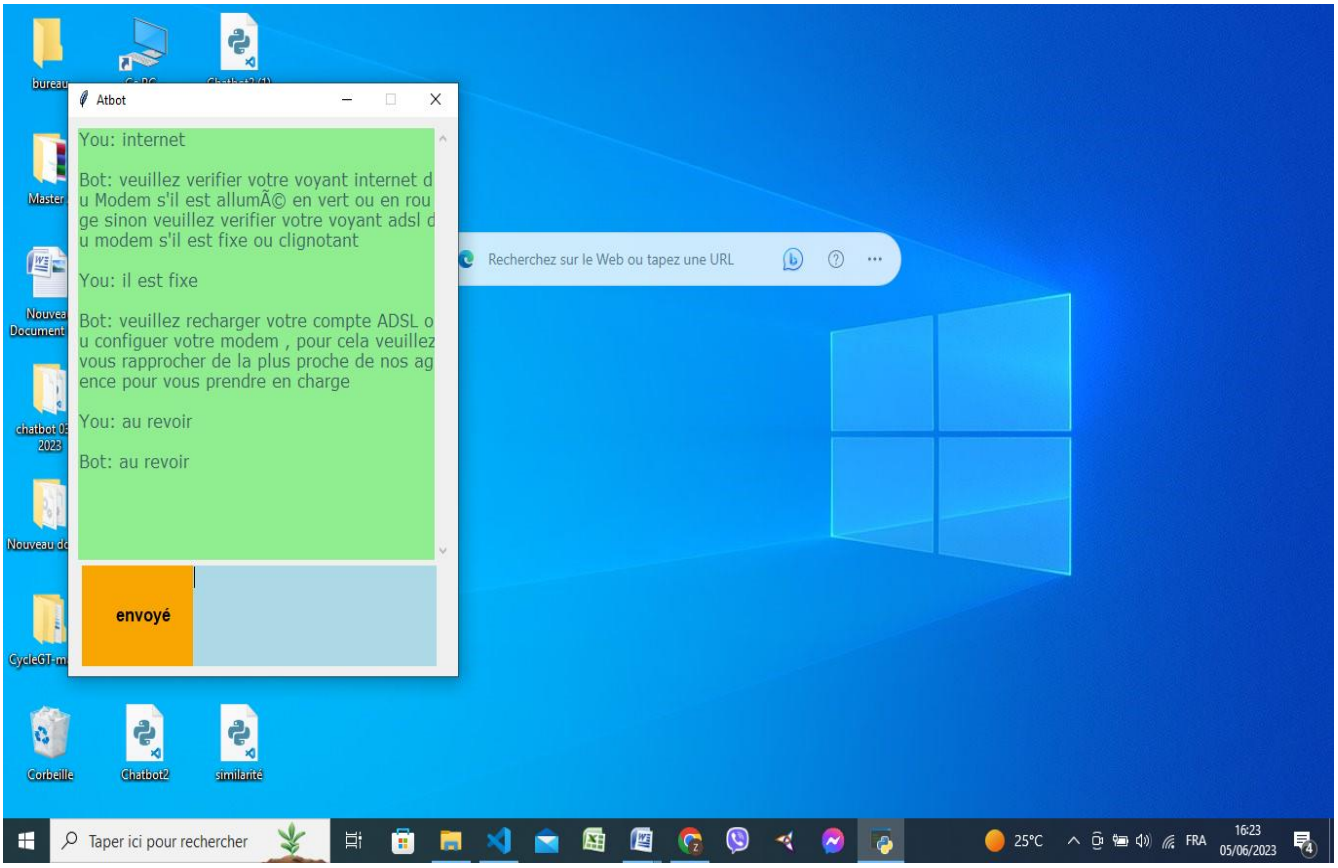

bureau
Ce PC - Raccourci
Chatbot2 (1)
Atbot
You: toujour pas de tonalite
Bot: veuillez changer le cordon reliant l'app areil telephonique a la prise murale
envoyé
Recherchez sur le Web ou tapez une URL
Corbeille
Chatbot2
similarité
Taper ici pour rechercher
25°C
FRA
16:10
05/06/2023


Atbot
You: toujour pas de tonalite
Bot: veuillez changer le cordon reliant l'app areil telephonique a la prise murale
You: tjr rien
Bot: I can help you to solve your problem or trouble with your phone or internet connecti on
You: toujours rien
Bot: donnez moi votre numero de telphone j e vais vous envoyÃ© une equipe d'intervent ion durant les deux jours qui suis pour regle r votre probleme de ligne
envoyé
Recherchez sur le Web ou tapez une URL
Taper ici pour rechercher
25°C
FRA
16:11
05/06/2023


Atbot
You: 048 41 2516
Bot: c'est pris en charge,votre derangement est signalÃ© aux service concernÃ©s
envoyé
Recherchez sur le Web ou tapez une URL
Taper ici pour rechercher
25°C
FRA
16:14
05/06/2023


Atbot
You: 048 41 2516
Bot: c'est pris en charge,votre derangement est signalÃ© aux service concernÃ©s
You: ok
Bot: See you later
envoyé
Recherchez sur le Web ou tapez une URL
Taper ici pour rechercher
25°C
FRA
16:15
05/06/2023


Atbot
You: j'ai un probleme internet
Bot: vous avez un derangement internet ou telephone
You: derangement internet
Bot: veuillez verifier votre voyant internet du Modem s'il est allumÃ© en vert ou en rouge sinon veuillez verifier votre voyant adsl du modem s'il est fixe ou clignotant
You: clignotant
Bot: donnez moi votre numero de telphone je vais vous envoyÃ© une equipe d'intervention durant les deux jours qui suis pour regler votre probleme d'internet
envoyé
Recherchez sur le Web ou tapez une URL
Taper ici pour rechercher
25°C
FRA
16:19
05/06/2023


Atbot
You: internet
Bot: veuillez verifier votre voyant internet d u Modem s'il est allumÃ© en vert ou en rou ge sinon veuillez verifier votre voyant adsl d u modem s'il est fixe ou clignotant
You: il est fixe
Bot: veuillez recharger votre compte ADSL o u configuer votre modem , pour cela veuillez vous rapprocher de la plus proche de nos ag ence pour vous prendre en charge
You: au revoir
Bot: au revoir
envoyé
Recherchez sur le Web ou tapez une URL
Taper ici pour rechercher
25°C
FRA
16:23
05/06/2023
Corbeille
Chatbot2
similarité

Conclusion

Récapitulation des principaux résultats obtenus :

En récapitulant les principaux résultats obtenus, ce mémoire a exploré la réalisation d'un chatbot de relève des dérangements pour Algérie Télécom. L'étude a permis d'identifier les besoins des clients en matière de gestion des dérangements et de mettre en évidence les avantages d'un chatbot dans ce contexte. L'analyse des problèmes de dérangements et l'étude des chatbots appliqués aux télécommunications ont fourni une base solide pour la conception du chatbot. La conception a inclus l'analyse des exigences fonctionnelles et non fonctionnelles, le choix des technologies, la conception des dialogues et l'intégration des outils de traitement automatique du langage naturel.

Apport du mémoire à la recherche et aux applications pratiques :

L'apport de ce mémoire réside dans la proposition d'une solution concrète et applicable pour améliorer le processus de relève des dérangements chez Algérie Télécom. En mettant en œuvre un chatbot, l'entreprise peut offrir à ses clients un moyen plus efficace et pratique de signaler et de résoudre les problèmes techniques. Cela peut entraîner une meilleure satisfaction client, une réduction des délais de traitement et une optimisation des ressources internes.

Analyse des résultats par rapport aux objectifs fixés :

L'analyse des résultats obtenus par rapport aux objectifs fixés démontre l'efficacité du chatbot de relève des dérangements d'Algérie Télécom. Il a permis d'améliorer la réactivité et l'accessibilité du support client, en fournissant une assistance automatisée et personnalisée. Les utilisateurs ont exprimé une satisfaction élevée quant à l'expérience utilisateur et à la résolution rapide des problèmes. De plus, le chatbot a contribué à réduire la charge de travail du personnel de support, leur permettant de se concentrer sur des tâches plus complexes.

Limitations de l'étude et perspectives d'amélioration :

Malgré les résultats positifs, cette étude présente certaines limitations. Tout d'abord, l'analyse se base sur un échantillon limité d'utilisateurs, ce qui peut limiter la généralisation des résultats. De plus, l'efficacité du chatbot peut varier en fonction de la complexité des dérangements rencontrés. Il est donc recommandé de poursuivre les études pour évaluer l'impact du chatbot sur une plus grande échelle et dans différents contextes.

Recommandations pour l'implémentation et le déploiement du chatbot :

Pour l'implémentation et le déploiement du chatbot de relève des dérangements, il est recommandé de mettre en place un processus de formation continu pour améliorer les capacités de compréhension et de réponse du chatbot. De plus, il est essentiel de maintenir une surveillance régulière des interactions afin d'identifier les lacunes et d'apporter des ajustements nécessaires. Enfin, la promotion et la sensibilisation des utilisateurs quant aux avantages et aux fonctionnalités du chatbot sont des recommandations clés pour favoriser son adoption et son utilisation optimale.

En conclusion, le chatbot de relève des dérangements d'Algérie Télécom présente des résultats prometteurs en termes d'efficacité et de satisfaction des utilisateurs. Cependant, des améliorations continues, une évaluation à plus grande échelle et une adaptation aux besoins

changeants des clients sont nécessaires pour optimiser son utilisation et garantir son succès à long terme.

Perspectives futures et ouvertures à d'autres travaux de recherche :

En ce qui concerne les perspectives futures, il serait intéressant d'étendre cette étude en évaluant l'impact réel du chatbot de relève des dérangements après sa mise en place chez Algérie Télécom. Une évaluation approfondie de l'efficacité et de la satisfaction des utilisateurs permettrait de mesurer les avantages concrets de cette solution. De plus, des améliorations continues du chatbot en termes de compréhension du langage naturel, de personnalisation des réponses et d'intégration avec d'autres systèmes pourraient être explorées. Enfin, cette recherche ouvre également la voie à d'autres travaux sur l'application des chatbots dans d'autres domaines des télécommunications et dans d'autres entreprises du secteur.

Bibliographie et Références

Référence :

[1].Adamopoulou, E., &Moussiades, L. (2020). Chatbots: History, technology, and applications. Machine Learning with Applications, 2, 100006.

[2].Ina. “The History of Chatbots – From ELIZA to ALEXA.”Onlim. 12 October 2017. https://onlim.com/en/the-history-of-chatbots/. (31 Décembre 2022).

[3]. LTSale. “Alice Bot.” . https://ltsale.2022outlet.ru/category?name=alice%20bot (31 décembre 2022).

[4]. Botwiki. “SmarterChild.”. https://botwiki.org/bot/smarterchild/ (Consulté le 31 décembre 2022).

[5]. Yeeply. “GPT-3 : Révolutionnaire de l'IA.”. https://fr.yeeply.com/blog/gpt-3-revolutionnaire-de-lia/ (le 31 décembre 2022).

[6]. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. "Deep Learning". MIT Press, 2016.

[7]. https://datascientest.com/train-test-split-tutoriel

[8]. "Three Ways To Mitigate Chatbot Risks." 2020.

[9]. Keras.io. "Adam Optimizer.". https://keras.io/api/optimizers/adam/ (le 31 décembre 2022).

[10]. Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks.In Advances in Neural Information Processing Systems (pp. 3104-3112).

[11]https://www.oracle.com/fr/database/what-is-json/

[12].http://dspace.univ-tlemcen.dz/bitstream/112/1062/6/ChapitreII.pdf

[13].https://datavalue-consulting.com/deep-learning-reseaux-neurones-recurrents-rnn/

Bibliographie :

[14]. "Comment réduire les risques lors du déploiement de solutions d'IA conversationnelle." 2020.

[15]. Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing. Pearson.

[16]. Moncoachdata. "Construire un chatbot en Python.". https://moncoachdata.com/blog/construire-un-chatbot-en-python/ (le 31 décembre 2022).

[17]. https://aclanthology.org/P15-1152.pdf

[18]. https://datatofish.com/json-string-to-csv-python/

[19]. https://www.math.univ-toulouse.fr/~besse/Wikistat/pdf/st-m-app-rn.pdf

[20]. https://www.lamsade.dauphine.fr/~croyer/ensdocs/TUN/PolyTUN.pdf

[21]. https://france.devoteam.com/paroles-dexperts/aller-plus-loin-en-deep-learning-avec-les-reseaux-de-neurones-recurrents-rnns

[22]. "4 Reasons for Enterprise Chatbot Failure and How to Overcome them using a Multi-Bot Approach." 2020.

[23]. Klopfenstein et al., 2017.

Annexes

1. Resource Description Framework (RDF) :

RDF (Resource Description Framework) est un modèle de données standard utilisé pour représenter des connaissances de manière structurée sur le Web. Il s'agit d'un cadre flexible qui permet de décrire des ressources, des relations entre les ressources et leurs attributs.

L'objectif principal de RDF est de fournir un moyen universel de représenter et d'échanger des informations sur le Web de manière sémantique. Il vise à permettre aux machines de comprendre et d'interpréter les données de manière automatisée en utilisant des vocabulaires et des schémas bien définis.

En utilisant RDF, les informations sont modélisées sous forme de triplets, composés d'un sujet, d'un prédicat et d'un objet. Le sujet représente la ressource principale que l'on souhaite décrire, le prédicat représente la relation ou la propriété qui lie le sujet à l'objet, et l'objet représente la valeur de la propriété ou une autre ressource liée.

L'adoption de RDF facilite l'interopérabilité des données sur le Web, car il permet de représenter les connaissances de manière structurée et standardisée. Les ressources RDF peuvent être liées entre elles, ce qui facilite la découverte et l'intégration des informations provenant de différentes sources.

En résumé, RDF offre un moyen puissant de représenter les connaissances de manière sémantique, favorisant ainsi l'échange et l'intégration de données sur le Web de manière cohérente et compréhensible par les machines.

- Modélisation des connaissances avec RDF :

Dans la modélisation des connaissances avec RDF (Resource Description Framework), plusieurs concepts clés sont utilisés pour représenter les informations de manière structurée. Ces concepts comprennent les ressources, les propriétés et les littéraux.

Une ressource dans RDF peut représenter une entité du monde réel, telle qu'une personne, un lieu, un objet, ou même un concept abstrait. Chaque ressource est identifiée de manière unique par une URI (Uniform Resource Identifier) qui sert d'identifiant global et persistant. L'utilisation de l'URI permet de référencer de manière univoque une ressource, facilitant ainsi son identification et sa récupération.

Les propriétés sont utilisées pour décrire les relations entre les ressources. Elles représentent les liens ou les attributs qui existent entre les ressources. Par exemple, on peut définir une propriété "a pour auteur" pour relier un livre à son auteur. Les propriétés sont également identifiées par des URI et peuvent être utilisées pour établir des connexions et des relations entre les ressources.

En plus des ressources et des propriétés, RDF permet également de représenter des valeurs littérales. Les littéraux sont des données de type simple comme des chaînes de caractères, des nombres, des dates, etc. Ils peuvent être utilisés pour attribuer des valeurs aux propriétés d'une ressource. Par exemple, une propriété "date de publication" peut être associée à un littéral représentant une date spécifique.

En utilisant RDF, les connaissances peuvent être modélisées en définissant des triplets, qui sont des assertions composées d'un sujet, d'un prédicat et d'un objet. Le sujet représente la ressource principale, le prédicat représente la propriété ou la relation, et l'objet peut être une autre ressource, une valeur littérale ou même une autre assertion.

La modélisation des relations et des attributs avec RDF permet de représenter les connaissances de manière structurée et cohérente. Elle facilite également l'intégration et l'échange de données entre différentes sources, en utilisant des identifiants uniques et des connexions sémantiques. Cette approche sémantique favorise la compréhension automatique des données par les machines et permet d'exploiter pleinement le potentiel des connaissances représentées dans RDF

- **Les langages et les outils RDF :**

Dans le domaine des langages et des outils RDF, plusieurs technologies sont utilisées pour travailler avec les données RDF et les connaissances représentées. Voici quelques-uns des principaux langages et outils RDF :

a. SPARQL : SPARQL (SPARQL Protocol and RDF Query Language) est un langage de requête spécifiquement conçu pour interroger les données RDF. Il permet d'effectuer des requêtes complexes pour récupérer des informations à partir d'un graphe RDF, en utilisant des clauses telles que SELECT, WHERE, FILTER, etc. SPARQL facilite l'interrogation et l'exploration des connaissances représentées dans RDF.

b. RDF Schema (RDFS) : RDF Schema est un langage de description utilisé pour définir des schémas pour les données RDF. Il permet de spécifier les classes, les propriétés et les relations entre les ressources. RDFS permet de définir des hiérarchies de classes, d'établir des relations entre les propriétés et d'ajouter des contraintes sur les données RDF.

c. OWL (Web Ontology Language) : OWL est un langage de modélisation d'ontologies utilisé pour représenter des connaissances formelles et explicites. Il offre des fonctionnalités avancées pour définir des hiérarchies de classes, des relations complexes, des règles d'inférence et des contraintes sémantiques. OWL permet de créer des ontologies riches et expressives pour décrire les domaines de connaissances.

d. Outils et bibliothèques : Il existe plusieurs outils et bibliothèques disponibles pour travailler avec RDF. RDFLib, Jena, Apache Fuseki, Virtuoso sont quelques exemples populaires. Ces outils offrent des fonctionnalités pour créer, manipuler, interroger et stocker des données RDF. Ils facilitent également l'interopérabilité et l'échange de données RDF entre différents systèmes.

L'utilisation de ces langages et outils RDF permet de tirer parti de la puissance du modèle RDF et d'exploiter les connaissances représentées de manière efficace. Ils offrent des fonctionnalités avancées pour la recherche, la requête, l'inférence et la gestion des données RDF, facilitant ainsi le développement d'applications basées sur les connaissances.

- **Utilisation de RDF dans le domaine des télécommunications :**

Dans le domaine des télécommunications, RDF (Resource Description Framework) trouve des applications pertinentes pour la gestion des données des opérateurs et la représentation des

entités et des relations dans les systèmes de télécommunications. Voici quelques exemples d'utilisation de RDF dans ce domaine :

a. Gestion des données des opérateurs : RDF peut être utilisé pour modéliser et gérer les données des opérateurs de télécommunications. Les informations telles que les abonnés, les contrats, les factures, les plans tarifaires, les équipements, etc., peuvent être représentées sous forme de graphes RDF. Cela permet une modélisation sémantique des données, facilitant l'interopérabilité, l'intégration et la recherche d'informations.

b. Représentation des entités et des relations : RDF offre un cadre flexible pour représenter les entités et les relations dans les systèmes de télécommunications. Par exemple, les abonnés, les lignes téléphoniques, les équipements réseau, les services, les dérangements, etc., peuvent être modélisés en tant que ressources RDF, avec leurs attributs et leurs relations spécifiées à l'aide de propriétés RDF. Cela permet de capturer les informations structurées et les liens entre les différentes entités, facilitant ainsi la gestion et l'analyse des données.

En utilisant RDF dans le domaine des télécommunications, il devient possible de réaliser des opérations de recherche avancées, de requête flexible, d'analyse de données et d'inférence. De plus, la représentation sémantique des données permet une meilleure compréhension des informations, favorisant ainsi l'interopérabilité et l'intégration entre les systèmes de télécommunications. Cela peut conduire à des améliorations significatives dans la gestion des données des opérateurs et la fourniture de services aux clients.

- **Avantages et limites de RDF :**

L'utilisation de RDF (Resource Description Framework) présente plusieurs avantages, mais elle comporte également certaines limites. Voici un aperçu des avantages et des limites de RDF :

Avantages de RDF :

1. Interopérabilité et réutilisation des données : RDF offre un modèle de données standardisé et basé sur des graphes, ce qui facilite l'interopérabilité entre les systèmes et la réutilisation des données. Les ressources RDF peuvent être facilement intégrées et liées à d'autres ressources, permettant ainsi une utilisation efficace des données dans différents contextes.

2. Extensibilité et évolutivité : RDF permet d'ajouter de nouvelles informations et de nouvelles relations de manière flexible, sans nécessiter de modifications majeures de la structure existante. Cela rend RDF adapté aux environnements en évolution où de nouveaux types de données et de relations peuvent apparaître avec le temps.

Limitations et défis de RDF :

1. Complexité de modélisation : La modélisation de données en RDF peut être complexe, surtout pour des domaines complexes ou des structures de données spécifiques. La création de modèles RDF bien conçus nécessite une compréhension approfondie du domaine et des compétences en modélisation sémantique.

2. Scalabilité des requêtes : Lorsque les ensembles de données RDF deviennent volumineux, la performance des requêtes peut être un défi. L'exécution de requêtes sur de

grands graphes RDF peut nécessiter des techniques d'optimisation et d'indexation avancées pour garantir des temps de réponse acceptables.

3. Adoption et familiarité : RDF et les technologies associées, comme SPARQL et les ontologies, peuvent nécessiter une courbe d'apprentissage pour les développeurs et les utilisateurs. L'adoption généralisée de RDF peut être limitée dans certains domaines en raison du manque de familiarité ou de compétences techniques.

Malgré ces limitations, RDF reste un outil puissant pour la représentation des connaissances, la modélisation de données et l'interopérabilité entre les systèmes. En comprenant ces avantages et limites, il est possible de tirer pleinement parti du potentiel de RDF tout en tenant compte des défis potentiels lors de sa mise en œuvre.

- **Cas d'étude : Application de RDF dans le chatbot de relève des dérangements :**

Dans le cas d'étude du chatbot de relève des dérangements, RDF est utilisé pour représenter les intentions, les motifs et les réponses du chatbot. Voici comment RDF est appliqué dans ce contexte :

1. Représentation des intentions : Les intentions du chatbot, qui correspondent aux différents types de demandes des utilisateurs, sont représentées en tant que ressources RDF. Chaque intention est identifiée par un URI unique, ce qui permet de les distinguer de manière univoque.

2. Représentation des motifs : Les motifs, qui sont les expressions ou les phrases-clés utilisées par les utilisateurs pour exprimer leurs demandes, sont également modélisés en tant que ressources RDF. Chaque motif est lié à son intention correspondante à l'aide de propriétés RDF, établissant ainsi la relation entre l'intention et les motifs associés.

3. Représentation des réponses : Les réponses fournies par le chatbot sont également représentées en tant que ressources RDF. Chaque réponse est liée à son intention correspondante à l'aide de propriétés RDF, permettant d'établir la correspondance entre l'intention de l'utilisateur et la réponse appropriée du chatbot.

L'utilisation de RDF présente plusieurs avantages dans la conception et le fonctionnement du chatbot de relève des dérangements :

- Flexibilité dans la modélisation des intentions : RDF permet de représenter les intentions de manière souple, ce qui facilite l'ajout ou la modification ultérieure de nouvelles intentions sans altérer la structure globale du chatbot.

- Liaison entre intentions, motifs et réponses : Grâce aux relations RDF, il est possible d'établir des liens entre les intentions, les motifs et les réponses correspondantes, ce qui permet au chatbot de comprendre et de répondre de manière appropriée aux demandes des utilisateurs.

- Réutilisation des ressources RDF : Les ressources RDF, telles que les intentions, les motifs et les réponses, peuvent être réutilisées dans d'autres applications ou projets liés aux dérangements des télécommunications. Cela favorise la cohérence et la gestion efficace des connaissances.

- Interopérabilité avec d'autres systèmes : En utilisant RDF, le chatbot peut facilement s'intégrer à d'autres systèmes ou sources de données qui utilisent également des modèles RDF. Cela facilite l'échange d'informations et la collaboration avec d'autres composants du système.

L'utilisation de RDF dans la conception du chatbot de relève des dérangements offre une approche flexible et interopérable pour représenter les intentions, les motifs et les réponses. Cela permet au chatbot d'offrir une expérience utilisateur améliorée et une gestion efficace des demandes de dérangements des utilisateurs.

2. JSON (JavaScript Objet Notation)

JSON (JavaScript Objet Notation) est un langage léger d'échange de données textuelles. Pour les ordinateurs, ce format se génère et s'analyse facilement. Pour les humains, il est pratique à écrire et à lire grâce à une syntaxe simple et à une structure en arborescence. JSON permet de représenter des données structurées (comme XML par exemple).

- Le fonctionnement de JSON :

Fondé sur un sous-ensemble de Javascript, JSON est un format texte totalement indépendant de tout langage. Pourtant, les conventions utilisées ne surprendront pas les codeurs familiers aux langages descendant du C tels que Javascript, Python, Pearl ou d'autres [11]

Autrement dit, il fonctionne un peu comme le XML (mais en moins développé) et facilite la structuration des informations présentes dans un document informatique. Comme il sert simplement à fluidifier l'échange de données, il n'est pas supposé contenir de commentaires, par exemple, ce qui le distingue d'un langage informatique à part entière. Toutefois, certaines bibliothèques en acceptent, s'ils sont écrits en JavaScript.

Le JSON fait partie des langages compréhensibles aussi bien par un esprit humain que par une machine. D'ailleurs, son apprentissage est facile et intuitif. Cependant, il reste très limité et ce qui le rend moins fiable et peu résistant en termes de sécurité.

- Les principaux cas d'utilisation du JSON :

1. Création d'un objet JSON à partir de données générées par l'utilisateur

Le JSON se prête parfaitement au stockage des données temporaires. Par exemple, les données temporaires peuvent être générées par l'utilisateur, notamment dans le cas d'un formulaire soumis sur un site Web. Le format de données JSON peut également être utilisé pour n'importe quel langage de programmation afin de fournir un niveau élevé d'interopérabilité.

2. Transfert des données entre les systèmes

Une base de données de site Web contient l'adresse postale d'un client, mais elle doit être validée via une API. Elle est envoyée au format JSON à l'API du service de validation d'adresse.

3. Configuration des données pour les applications

Lors du développement d'applications, chaque application a besoin des informations d'identification pour se connecter à une base de données ainsi qu'à un chemin de fichier journal. Les identifiants et le chemin du fichier peuvent être consignés dans un fichier JSON.

4. Simplification des modèles de données complexes

JSON simplifie les documents complexes jusqu'aux composants qui ont été identifiés comme significatifs en convertissant le processus d'extraction de données en un fichier JSON prévisible et lisible par l'humain.

Pourquoi le format JSON est-il populaire auprès des développeurs

Le JSON a pris de l'ampleur dans la programmation de code API et les services Web, car il permet d'accélérer l'échange de données et les résultats des services Web. Il s'agit d'un format textuel de données léger et facile à analyser qui ne nécessite aucun code supplémentaire pour l'analyse syntaxique. Pour les services Web, la nécessité de retourner et d'afficher beaucoup de données fait du JSON le choix idéal.

Qu'est-ce qu'une base de données de documents

Une base de données de documents est un type de base de données non relationnelle conçue pour stocker, extraire et gérer des informations orientées document. Au lieu d'avoir un schéma défini à l'avance, les bases de données de documents permettent de stocker des données dans des collections constituées de documents. Les bases de données NoSQL et JSON sont des types de base de données de documents.
Les bases de données de documents sont souvent populaires chez les développeurs, car elles permettent de stocker les données dans un format document-modèle (semi-structured) plutôt que relationnel (structured). Les bases de données de documents offrent davantage de flexibilité, car les développeurs n'ont pas à planifier les schémas à l'avance et peuvent utiliser le même format que celui utilisé dans leur code d'application. La planification minutieuse d'une base de données SQL s'avère donc moins nécessaire, ce qui rend les bases de données de documents utiles pour l'évolution rapide des schémas. Ces processus peuvent être courants dans le développement logiciel. Toutefois, cet avantage peut entraîner des problèmes de vitesse, de taille et de spécificité [11].

Qu'est-ce qu'une base de données de documents JSON

Les applications qui utilisent différents types de données JSON et un langage de requête orienté JSON peuvent interagir avec les données stockées dans une base de données de documents JSON. Une base de données de documents JSON prend également en charge nativement les documents JSON [11].

Voici les caractéristiques d'une base de données de documents JSON :

- Une base de données de documents JSON est une base de données non relationnelle conçue pour stocker et interroger des documents JSON
- Les données JSON de la base de données sont textuelles, mais le texte peut être stocké à l'aide du type de données BLOB, VARCHAR2, CLOB ou du type de données JSON binaire dans 21c
- L'accès aux données JSON stockées dans la base de données est similaire à l'accès à d'autres données de base de données, y compris à l'aide d'OCI, .NET et JDBC
- Les données JSON d'une base de données de documents JSON peuvent être stockées, indexées et interrogées sans schéma définissant les données

L'utilisation d'une base de données de documents JSON

Comme expliqué précédemment, le format JSON permet un transfert de données léger et constitue la norme pour l'échange de documents. À présent, voyons comment stocker et gérer des données JSON dans une base de documents JSON.

Stockage de données JSON

Le stockage des données JSON dans une base de données de documents JSON utilise des colonnes dont les types de données sont VARCHAR2, CLOB, BLOB ou JSON binaire dans 21c. Le choix du type à utiliser est généralement déterminé par la taille des documents JSON. Le stockage des données JSON dans la base de données à l'aide de types de données SQL standard signifie que les données JSON peuvent être manipulées comme n'importe quel autre type de données [11].

Gestion de données JSON

Les données JSON peuvent être gérées et manipulées à l'aide de tables dans une base de données de documents JSON, quel que soit le type de données. Le choix de la table à utiliser est généralement déterminé en fonction de la taille des documents JSON. Database propose notamment une fonctionnalité pour répliquer facilement des tables contenant des colonnes à l'aide de données JSON.

Les cas d'utilisation d'une base de données de documents JSON

Une base de données de documents JSON native permet d'afficher des données, d'en créer et en fin de compte d'en savoir plus.

Non seulement les clients d'une base de données sont identifiés par un nom et une adresse, mais leurs attributs respectifs peuvent être affectés en tant que valeurs et apparaissent dans des tableaux pour des recommandations de produits personnalisées et des interactions plus convaincantes.

L'efficacité des données JSON dépend uniquement de la base de données où elles se trouvent.

Les données JSON et les bases de données autonomes

Une base de données de documents JSON offre non seulement une prise en charge native pour les types de données JSON, mais également une migration facile, un développement low code et aucune modification du schéma en ce qui concerne le stockage et la gestion. Et s'il y avait un moyen de tirer parti des avantages de JSON dans un environnement sans serveur ? Une base de données de documents cloud facilite le développement d'applications JSON avec provisionnement, mise à l'échelle et réparation automatisés tout en fournissant une disponibilité de 99,995 %.

3. OWL (Web Ontology Language)

L'OWL (Web Ontology Language) est un langage de représentation des connaissances utilisé pour décrire des ontologies sur le Web sémantique. Il est conçu pour représenter des connaissances complexes et formelles, en utilisant des structures et des règles logiques, afin de permettre une représentation précise et un raisonnement automatisé sur les données.

L'Ontologie :

Une ontologie est une représentation formelle et explicite d'un domaine de connaissances. Elle définit les concepts, les relations et les contraintes qui régissent ce domaine spécifique. L'ontologie permet de structurer et de représenter les connaissances de manière logique et cohérente, ce qui facilite la compréhension, la recherche et l'inférence automatique sur les données.

Les fondements de l'OWL

L'OWL est basé sur la logique de description, qui permet de décrire formellement les classes, les propriétés et les relations entre les concepts. Il fournit un ensemble de syntaxes et de règles pour représenter les connaissances de manière précise et univoque.

L'OWL se compose de trois sous-langages, ou profils, avec différents niveaux d'expressivité : OWL Lite, OWL DL et OWL Full. Chaque profil a ses propres caractéristiques et restrictions, allant de la simplicité et de la limitation des fonctionnalités à la pleine expressivité avec plus de flexibilité mais aussi plus de complexité [12].

Les éléments de l'OWL

L'OWL utilise différents éléments pour décrire les ontologies :

- **Les classes** : représentent les concepts ou les catégories dans un domaine. Par exemple, une classe "Animal" peut être utilisée pour représenter tous les animaux.

- **Les propriétés** : décrivent les relations entre les classes ou entre les individus. On distingue les propriétés d'objet (qui relient les classes) et les propriétés de données (qui relient les valeurs de données aux individus).

- **Les individus** : représentent des instances spécifiques des classes. Par exemple, un individu "Chien" peut être une instance de la classe "Animal".

- **Les axiomes** : fournissent des déclarations logiques qui spécifient les relations entre les classes et les propriétés. Ils permettent de définir des contraintes et des règles pour le raisonnement automatique.

Les avantages de l'OWL

L'OWL présente plusieurs avantages pour la représentation des connaissances :

- Sémantique précise : L'OWL permet de spécifier de manière formelle et précise les relations entre les concepts, ce qui facilite l'interprétation et le raisonnement automatique sur les données.
- Raisonnement automatisé : L'OWL fournit des mécanismes de raisonnement qui permettent d'inférer de nouvelles connaissances à partir des informations existantes. Cela facilite l'exploitation des connaissances et l'obtention de réponses précises à partir de requêtes complexes.
- Interopérabilité : L'OWL permet de représenter les connaissances de manière standardisée et partagée, ce qui facilite l'échange et l'intégration des données entre différents systèmes et applications.
- Évolutivité : L'OWL permet de construire des ontologies évolutives, en ajoutant de nouveaux concepts, propriétés et règles au fur et à mesure de l'évolution des connaissances dans un domaine spécifique.

Utilisations de l'OWL :

L'OWL est largement utilisé dans de nombreux domaines pour représenter et raisonner sur les connaissances, notamment :

- Représentation des ontologies dans le domaine de la santé, de la biologie et de la médecine pour faciliter la recherche et l'interopérabilité des données.
- Représentation des connaissances dans le domaine de l'intelligence artificielle pour permettre le raisonnement automatisé et l'inférence logique.
- Représentation des données dans le domaine de la gestion des connaissances d'entreprise pour faciliter la recherche, la découverte et la gestion des informations.
- Représentation des connaissances dans le domaine des systèmes experts et des chatbots pour améliorer la compréhension et les réponses précises aux requêtes des utilisateurs.

En conclusion, l'OWL est un langage puissant et expressif pour la représentation des connaissances et la construction d'ontologies. Il facilite l'interopérabilité des données, le raisonnement automatisé et l'exploitation des connaissances dans différents domaines. En utilisant l'OWL, les développeurs et les chercheurs peuvent construire des systèmes plus intelligents et plus efficaces qui tirent parti de la richesse des connaissances disponibles sur le Web sémantique.