Institut Supérieur de l'Électronique et du Numérique
Tél. : +33 (0)2.98.03.84.00
Fax : +33 (0)2.98.03.84.10
20, rue Cuirassé Bretagne
CS 42807 - 29228 BREST Cedex 2 - FRANCE

Projet M1

Année scolaire 2021/2022

Chatbot pour l'école ISEN Brest

Proposé par : Monsieur Youssef MOURCHID
Encadré par : Madame Nadine ABDALLAH SAAB

Thématique : Informatique

DEPONTIEU Charles Domaine professionnel : Intelligence Artificielle
DEPIROU Guillaume Domaine professionnel : Génie Logiciel

Tables des Matières

1.Table des Figures

2.Glossaire

Liste des abréviations utilisées dans ce rapport :

API : Application Programming Interface est un type de programme informatique permettant à plusieurs applications de communiquer entre elles.

CSS : Cascading Style Sheets, langage de programmation web permettant de styliser et mettre en forme une page html.

HTML : HyperText Markup Language, c'est un langage de programmation permettant de créer et représenter une page web et sa structure.

IA : Intelligence Artificielle est un domaine informatique ayant pour objectif d'imiter l'intelligence humaine.

IDE : Integrated Development Environment est un logiciel intégrant des outils pour la programmation. D'une certaine façon, ce type de logiciel accompagne la mise en place de projets. Le choix de l'IDE de programmation est motivé par les spécificités du projet (analyse de données, recherche, développement web…).

JSON : JavaScript Object Notation est un format de fichier spécialisé dans le stockage de données. Créé au début des années 2000, ce format est supporté par un grand nombre de langages.

LinearSVC : Support Vector Classifier Linear est un algorithme d'optimisation des hyperparamètres en divisant ou classifiant les données d'entrées. Cette méthode se complète avec le SVM.

NLP : Natural Language Processing est un domaine sous-jacent du machine learning. Celui-ci vise la compréhension du langage humain par les machines.

RAM : Random Access Memory est un composant informatique permettant le stockage temporaire de données.

SaaS : Sofware as a Service est un type de logiciel stocké sur le drive. Ce type de logiciel apporte certains avantages sur le partage de fichiers et se démocratise de plus en plus.

SGDClassifier : Stochastic Gradient Descent Classifier est un algorithme d'optimisation utilisé pour trouver les meilleurs paramètres d'un modèle.

SVM : Support Vector Machine est un modèle linéaire de machine learning. Celui-ci est adapté aux problèmes de classification et de régression. De plus, cet algorithme permet de résoudre des problèmes linéaires ou non linéaires et est connu pour sa polyvalence.

VoIP : Voice over Internet Protocol est un ensemble de technologies dédiées à la communication en utilisant l'adresse Internet Protocol d'appareils.

VSCode : Visual Studio Code est un IDE fourni par Microsoft. Celui-ci se veut polyvalent dans de multiples langages de programmation. Particulièrement adapté aux projets web, des extensions permettent en effet d'accompagner le développement du code.

XML : Extensible Markup Language est un langage de balisage extensible. Il est utilisé pour faciliter l'échange automatisé entre deux applications.

3.Remerciements

Le développement de ce projet a impliqué de nombreux interlocuteurs que nous souhaitons remercier pour leur temps et leur accompagnement.

Dans un premier temps, nous tenons à saluer la bienveillance de Madame Abdallah Saab qui a su nous accompagner tant sur le point de vue technique que sur le point de vue gestion de projet.

Un remerciement particulier s'impose pour Monsieur Derrien dont les conseils avisés nous ont permis de définir la charte graphique de notre projet.

Nous tenons aussi à remercier vivement les auteurs de l'ensemble des informations sur lesquelles nous avons pu nous appuyer durant le rapport tels que python engineer et machine learnia.

Nous voulons exprimer notre reconnaissance sincère à Madame Kusberg pour le temps qu'elle a consacré à nous informer sur l'école IA Microsoft. Ces informations nous ont été précieuses pour comprendre et adapter notre projet au public souhaitant se renseigner sur cette école.

Ensuite, nous voulons adresser notre gratitude à Monsieur Mourchid qui nous a grandement aidé grâce à son implication par le biais d'un suivi régulier de notre avancement.

Nous avons aussi une pensée particulière pour l'ensemble de l'administration ayant contribué à l'organisation optimale et fructueuse de cette période de projet. En effet, la grande diversité des projets proposés en accord avec le nombre d'étudiants par Domaine Professionnel nous a permis, ainsi qu'à l'ensemble des étudiants, d'approfondir des thèmes et technologies étudiées précédemment.
De plus, la mise à disposition de salles nous a permis d'avancer notre projet en présentiel malgré la crise sanitaire. Enfin, nous tenons à faire part de notre reconnaissance sincère aux professeurs qui ont su gérer la distribution des projets d'une main de maître.

4.Présentation du projet et contexte

Ce projet a été réalisé dans le cadre de la formation d'ingénieur à l'ISEN Brest. Les étudiants en charge de ce projet sont : Guillaume Depirou & Charles Depontieu.
Le but de ce projet n'est pas unique. En effet, une multitude d'objectifs sont au cœur de celui-ci allant de l'appropriation des exigences liés au domaine de l'intelligence artificielle à la rapide mise à disposition d'informations précises et cohérentes pour l'utilisateur du chatbot. En se rapprochant de situations professionnelles, les étudiants s'habituent à une situation similaire à celles qui pourront se présenter à eux lors de leur entrée sur le monde du travail.

Pour favoriser cette simulation de projets professionnels, l'encadrant en charge du projet détient deux rôles :

- Le client : en jouant le client, le référent du projet cherche à proposer un problème à analyser par les étudiants. Cela leur permet d'adopter une méthodologie de suivi de projet, d'analyse et de recherche du besoin. De plus, ceux-ci doivent chercher à optimiser les solutions à proposer.
- Le référent technique : ce rôle permet d'accompagner les étudiants dans leurs recherches des technologies répondant de manière optimale aux problèmes du client.

Le présent rapport répond à la demande de développement d'un chatbot pour les écoles ISEN et IA Microsoft. Le rapport dresse les solutions apportées mais s'attache également à décrire d'autres aspects du projet comme celle de la gestion de celui-ci.

5.Introduction

Les chatbots sont, depuis quelques années, revenus sur le devant de la scène. En effet, l'utilisation de ces programmes informatiques permettant d'établir une interaction entre un utilisateur et un robot s'est accrue concomitamment à la numérisation de l'économie et des services. Cette technologie apparaît dans les années 1950 avec le test de Turing. Ils permettent d'automatiser une communication, d'informer un consommateur ou de renseigner un élève. Le retour des chatbots, disparu dû à une technologie sous-développée, est principalement lié à l'avènement des réseaux sociaux et des applications de messagerie comme Messenger ou WhatsApp. En effet, les chatbots fonctionnent sur tout type de site ou d'application et cela est devenu un aspect essentiel pour une entreprise. Avec l'évolution des technologies liées l'intelligence artificielle, les chatbots s'imposent de manière croissante sur les plateformes numériques comme gages d'optimisation de l'expérience client.

Les chatbots sont une technologie nécessitant plusieurs composantes techniques. La première composante, sa partie émergée, correspond à un script ou un modèle permettant de trouver une réponse à une phrase donnée. L'intelligence artificielle intervient à ce stade, plusieurs modèles sont possibles pour trouver une réponse et les améliorations des technologies les rendent toujours plus performants. Cependant, le chatbot nécessite une partie supplémentaire : la page web qui va permettre de récupérer les questions, transmettre les réponses à un utilisateur, tout simplement une messagerie. Ces agents conversationnels sont souvent perçus sur des sites web comme des petites boites de dialogue. Une application permet de récupérer les questions, de trouver la réponse grâce au modèle d'intelligence artificielle et de la renvoyer au site web qui l'intègre alors dans sa boîte de messagerie pour l'afficher à l'utilisateur.

L'évolution constante des chatbots et la multitude de tâches associées à ce projet sont vecteurs d'un intérêt certain et d'une motivation curieuse. De plus, la complexité et les nouveautés inhérentes à un chatbot sont des éléments formateurs, en lien avec l'actualité, pour tout élève ingénieur. La mise en place et l'utilité d'un tel projet dans les perspectives de croissance d'une entreprise se révèle être un enjeu stratégique majeur.

Ce rapport se décompose selon les différentes étapes de conception du chatbot, mais aussi en prenant en compte l'évolution de celui-ci, les différents choix techniques effectués et leurs justifications.

Dans la première partie, la présentation du cahier des charges est détaillée, avec les objectifs à atteindre pour la fin du projet.

Ensuite, la deuxième partie s'attache à la gestion de projet, en précisant les délais fixés et les délais réels ainsi que les outils utilisés pour mener à bien la tâche.

La troisième partie permet de se centrer sur l'explication du fonctionnement de la partie intelligence artificielle de ce projet, notamment sur la manière dont le chatbot sélectionne ses réponses mais aussi sur les difficultés rencontrées.

A la suite de cette partie, une présentation de l'application et de la partie web du chatbot est proposée, les choix techniques effectués et les changements correspondants sont également mentionnés.

Enfin, la conclusion met en exergue les points techniques rencontrés et les améliorations possibles de notre projet.

6.Cahier des charges

6.1.Présentation

Le présent rapport collectif s'attache à l'élaboration technique et à la mise en place pratique d'un système de chatbot pour deux écoles de l'enseignement supérieur. En effet, l'ISEN Brest et l'école IA Microsoft pourraient ainsi voir leurs sites internet dotés d'une telle technologie. Ces deux formations farouchement liées au secteur numérique se doteraient alors d'une technologie à l'avenir florissant gage de confort et d'efficacité pour les utilisateurs.

6.2.Présentation ISEN Brest

L'Institut Supérieur de l'Électronique et du Numérique (ISEN) de Brest est une grande école d'ingénieurs des hautes technologies et du numérique, accessible après le bac (école d'ingénieurs en 5 ans).
La formation de l'ISEN Brest ne se cantonne pas aux matières techniques ou scientifiques. Les élèves-ingénieurs apprennent à communiquer, à manager, à prendre de la hauteur sur leurs activités et surtout, comme personne ne connaît les technologies de demain, ils apprennent à apprendre. L'ISEN a un fonctionnement binaire : 3 ans pour se former, 2 ans pour se spécialiser. L'ISEN propose un large éventail de cursus allant du cycle généraliste au cycle informatique et réseaux impliquant de nombreux domaines professionnels, tels qu'Intelligence Artificielle ou Génie Logiciel.
L'ISEN n'est pas seulement une école, en effet on y retrouve également le laboratoire de recherche L@bISEN – Yncréa Ouest. Trois lignes de force portent les activités de recherche du L@bISEN : les réseaux de capteurs, le traitement des données et l'énergie.

6.3.Présentation école IA Microsoft

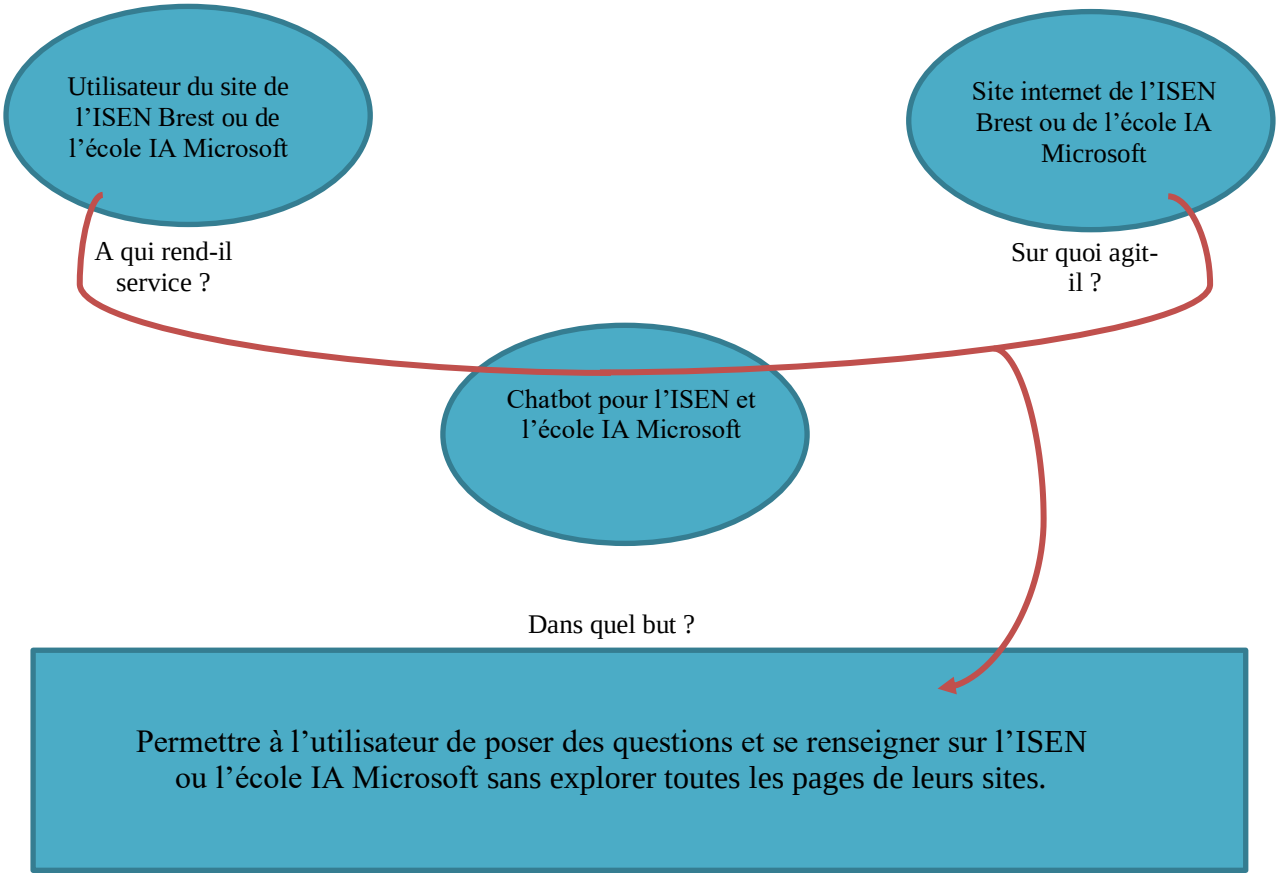
L'IA Microsoft de Brest fait partie du réseau Simplon. Il est constitué de formations numériques inclusives en France et à l'étranger. Simplon a ainsi formé plus de 7000 personnes à travers 111 fabriques dans le monde.

Le programme école IA Microsoft By Simplon a permis de former plus de 400 demandeurs d'emploi en France depuis sa création en 2018. Cette formation gratuite est à destination de publics éloignés de l'emploi ou sous représentés dans le monde du numérique, sans prérequis de diplôme. Ainsi, une formation en pédagogie active s'appuyant sur des « use cases » entreprises, avec une phase intensive puis un contrat en alternance, est dispensée dans cet établissement.

Enfin, celle-ci se déroule sur 7 mois en formation intensive et 12 mois en formation en alternance ce qui apporte une grande professionnalisation aux apprenants.

6.4.Les objectifs

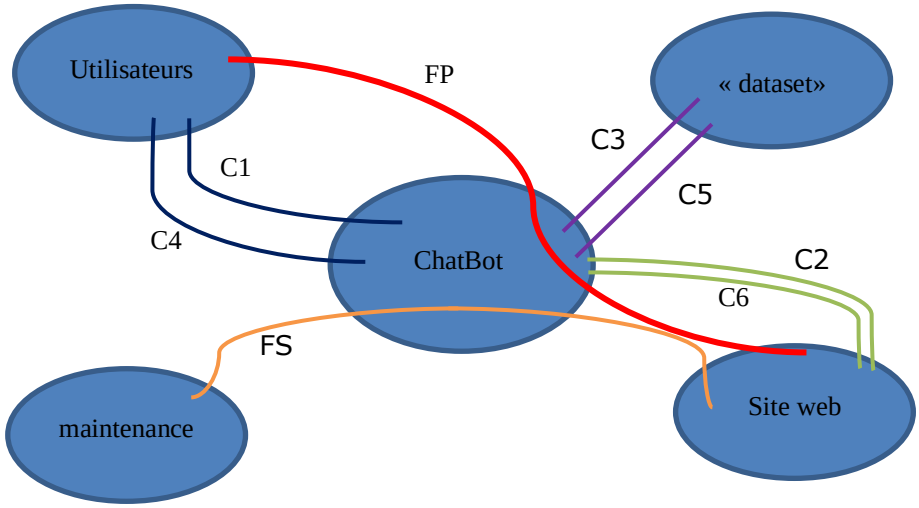
Méthode de formulation de projets privilégiée à l'ISEN Brest, la « Bête à corne » permet de préciser celui-ci explicitant le cadre dans lequel il s'inscrira. Cette technique apporte également de précieuses informations sur le périmètre du projet. Enfin, cette conceptualisation génère un gain de temps et d'objectivité non négligeable dans la conduite d'un projet de long terme.



6.5.Diagramme pieuvre

Fonction principale (FP), Fonction Secondaire (FS) et contraintes (C)		
	Priorité (MoSCoW)	
FP	Must have this	Renseigner l'utilisateur à travers une discussion
FS	Could have	Apprendre des réponses de l'utilisateur (apprentissage par renforcement)
C1	Constraint	Facile d'accès et d'utilisation
C2	Constraint	Faible temps de réponse
C3	Constraint	Intelligent (comportement cohérent)
C4	Constraint	Responsif
C5	Constraint	Être cohérent avec le discours de l'ISEN / Microsoft IA
C6	Constraint	Adaptable au site de l'ISEN et/ou Microsoft ISEN

Éléments extérieurs : utilisateur, « dataset », site web, maintenance



6.6.Fonctionnalités et types de chatbot

La dissociation de deux types de chatbots pour une compréhension optimale de ces choix de technologies : les chatbots "simples" et les chatbots "intelligents".

Les premiers ne fonctionnent pas avec de l'IA. En effet, les « simples » se basent sur un arbre de décision. C'est-à-dire que le chatbot va guider l'utilisateur vers un enchaînement de questions. Les structures et réponses sont ainsi prédéfinies. Pour une meilleure compréhension, en voici un exemple.

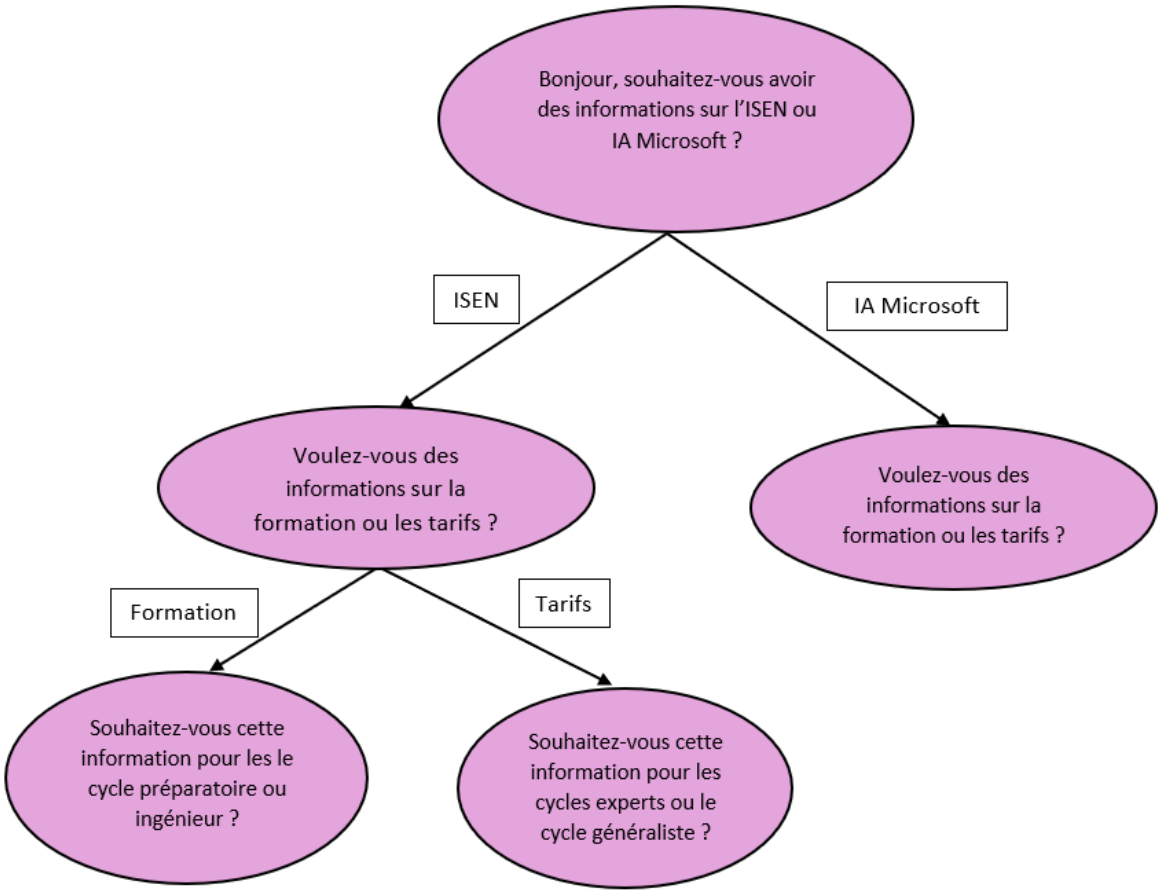


Figure 1 : Exemple de chatbot dit "simple"

Les chatbots fonctionnant avec de l'IA sont définis comme apprenants ou intelligents. En se basant sur le Natural Language Processing (NLP), ces IA se basent sur l'apprentissage supervisé. La phase d'apprentissage nécessite un corpus composé de tags (classes), patterns (phrases) et réponses associées. L'objectif est de prédire l'appartenance de la question de l'utilisateur à un tag. Ainsi, les réponses du corpus seront utilisées pour répondre à l'utilisateur.

	Avantages	Inconvénients
Chatbots « simples »	Facile à mettre en œuvre, peu coûteux, pas d'entraînement nécessaire, contrôle parfait des réponses.	Ne comprend pas les questions pour lesquelles il n'est pas prévu, ne peut pas s'améliorer par l'apprentissage, l'échange paraît peu naturel, fonctionnement par bouton/mots clefs uniquement.
Chatbots « intelligents »	Peut s'améliorer par l'apprentissage, l'échange paraît naturel, peut répondre à des questions sur lesquelles il n'a pas été entraîné.	Difficile à mettre en place, entraînement à faire, coûteux.

6.7.La partie recherche / étude analytique

La première chose lorsque l'on souhaite développer un outil est de regarder si cela existe déjà. Si c'est le cas, il est utile de regarder ce qui est fait, comment cela est fait et les éléments dont il faut s'inspirer. De plus, des idées de fonctionnalités supplémentaires en les testant permettrait d'améliorer ce projet. Une des premières missions réalisées a donc été de s'informer sur les différents chatbots, plus ou moins développés par les entreprises. Ces informations ont permis une première esquisse du chatbot que ce projet allait permettre de développer.

Le site de l'ISEN a servi de base pour repérer les différentes questions que pourrait poser un futur utilisateur. Le parcourir a permis de commencer la base de données et la développer à partir des informations trouvées. Dans notre recherche sur l'école IA Microsoft, nous n'avons trouvé que très peu d'éléments à son sujet. Après contact avec Madame Kusberg, plus d'informations sur cette formation ont pu être obtenues ainsi qu'une liste de questions fréquemment posé concernant l'école IA Microsoft. Madame Kusberg a également pu nous orienter vers des étudiants ayant réalisé un chatbot pour l'école IA Microsoft et leur GitHub.

Une recherche technique a également été effectué pour comprendre les technologies utilisées lors de la création d'un chatbot et son intégration à un site web. A l'aide de formations et

d'explications sur youtube, ainsi que d'aides de forums, une certaine compréhension des taches à réalisés s'est formée.

Cette partie de recherche effectuée, elle constituera la base du projet et de son développement. Les recherches se poursuivront tout au cours du projet en fonction des difficultés ou changements à apporter.

6.8.Définition du livrable

6.8.1.API

L'API a pour vocation d'améliorer le chatbot et d'intégrer la database au site, il permet également d'accéder au corpus pour le modifier. L'objectif est de continuer le développement d'un chatbot vers un apprentissage à renforcement.

6.8.2.Rapport Technique

Un rapport technique sera livré à la fin du projet. Celui-ci évoquera les différentes solutions que nous avons choisi pour établir ce chatbot ainsi que les problèmes que nous avons pu rencontrer.

6.8.3.Lien Github & Trello

Un lien github et un lien trello sont attendus à la fin du projet pour suivre notre gestion de projet et pouvoir évaluer le projet final et son code. Le planning sera fait automatiquement sur le trello.

Lien github : https://github.com/gdepirou/chatbot-isen

Lien trello :
https://trello.com/invite/b/F1SjO9Hp/cf530294e8b5dbba2307cbba6be2f394/gestion-de-projet

6.9.Pérennisation et futur du livrable

6.9.1.Intégration

L'intégration du chatbot élaboré à la page internet de l'ISEN Brest est l'objectif final de ce projet. L'accès à un serveur n'étant pas possible, le développement du livrable s'est réalisé en local. De plus, le script final devrait être facilement intégrable sur le site de l'ISEN. Enfin, la présentation d'une architecture claire du code du projet sera requise.

6.9.2.Fonctionnalités supplémentaires

A terme, le chatbot pourrait recevoir des fonctionnalités supplémentaires à sa fonction essentielle de "question-réponse". Les quelques fonctionnalités qui pourraient être ajoutées seraient : un bouton permettant la réinitialisation du chatbot, des boutons proposant à l'utilisateur une question selon le sujet qu'il choisit, un affichage plein écran. Cependant, d'autres pistes d'améliorations pourraient être mises en œuvre tout au long du projet pour suivre une méthode agile.

7.Gestion de Projet

La gestion de ce projet dans un cadre temporel prédéfini est un élément professionnalisant pour les étudiants en ingénierie numérique. En effet, il est important d'apprendre à organiser la temporalité du projet avant et pendant celui-ci. Dans cette optique, un diagramme de Gantt a été planifié au début de ce projet, celui-ci a été mis à jour au fur et à mesure de l'avancement.

7.1.Difficultés rencontrées

L'encadrant du projet, Monsieur Mourchid, a présenté ses attentes sur le projet dès le mois de décembre. Ainsi, il a été possible de commencer la maquette au début janvier comme initialement prévu. Le défi majeur mais primordial pour le développement de ce chatbot était la formation. En effet, le chevauchement des cours de domaines professionnels et la réalisation de ce projet a constitué une difficulté supplémentaire dans l'apprentissage d'éléments spécifiques à la technologie du chatbot.

Une autre difficulté rencontrée concerne le changement d'encadrant. À la suite du départ de Monsieur Mourchid au mois de février, Madame Abdallah Saab a repris le projet. Cependant, ce remplacement a été vecteur de nouvelles méthodes de travail et d'organisation ainsi que d'un accompagnement renforcé. En effet, le développement d'un réseau de neurones était demandé. Madame Abdallah Saab a orienté le développement sur une approche focalisant le propos autour du deep et machine learning. Cette analyse a conduit vers différents algorithmes de machines learning adaptés au projet. Suite au changement d'algorithme, aucune solution n'a été trouvé pour garder l'API demandé et procurer un apprentissage par renforcement.

Par ailleurs, une exigence supplémentaire est survenue au mois de février, le cahier des charges apparaissant désormais comme un élément obligatoire. La rédaction de ce document s'est déroulée en même temps que la phase de programmation, ce qui a été pénalisant pour l'avancement général.

7.2.Outils utilisés

Une méthode agile a été utilisée pour s'adapter aux contraintes du projet. En effet, le choix d'une communication régulière par le biais de Teams s'est rapidement imposé afin de fluidifier les échanges avec la référente du projet.

Microsoft Teams est un logiciel de communication en mode SaaS. Principalement utilisé en tant que plateforme collaborative, elle est utilisée en complément des autres logiciels de Microsoft comme la suite Office 365.

Concernant les autres outils mis en place pour ce suivi continu et régulier du bon déroulement du projet, il est intéressant de noter leurs spécificités.

Trello est un logiciel de gestion de projet en ligne. Avec son interface intuitive et ergonomique, cet outil propose un fonctionnement modulable par un système de carte correspondant à des tâches. Enfin, son grand nombre d'extensions lui permet de se connecter à d'autres services tel GitHub ou TeamGantt présenté ci-dessous.

L'utilisation de Trello pour ce projet collectif s'est imposée comme nécessaire. En effet, ce logiciel a permis au binôme d'identifier les tâches qui lui été dédiées. Cet outil a été utilisé pour le suivi global du projet dont l'organisation est visible sur la figure 2.

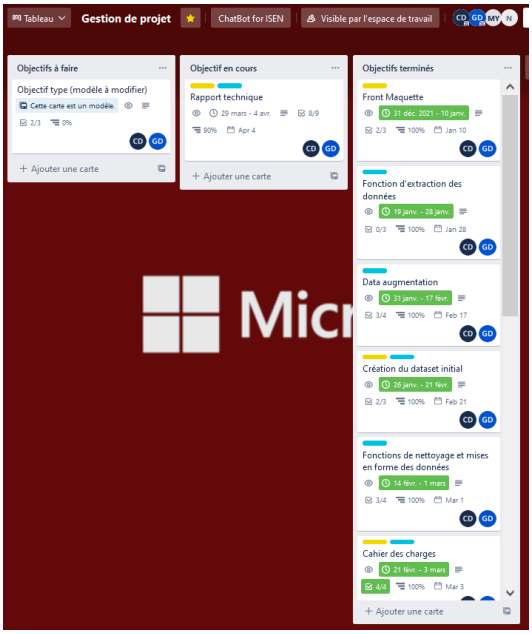


Figure 2 : Tableau Trello utilisé

Un diagramme de Gantt est un type de diagramme adapté pour la gestion de projet. Cette figure tend vers un travail de groupe idéalement réparti. En effet, ce diagramme a pour spécificité d'établir un lien de dépendance entre toutes les tâches et le temps qui sera nécessaire à leur réalisation. Ainsi, il s'agit d'un outil d'optimisation d'élaboration et de mise en œuvre d'un projet. L'extension TeamGantt de Trello est disponible pour rendre un diagramme de Gantt automatique et permettre un suivi réel de la progression des tâches.

L'utilisation de cette extension a permis de suivre le projet et d'adapter la répartition des tâches. Une présentation de notre Gantt prévisionnel puis réel est disponible ci-dessous.

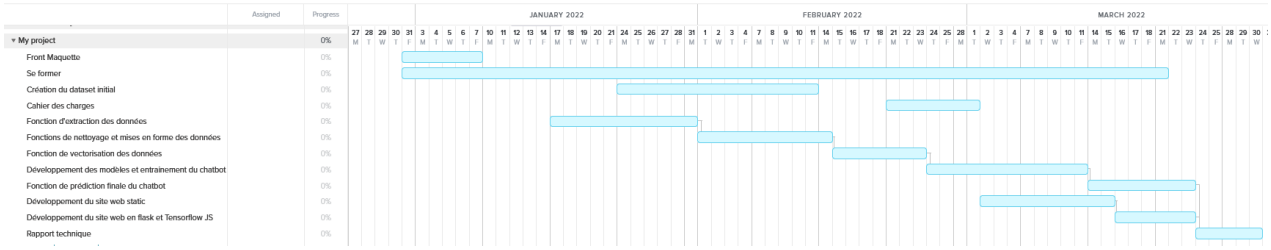


Figure 3 : Diagramme de Gantt prévisionnel

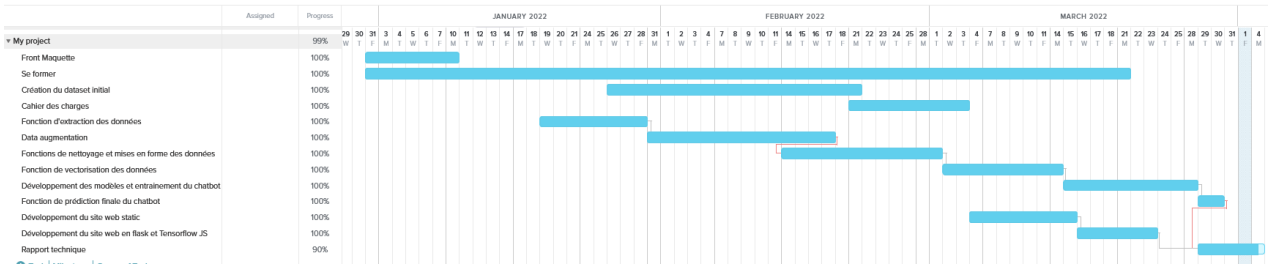


Figure 4 : Diagramme de Gantt réel

Github est un logiciel d'hébergement et de gestion de développement informatique en ligne. En se basant sur le logiciel de gestion de version « git », cette version web y apporte une interface ergonomique.

Ce logiciel a permis une sauvegarde permettant un suivi sécurisé du code du chatbot. De plus, la présentation de l'aspect informatique du projet est adaptée à son interface en ligne.

Discord est un logiciel de type VoIP. Celui-ci permet la création de salons, souvent utilisés pour catégoriser les types ou thèmes d'échanges. Initialement créé autour des jeux vidéo, l'outil a évolué vers une utilisation polyvalente.

Tandis que Trello a été utilisé pour le suivi global du projet, Discord a permis une communication interne efficace. Le choix de cet outil a été motivé par la possibilité de séparer les sujets de discussions (planning, IA, Web, Rapports journaliers) et de garder un historique des fichiers partagés (cf Annexe 3).

7.3.Organisation générale

Les lieux de travails sont restés volontairement libres. Chaque membre a pu choisir entre l'ISEN et le distanciel. Concrètement, un mélange entre ces deux formats s'est imposé suivant les besoins d'échanges internes au binôme ou avec Madame Abdallah Saab.

De plus, un suivi hebdomadaire a été convenu avec le client, quand cela était possible, pour conserver une gestion agile. L'outil de compte-rendu de réunion a été mis de côté, d'un commun accord, avec notre premier professeur référent.

Concernant les horaires de travail, une plage horaire allant de 9h à 18h a été établi pour permettre à chacun de travailler efficacement. De plus, chaque membre était libre de travailler en dehors de ces horaires. Cela a permis de faciliter les échanges en collaborant fréquemment.

7.4.Evolution chronologique

La première semaine a été consacrée à la compréhension du projet et à l'établissement d'un premier rendu pour le futur site. Les semaines suivantes se sont surtout axées sur la réalisation de la maquette et de la charte graphique, ainsi que sur l'apprentissage des différents outils utilisés lors du projet. En effet, n'ayant jamais réalisé de chatbot auparavant, une documentation sur cette technologie et ses applications existantes était nécessaire. Des premiers tests concernant les procédés d'intelligence artificielle ont été effectués de même que la réalisation du cahier des charges.

Tandis que le chatbot était en construction, une première version du site incluant le JavaScript et Flask a pu être élaborée, avec une réponse test d'un script python. L'étape suivante de l'intégration s'est faite relativement sans problème. Celle-ci a pu mettre en lumière certains défauts qui ont pu être corrigé par la suite.

7.5.Planning et délais

Le planning initial (visible sur la figure 3) a relativement bien été respecté. Un retard sur le développement de chacune des fonctions est visible sur le Gantt réalisé.

Ce délai est dû au temps de formation nécessaire en parallèle. De plus, certaines fonctions, comme « data augmentation », n'étaient initialement pas prévues mais en réalité, nécessaires aux performances du modèle. De plus, la création de la base de données initiale a été faite en parallèle de la programmation. De fait, le temps consacré à celui-ci a été sous-évalué. De plus, des reprises ont été nécessaires sur les fonctions précédentes alors même que les suivantes étaient d'ores et déjà en phase de conception, ce qui a rallongé le délai accordé à chacune des tâches.

Ce retard a été compensé par la réduction du temps consacrée à la rédaction du rapport ainsi qu'à l'utilisation de la marge de sécurité de 5 jours prévue initialement. En effet, le diagramme de Gantt prévisionnel s'arrête volontairement le 30 mars.

Les délais ont été respectés dans leur globalité, bien qu'un peu de retard a été accumulé à la suite d'imprévus. Notre planning prévisionnel était donc cohérent, cependant une marge de sécurité un peu plus importante aurait pu être judicieuse.

7.6.Technologies utilisées

Pour la bonne réalisation du chatbot, il a été décidé d'utiliser certaines technologies, sélectionnées pour leurs avantages par rapport au projet réalisé.

Python est un langage de programmation interprété de haut niveau. Ses principaux avantages sont : une programmation facilitée et polyvalente permettant une gestion automatique de la mémoire, des exceptions et un typage dynamique, tout en restant adapté aux débutants et largement documenté sur internet.
A contrario, son principal défaut est sa lenteur et les ressources matérielles qu'il exige.
Mais encore, des librairies importantes et essentielles pour le projet tel que Pandas, Numpy ou scikit-learn sont basées sur python.

Ce projet de chatbot étant un projet étudiant, il a été décidé de choisir ce langage, appris et travaillé lors de la formation, pour se concentrer sur les parties algorithmiques.

Le HTML, le CSS et le JavaScript sont des langages complémentaires permettant de créer un site web et de rendre celui-ci dynamique. Le HTML créera la structure du site, tandis que le style du site sera ajusté à l'aide du CSS. Le JavaScript est là quant à lui pour la partie dynamique du site, les différentes animations et algorithmes sont contenues dans ce fichier. Il permet également de relier une page web à un framework tel flask.

Flask est un framework open-source de développement web python. Celui-ci est considéré comme un microframework pour sa légèreté. Dans un souci d'optimisation, Flask est apparu comme une solution plus rapide que son concurrent Django et possède également une utilisation moins complexe et plus simple à mettre en place.

Anaconda est une distribution open-source de python. Particulièrement adapté aux projets de data-science, ce package comprend des logiciels tels que JupyterLab, Notebook, Spyder ou encore VSCode. De plus, il offre une possibilité de programmation dans des langages adaptés à la data science tel que R. Enfin, celui-ci présente une interface intuitive pour l'importation de librairies ou la création d'environnements virtuels.

Son principal inconvénient concerne sa lenteur et son besoin important de RAM.
Ce choix de distribution a été motivé pour sa praticité.

JupyterLab est la version la plus complète de développement python sous format notebook. Sa flexibilité permet l'optimisation de projets de data science et d'intelligence artificielle. Son principal atout est la modularité que le système de notebook permet, divisant les multiples phases de recherche pour améliorer la clarté du projet. Cependant, sa lenteur en fait un outil adapté à l'innovation et aux phases exploratoires mais peu pour des usages commerciaux.

Les limites de l'outil se sont fait sentir à la fin du développement du chatbot.

Virtual Studio Code est un IDE gratuit développé par Microsoft. Il comprend plusieurs fonctionnalités comme la mise en évidence de la syntaxe selon le type de fichier, la complétion ou la prise en charge du débogage. Les nombreuses extensions en font également un outil très pratique. Cependant, l'affichage de donnée n'est pas optimisé et peu visuel.
La multitude de type de fichier, allant du python au html, permet un développement facilité sur une seul interface pour ce projet.

Adobe XD est un logiciel gratuit, dont il existe une version payante, qui permet la création de maquette de site web ou d'application. La version gratuite est reconnue comme étant un des meilleurs concepteurs de maquette. L'ajout d'action permet un rendu dynamique des applications, les fonctionnalités sont faciles d'usages et multiples. Le seul inconvénient aurait pu être l'exclusivité de certaines fonctionnalités à la version payante, mais l'usage ne celle-ci n'ayant pas été nécessaire, ce fut le logiciel utilisé pour notre maquette (cd Annexe 1).

Les détails concernant les librairies utilisées durant ce projet sont disponibles ci-dessous :

Librairie NLTK. Cette librairie est spécialisée pour le NLP et permet l'apport de plusieurs dizaines de corpus et de ressources pour tester ses fonctions. Elle permet notamment de catégoriser les textes, d'y sélectionner les mots et idées importantes et de faire une analyse linguistique sur une phrase. De plus, son aspect open-source lui permet d'avoir un nombre important de documentation. Son principal inconvénient est la langue de construction des fonctions, la majeure partie ne fonctionnant qu'en Anglais.

Librairie deep_translator. Cette librairie a été choisie pour pallier l'inconvénient de NLTK. Effectivement, en comprenant plusieurs traducteurs et de la détection automatique de langage, elle offre des opportunités indéniables. De plus, son usage est aisé dû aux fonctions intégrées, faciles à prendre en main et à mettre en place. Enfin, c'est l'une des rares librairies proposant ces fonctionnalités de manière stable et gratuite.

La librairie.json est spécialisée dans le transfert de données entre fichiers. Elle est utilisée pour récupérer les données d'un fichier.json. Son fonctionnement est simple et similaire aux librairies Marshal et Pickle.

La librairie Pickle est faite pour sauvegarder un objet python dans un fichier. Elle stocke cet objet sous format binaire. Ainsi, elle permet de sauvegarder les données d'un objet python (tuple, dictionnaire ou liste…) et d'éviter l'utilisation d'une base de données. Contrairement à la librairie précédente, elle n'est pas limitée à un seul format d'objet ce qui la rend polyvalente.

La librairie joblib fait partie des essentielles. Elle permet notamment de choisir le nombre de processeurs à utiliser pour l'exécution d'un programme. Le choix de ce paramètre permet de gérer la charge prise par le CPU et la rapidité d'exécution de celui-ci. Cette librairie est utilisée d'une manière similaire à Pickle.

La librairie String contient des constantes, des fonctions et classes pour la manipulation de chaînes de caractère. Elle a été utilisée dans ce projet pour optimiser la création de certaines listes tel les lettres de l'alphabet.

Numpy est une librairie utilisée pour la manipulation de tableaux, matrices et fonctions mathématiques. Créée en 2005, elle reprend des fonctions de la librairie Numeric. Ce module open-source est utilisé dans de nombreux domaines et est reconnu comme faisant partie des impératifs à utiliser. Contrairement à d'autres librairies de ce projet, Numpy est développée en C et disponible sous plusieurs langages de programmation. En l'important sous python, son fonctionnement se rapproche de Matlab qui est aussi un langage interprété.

Le module pandas créé en 2008 est aujourd'hui pratiquement obligatoire pour les projets avec manipulation et analyse de données. En effet, ses fonctions intégrées sont reconnues comme les plus développées dans ce domaine. Ainsi, un chatbot utilisant des données d'entraînement nécessite les librairies NLTK, Numpy et pandas.

Le module sklearn (ou scikit-learn) est sans doute la librairie la plus utilisée pour du machine learning en python, suite aux classes, fonctions et modèles comprises dans celle-ci. Le chatbot a nécessité l'utilisation de plusieurs de ces fonctions (le détail se fera dans la suite du rapport) :

- Sklearn.preprocessing avec la fonction OneHotEncoder
- Sklearn.model_selection avec les fonctions GridSearchCV et train_test_split
- Sklearn.ensemble avec la fonction RandomForestClassifier
- Sklearn.linear_model avec la fonction SGDClassifier
- Sklearn.metrics avec la fonction ConfusionMatrixDisplay

8.Partie Intelligence Artificielle

8.1.Choix du stockage du « dataset »

Le choix du format pour conserver notre jeu de données initiale a été motivé par plusieurs raisons. Tout d'abord, l'impossibilité d'accéder à un serveur pour un stockage des données. Puis, faire en sorte que ce format soit adapté au remplissage manuel des données.

Le format .json a donc été sélectionné car les données étaient sous format texte. De même, celui-ci est adapté à conserver un grand nombre de données. De plus, ce type de fichier est plus adapté à être rempli de manière automatique. Bien que non réalisé ici, ceci est une piste d'amélioration pour ce chatbot.

De plus, n'ayant pas de « dataset » initial sur lequel se baser, celui-ci a été créé à partir des textes de communication de l'ISEN ainsi que ceux de l'école IA Microsoft.

Les textes réponses sont pour la plupart issus de ces documents. En ce qui concerne les « tags » (thème) et les « labels » (questions correspondant aux réponses), ceux-ci ont pu être complétés manuellement. Le « context » permet de définir si les données sont reliées à l'ISEN ou à l'école IA Microsoft.
En effet, certaines questions peuvent avoir une réponse différente suivant les deux contextes.

Voici un exemple d'un tag du « dataset » dans un fichier .json :

```
"tag": "bienvenue",
"patterns": ["Salut", "Salut à tous", "Bonjour à tous", "Bonjour", "Coucou", "Comment ca-va?", "Comment tu vas?", "je recherche de l'aide","oyez oyez","hey"],
"responses": ["Bonjour, comment puis-je vous aider?"],
"context": ["ISEN"]
```

Figure 5 : Exemple de composition du "dataset" initial au format json

Concernant le nombre de questions (patterns) par classe (tag), il est nécessaire de l'équilibrer. Seul le corpus de l'ISEN l'est. Cela justifie les meilleures performances de ce corpus comparé à celui de l'école IA Microsoft.

8.2.Présentation du fonctionnement du code

La figure 6 ci-dessous cartographie le fonctionnement général des fonctions amenant à la prédiction du chatbot.

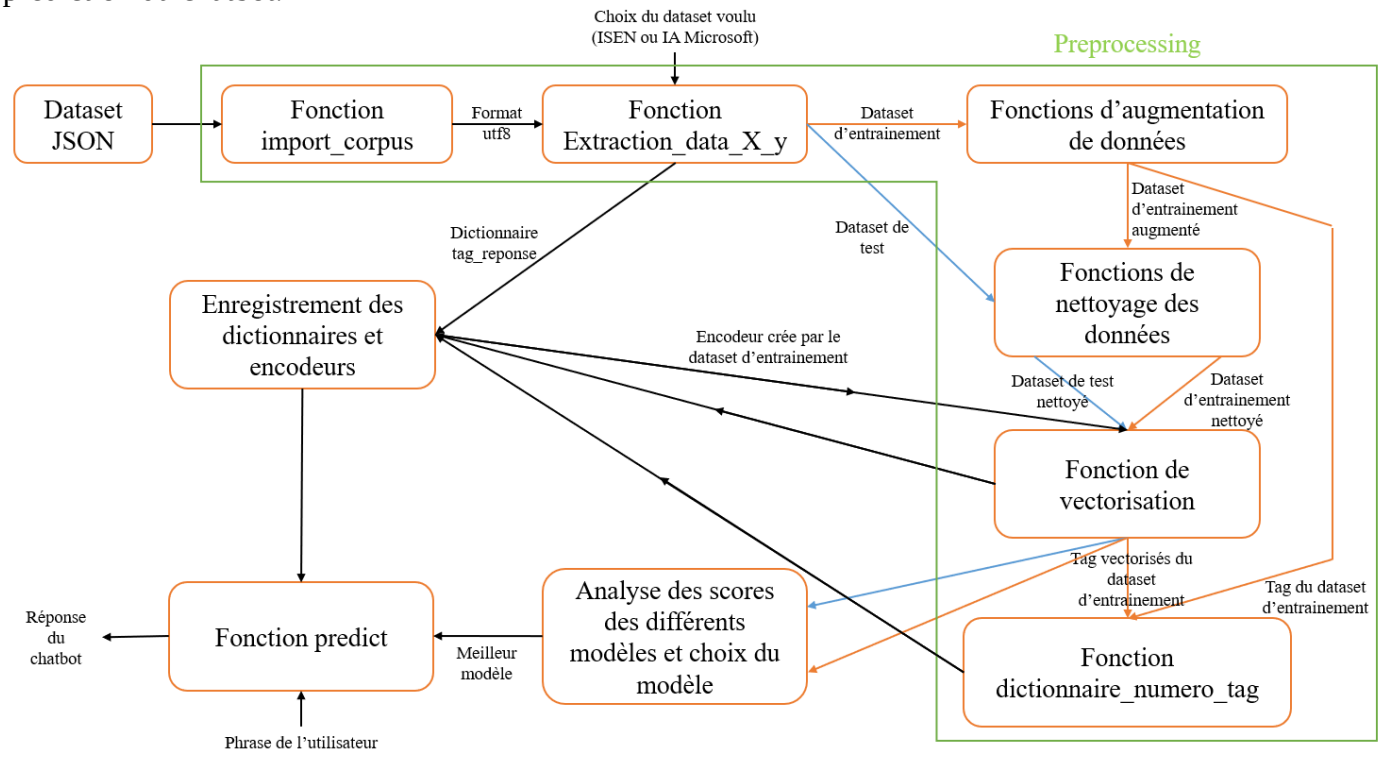


Figure 6 : Cartographie du fonctionnement général des fonctions amenant à la prédiction du chatbot

8.2.1.Fonction import_corpus

```python
# import our chat-bot intents file
import json

def import_corpus(jsonfile): #permet d'importer le corpus
    try:
        with open(jsonfile, encoding='utf-8') as json_data:
            intents = json.load(json_data)
        return intents

    except OSError as err:
        print("OS error: {0}".format(err))
    except ValueError:
        print("Your .json is wrong, look at https://jsonlint.com/")
    except BaseException as err:
        print(f"Unexpected {err=}, {type(err)=}")
        raise
```

Figure 7 : Fonction import_corpus

La première partie concerne l'importation de l'ensemble du corpus, défini dans le fichier "intents.json". Cette fonction nécessite en entrée le document .json et sa particularité est la récupération des données encodés en utf-8 et non ASCII.

8.2.2.Fonction Extraction_data_X_Y

L'importance de la séparation des données d'entrées, en une partie d'entrainement et une partie de test, nécessite un exemple pour être mis en exergue. Ainsi, si un modèle apprend à partir des phrases suivantes :

- Quels sont les tarifs du cycle CIR ?
- Quels sont les frais de scolarité du cycle CIR ?

Le modèle aura tendance à classifier les phrases contenant les mots "CIR", "tarifs" ou encore "frais" dans le tag "tarif_CIR". Cependant, si un utilisateur pose la question suivante : "Combien coûte le cursus Cycle Informatique et Réseaux ?", le modèle ne prédira pas cette

phrase dans le tag "tarif_CIR". En effet, aucun des mots déterminants, trouvé à l'aide de la partie entrainement des données, n'est présent dans la phrase inscrite par l'utilisateur.

Le manque de données d'entrainements peut être démontré ici à l'aide de phrases sur lesquelles le modèle n'a pas appris. Ces phrases constitueront la partie test du jeu de données et permettront la vérification des performances du chatbot.

Pour obtenir une réponse à chaque recherche sans avoir à réutiliser le fichier.json, la création d'un dictionnaire associant le tag et une réponse adéquate est appropriée. Ce dictionnaire aura donc comme clef le tag et comme valeur associée sa réponse.

La fonction « extraction_data_X_y » est voulue polyvalente, elle a deux modes de fonctionnement possibles :

- Extraire les données du corpus avec les arguments suivants : testing = false, phrase = None
    - La fonction va récupérer les tags, patterns et réponses. L'utilisateur de la fonction peut choisir de récupérer les données concernant l'ISEN (context = "ISEN") ou celles de l'école IA Microsoft (context = "IA-MICROSOFT").
      De plus, la fonction va enlever les ponctuations des patterns. Ceci afin de commencer à enlever les informations sans plus-value.
      Ensuite, le « dataset » des patterns (X) et des tags (y) est réparti en deux jeux de données différents : le premier dédié à l'entraînement et un autre dédié au test. Par convention, le premier comprendra 80% des données. D'autres pourcentages ont pu être testés sans réels avantages.
      Enfin, le dictionnaire expliqué plus tôt, associant chaque tag avec la réponse associée, est retourné par la fonction.

      Pour mieux comprendre, voici ce que rend la fonction avec les paramètres suivant : jsonfile, testing = False, phrase = None, context = « ISEN »

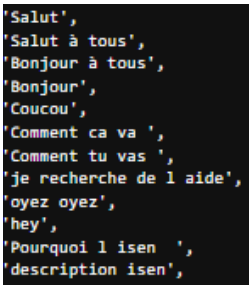


Figure 8 : Rendu X_texte (liste des labels) de la fonction extraction_data_X_y

:

```
'bienvenue',
'bienvenue',
'bienvenue',
'bienvenue',
'bienvenue',
'bienvenue',
'bienvenue',
'bienvenue',
'bienvenue',
'bienvenue',
'raison_isen',
'raison_isen',
```

Figure 9 : Rendu y_texte (liste des tags) de la fonction extraction_data_X_y

```
{'bienvenue': ['Bonjour, comment puis-je vous aider?'],
 'raison_isen': ['La maitrise de ces compétences est au cœur de la formation ISEN. Elle permet une approche métier transdisciplinaire dans des
domaines d'application aussi variés que les technologies pour la santé, l'environnement, les transports, l'aéronautique… ou une approche par s
pécialité (cyber sécurité, production d'énergie, cloud computing, robotique, objets connectés, big data…).'],
 'cursus_isen': ['Vous construisez votre parcours sur 5 ans. Après le baccalauréat, vous optez pour un cycle de formation en 3 ans centré sur
les sciences et le numérique. En fonction de votre profil, cinq cursus différents sont proposés. Pendant ces trois premières années, vous réfl
échissez à votre projet professionnel et vous pouvez changer de cursus. Ensuite, vous accédez sans critère de classement aux 40 domaines profe
ssionnels proposés par l'ISEN et les autres écoles d'ingénieurs d'Yncréa (HEI, ISA) . C'est le temps de la spécialisation.'],
```

Figure 10 : Rendu dico_tag_responses de la fonction extraction_data_X_y

- Enlever les ponctuations d'une phrase avec les arguments suivants : testing = True, phrase = "phrase testée par l'utilisateur". Ce cas est utilisé comme une étape de nettoyage de la phrase donnée par l'utilisateur du chatbot.

Pour mieux comprendre, voici ce que rend la fonction avec les paramètres suivant :
testing = True, phrase = " Ceci est une phrase pour tester la disparition des ponctuations, comme la suppression de l'apostrophe."

```
extraction_data_X_y(testing = True,
                    phrase = "Ceci est une phrase pour tester la disparition des ponctuations, comme la suppression de l'apostrophe.")

'Ceci est une phrase pour tester la disparition des ponctuations  comme la suppression de l apostrophe '
```

Figure 11 : Rendu de la fonction extraction_data_X_y

8.2.3.Fonctions d'augmentation de données

Dans une problématique d'augmentation du nombre de données pour éviter un « overfitting » (surapprentissage), il a été requis de faire une « data augmentation ». En effet, un risque du développement de modèle d'Intelligence Artificielle est un manque de généralisation du modèle. Durant le développement du chatbot, ce problème a été identifié par une grande performance sur les données d'entraînement et une très faible performance sur les données de test (les données sur lesquelles le modèle n'a pas été entraîné).
Dans le cadre du NLP, ce processus consiste à créer de nouvelles phrases à partir des phrases existantes dans notre « dataset ».

La fonction data_augmentation_syn est composée de deux autres fonctions que voici :

- Find_synonyms : En utilisant la librairie « deep_translator » avec « GoogleTranslator », cette fonction prend un entrée un mot et rend les synonymes. Le choix de « GoogleTranslator » est justifié par la stabilité de l'API de Google. En effet, des tests avec d'autres traducteurs comme « Deepl » se sont révélés plus performants sur les

traductions mais largement moins stables. Le principal inconvénient de cette fonction est sa logique à double traduction. En passant du français en anglais puis inversement, certaines traductions se révèlent peu pertinentes. Cet inconvénient est en partie réduit avec une limitation sur la taille des mots à traiter. En effet, un mot de moins de 3 lettres ne subira pas la fonction « find_synonyms ».

```
find_synonyms("Bonjour")

['Bonjour', 'Tiens', 'salut', 'comment allez-vous']
```

Figure 12 : Rendu de la fonction find_synonyms

- Create_set_of_new_sentences : Cette fonction utilise la fonction précédente pour générer des phrases synonymes. Celle-ci est paramétrable avec le « max_syn_per_word » et « nbr_new_sentences ». La première fixe le nombre de synonymes par mot que cherche la fonction. La seconde indique le nombre de phrases synonymes que la fonction doit retourner.

```
create_set_of_new_sentences("C est quoi le cursus de l école", nbr_new_sentences = 2)

['C est quoi le programme de l école',
 "C est quoi le programme d'études de l école"]
```

Figure 13 : Rendu de la fonction create_set_of_new_sentences

- Data_augmentation_syn : Cette fonction permet d'obtenir une liste de « label » (X) et de « tag » (y) renforcée par « data augmentation ». Ainsi, avec un « dataset » initial (cas context = "ISEN"), on passe de 272 phrases à 754. De la même manière, nous passons (cas context = "IA-MICROSOFT") de 237 à 694 phrases. Néanmoins, cette fonction présente comme inconvénient majeur sa lenteur. En effet, son utilisation d'internet via l'API de « GoogleTranslator » n'optimise pas son temps d'exécution.
-

8.2.4.Fonctions de nettoyage des données

L'objectif de cette série de fonctions est de permettre de "sélectionner" le contenu d'une phrase. Pour comprendre les programmes ci-dessous, il est nécessaire de connaître deux opérations :

- Les « stop-words »
- Le « stemming »

Ces deux concepts ont été choisis dans la librairie NLTK pour leur performance en français. En effet, d'autres fonctions intéressantes se sont révélées inutilisables en français, et ce même après passage dans un traducteur.

Pour la compréhension d'une phrase, il est admis que les humains accordent plus d'importance à certains mots clés. De la même façon, il est également reconnu que certains mots sont considérés comme inutiles. Dans ce projet, les mots n'apportant que peu d'information, donc inutiles à la phrase, sont à exclure. Ce concept correspond à supprimer, des phrases, les mots présents dans la liste des « stop-words ».
Voici la liste des « stop-words » français :
au, aux, avec, ce, ces, dans, de, des, du, elle, en, et, eux, il, ils, je, la, le, les, leur, lui, ma, mais, me, même, mes, moi, mon, ne, nos, notre, nous, on, ou, par, pas, pour, qu, que, qui, sa, se, ses, son, sur, ta, te, tes, toi, ton, tu, un, une, vos, votre, vous, c, d, j, l, à, m, n, s, t, y, été, étée, étées, étés, étant, étante, étants, étantes, suis, es, est, sommes, êtes, sont, serai, seras, sera, serons,

serez, seront, serais, serait, serions, seriez, seraient, étais, était, étions, étiez, étaient, fus, fut, fûmes, fûtes, furent, sois, soit, soyons, soyez, soient, fusse, fusses, fût, fussions, fussiez, fussent, ayant, ayante, ayantes, ayants, eu, eue, eues, eus, ai, as, avons, avez, ont, aurai, auras, aura, aurons, aurez, auront, aurais, aurait, aurions, auriez, auraient, avais, avait, avions, aviez, avaient, eut, eûmes, eûtes, eurent, aie, aies, ait, ayons, ayez, aient, eusse, eusses, eût, eussions, eussiez, eussent.

L'idée du deuxième concept, le « stemming », est une méthode de normalisation. En effet, tous les mots d'une même famille apportent une information similaire. Cependant, ce ne sont pas exactement les mêmes. L'objectif est de réduire le mot à sa racine.
Ainsi, "Ce rapport est superbement réussi" et "Ce rapport aura une superbe note" doit être compréhensible de la même manière.
Après l'étape de stemming, on obtient les phrases suivantes : "Ce rapport est superb réuss" et "Ce rapport aur une superb not". On observe que les mots superbement et superbe ont rendu le même mot. L'étape de « stemming » a permis, en gardant la racine des mots, de normaliser leur analyse.

Ainsi, plusieurs étapes sont réalisées :

- Clean_text : Cette fonction prend en entrée une phrase et retourne une liste de mots. Ceux-ci correspondent à la phrase après suppression des « stop words » et « stemming ».

```
clean_text("Ce rapport aura une superbe note pour leur travail de qualité")

['rapport', 'superb', 'not', 'travail', 'qualit']
```

Figure 14 : Rendu de la fonction clean_text

- Nettoyage : Cette fonction permet d'effectuer clean_text sur l'ensemble d'une liste de phrases.

- Clean_multiple : Cette fonction découle d'un inconvénient de la fonction « Create_set_of_new_sentences ». En effet, en basculant du français en anglais puis inversement, il est possible que plusieurs phrases synonymes en anglais donnent la même phrase en revenant en français. Nous voulons donc supprimer ces doublons. La fonction prend en entrée la liste des « labels » (X) et des « tags » (y) et les adaptent en enlevant les répétitions. Suite à l'exécution de cette fonction, le nombre de « label » par « tag » (phrase par classe) dans le « dataset » d'entraînement n'est plus équilibré. Cela peut amener certaines erreurs de classification. En effet, si un tag est entraîné avec plus de labels qu'un autre, il est possible que plus de phrases tests lui soient associées. La gestion du nombre de labels par tag avant vectorisation peut être une piste d'amélioration.

```
phrase initiale : Salut à tous
['salut à tous', 'Salut à tous']
NOUVELLE phrase : salut à tous
NOUVELLE phrase : Salut à tous

phrase initiale : Bonjour à tous
['Tiens à tous', 'salut à tous']
NOUVELLE phrase : Tiens à tous
NOUVELLE phrase : salut à tous
```

Figure 15 : Certains rendus spécifiques de la fonction create_set_of_new_sentences

- Collage : les fonctions précédentes rendent des listes de listes de mots. Pour reformer les phrases, il est nécessaire de rassembler ces mots avec la fonction collage.

```
Avant collage : [['écol', 'ingénieur', 'général'], ['scolar', 'ingénieur', 'général']]
Après collage : ['écol ingénieur général', 'scolar ingénieur général']
```

Figure 16 : Rendu de la fonction collage

8.2.5.Fonction Vectorizer

Le but de cette fonction est la vectorisation des phrases. En effet, le chatbot ne comprend pas tant le français que d'autres langues. Les programmes informatiques ne comprennent que le langage binaire, composé de 1 et de 0. Dans ce cadre, il est nécessaire de transformer un texte en vecteur binaire. Ainsi, l'ordinateur pourra comprendre le texte et sa signification. Ce processus s'appelle la vectorisation de texte et convertit donc un texte en nombre compréhensible par une machine.

Le détail du processus est expliqué ci-dessous :

Phrase d'apprentissage 1 : mot1 mot2 mot3
Phrase d'apprentissage 2 : mot4
Phrase d'apprentissage 3 : mot1 mot3

Phrase de test : mot1 mot2 mot3 mot5

Liste de mots de la base d'apprentissage	mot1	mot2	mot3	mot4
Phrase d'apprentissage 1	1	1	1	0
Phrase d'apprentissage 2	0	0	0	1
Phrase d'apprentissage 3	1	0	1	0
Phrase de test	1	1	1	0

Une observation de la disparition des mots inconnus dans les données de test peut être effectué. En effet, les mots jamais rencontrés lors de la phase de test sont automatiquement supprimés.

Le choix de l'encodeur a amené à une série de test et de réflexion pour trouver celui correspondant le mieux aux données. En effet, certaines étapes automatiquement réalisées par certains encodeurs ont pu être développées à la main dans un premier temps. Ce développement a permis de mieux comprendre le fonctionnement de chaque encodeur et de finaliser notre choix.
A titre d'exemple, un fonctionnement testé est la récupération de tous les mots composant la base d'apprentissage, pour pouvoir les enlever de la base de test. Ensuite, un regroupement de la base d'apprentissage et de test était requis pour la fonction de vectorisation. Enfin, une séparation des vecteurs formés pour reformer une base de test et une base d'entraînement était effectuée. Un fonctionnement séparant la création de l'encodeur et la vectorisation a été préféré. Ce système permet une meilleure lisibilité, un meilleur débugage et une plus grande rapidité d'exécution.

La fonction « vectorizer » permet de traiter deux cas possibles :

- La vectorisation de l'ensemble du « dataset» d'entraînement (les labels ET les tags) avec les arguments suivants : train = True, encodeur = None
    - En prenant en entrée la liste (label X ou tag y) d'entraînement, cette fonction renvoie la liste vectoriser ainsi que l'encodeur créé.
- La vectorisation d'une phrase ou d'un « dataset » de phrases tests inconnues (labels) avec les arguments suivants : X_test, train = False, encodeur = encodeur créé lors de la vectorisation du « dataset» d'entraînement.

- La fonction utilise l'encodeur donné en entrée et rend la phrase ou l'ensemble de phrases vectorisées.

```
Données d'entrainements sur lesquels l'encodeur sera construit :
['mot1 mot2 mot3', 'mot4', 'mot1 mot3']

Données d'entrainements vectorisées :
[[1 1 1 0]
 [0 0 0 1]
 [1 0 1 0]]
----------------------
Données de test sur lesquels on applique l'encodeur :
['mot1 mot2 mot3 mot5']

Données de test vectorisées :
[[1 1 1 0]]
```

Figure 17 : Entrées et sorties de la fonction vectorizer

8.2.6.Fonction dictionnaire_numero_tag

Un modèle de classification d'Intelligence Artificielle ne prédit pas une réponse. En effet, il requiert des vecteurs binaires en entrée et retourne un chiffre correspondant à un tag. Cependant, il est nécessaire d'associer ces deux éléments pour la mise en œuvre de la fonction « predict ». Contrairement à un tuple (chiffre_prédit, "tag_associé"), un dictionnaire n'est pas ordonné. Les chiffres prédits ont été placés dans un dictionnaire comme clef des noms de tags associés.

Pour ce faire, la fonction « dictionnaire_numero_tag » prend en entrée les tags vectorisés et les tags initiaux. Celle-ci permet de former un dictionnaire comme décrit ci-dessous.

```
{4: 'contact_IA_MICROSOFT',
 17: 'recrutement_IA_MICROSOFT',
 6: 'duree_IA_MICROSOFT',
 5: 'cout_IA_MICROSOFT',
 12: 'localisation_IA_MICROSOFT',
 18: 'teamwork_IA_MICROSOFT',
 9: 'evaluation_IA_MICROSOFT',
 2: 'certification_IA_MICROSOFT',
 11: 'inclusion_IA_MICROSOFT',
 13: 'logement_IA_MICROSOFT',
 1: 'alternance_IA_MICROSOFT',
 10: 'formateurs_IA_MICROSOFT',
 14: 'materiel_IA_MICROSOFT',
 3: 'confinement_IA_MICROSOFT',
 16: 'pédagogie_IA_MICROSOFT',
 15: 'presentation_IA_MICROSOFT',
 8: 'entretiens_IA_MICROSOFT',
 7: 'emploi_IA_MICROSOFT',
 19: 'technologie_IA_MICROSOFT',
 0: 'actionnaires_IA_MICROSOFT'}
```

Figure 18 : Dictionnaire du corpus IA_MICROSOFT associant les chiffres prédits aux tags

```
{9: 'bienvenue',
 31: 'raison_isen',
 15: 'cursus_isen',
 12: 'campus_ouest',
 13: 'certifications',
 42: 'Taux_insertion',
 45: 'Yncrea',
 30: 'Nombre_etudiants_alumni_etudiants',
 38: 'tarif_CGSI',
 39: 'tarif_CIR',
 36: 'tarif_BIOST',
 37: 'tarif_CENT',
 41: 'tarif_EST',
 35: 'tarif_BIAST',
 40: 'tarif_cycle_ingenieur',
 10: 'Bourses',
 34: 'SHN',
 26: 'ISEN_logement',
 27: 'ISEN_restauration',
 44: 'visite_virtuelle_PO',
 25: 'Handicap',
 14: 'Clubs_etudiants',
 11: 'La vie à Brest',
 28: 'Job_ISEN',
 2: 'admission_general',
 0: 'admission_aprèsbacgeneral',
 1: 'admission_bac+2',
 4: 'admission_parallele_1ereannee',
 5: 'admission_pass',
 3: 'admission_par_apprentissage',
 32: 'Recherche',
 22: 'differentes_formations',
 20: 'cycle_généraliste',
 21: 'cycle_informatique',
 17: 'cycle_biologie',
 18: 'cycle_economie',
 19: 'cycle_environnement',
 16: 'cycle_agronomie',
 23: 'differents_domaine_pro',
 24: 'doubles_diplomes',
 6: 'alternance',
 7: 'apprentisage',
 8: 'au revoir'}
```

Figure 19 : Dictionnaire du corpus ISEN associant les chiffres prédits aux tags.

8.2.7.Enregistrement des dictionnaires et encodeurs

La plupart des programmes ci-dessus ne sont à utiliser qu'avant l'entraînement du modèle. Ainsi, leur temps de chargement important et de développement sous python ne pose relativement pas de problème lors de l'utilisation du chatbot. Une sauvegarde des modèles entraînés est cependant requise, ainsi que celle des éléments permettant le fonctionnement du chatbot. Pour ce faire, l'utilisation des librairies « Pickle » et « joblib » est nécessaire. Celles-ci permettent de sauvegarder au format .sav pour les modèles et au format .pkl pour les dictionnaires.

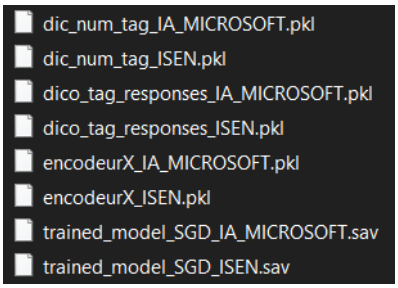


Figure 20 : Enregistrement des dictionnaires et modèles

8.2.8.Modèle sélectionné

Pour compléter les informations au début de ce rapport, plusieurs modèles ont pu être testés. La sélection des algorithmes s'est basée sur une cartographie par « sklearn » sous forme d'arbre de décision.

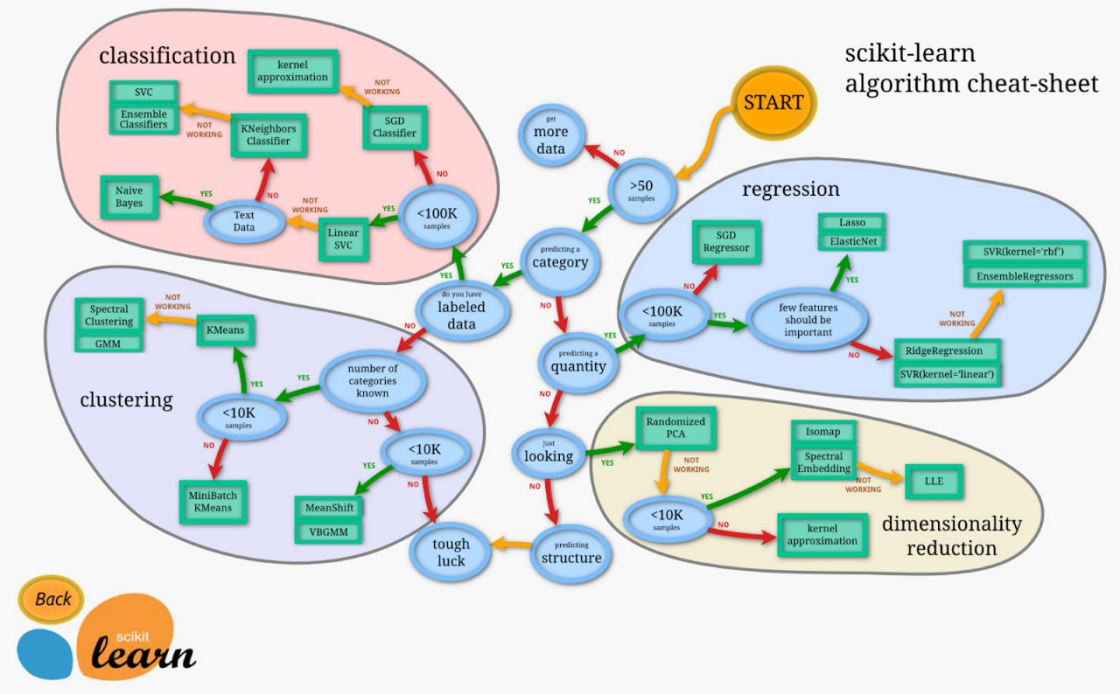


Figure 21 : Cheat sheat algorithme par "sklearn"

Cette cartographie est fournie comme une aide et non comme une règle. En effet, celle-ci indique un cheminement à suivre avec une partie des algorithmes existants. Ainsi, certains algorithmes comme le « random forest » ne sont pas présents.
Parmi ceux-ci, il a été testé le SGDClassifier, le LinearSVC et le Random Forest.

Le « random Forest » n'étant pas décrit dans cet inventaire d'algorithmes, celui-ci se démarque par certains avantages par rapport au chatbot :

- Utilise la logique « d'ensemble learning » en combinant les prédictions d'un ensemble d'arbres de décisions. Cette logique permet d'éviter un « overfitting », généralise les prédictions et réduit la sensibilité de l'algorithme aux valeurs aberrantes.
- Peut gérer automatiquement les valeurs manquantes.
- Est reconnu pour sa stabilité.

Cependant, cet algorithme n'a pas obtenu de bons scores. Cela est principalement dû à sa complexité. En effet, en entraînant un ensemble important d'arbres et en combinant leur prédiction, l'algorithme nécessite une grande période d'entraînement et un grand nombre de données, ce qui est une faiblesse de notre « dataset ».

Le linearSVC a été testé car étant adapté à une classification de moins de 100 000 données (ce qui est le cas ici). De plus, cet algorithme est efficace lorsqu'il y a une grande différence entre chaque classe (les « tags »). Malgré notre nettoyage des données, certaines classes (concernant les tarifs de chaque filière par exemple) sont relativement proches.

Au départ de notre projet, il avait été indiqué d'utiliser un réseau de neurones pour faire ce chatbot. Cependant, le choix entre machine et deep learning est principalement défini par la

quantité de données à disposition. En effet, en définissant certaines règles (sélection des mots importants…) en machine learning, l'apprentissage nécessitera un « dataset » moins important qu'en deep learning.

Enfin, l'algorithme choisi, pour ses meilleures performances, est le SGDClassifier. Cela s'explique principalement par sa capacité à minimiser rapidement la fonction de perte. Le « dataset », certes petit, amène assez de données à l'algorithme pour l'apprentissage. De plus, le choix de cet algorithme a été motivé par son évolutivité pour le projet. En étant particulièrement adapté à un grand « dataset », un agrandissement de la base de données initiales permettra de conserver ce modèle.

8.2.9.GridSearchCV et CrossValidation

Les précédentes fonctions ont nettoyé, normalisé et vectorisé les données d'entrainement, il est maintenant nécessaire de choisir les paramètres de notre modèle. En faisant varier ces derniers, l'objectif est d'améliorer les performances du modèle en les rapprochant le plus possible de 1 (100% de bonnes classifications, score théorique à atteindre).

Dû à un nombre de données relativement faible, ce projet a rapidement souffert « d'overfitting ». A l'aide de la fonction « validation_curve » de « sklearn », une analyse des performances d'apprentissage d'un modèle est possible en faisant varier un paramètre. Après de nombreux essais, le nombre de données du « dataset » initial et donc celui pour l'apprentissage s'est révélé trop petit. Un problème de généralisation était visible.
Une phrase A venant du « dataset » d'entraînement du modèle était bien prédite. Une phrase B proche de la A venant du « dataset » de test n'était presque jamais bien prédite.
Pour palier cela, une data-augmentation a été réalisée dans une fonction précédente ainsi qu'un réglage automatique des paramètres avec « GridSearchCV ».

Le choix du paramètre « CV » est à régler avec la fonction « GridSearchCV ». Celui-ci correspond au nombre de « split » (séparation possible du jeu de données) du « cross-validation ».
Pour évaluer la performance d'un modèle, il est nécessaire de le tester sur des données qu'il n'a jamais rencontrées. Cependant, qu'est ce qui indique que le découpage entre données de test et d'entraînement est représentatif de tous les découpages possibles ?
C'est à ce niveau qu'intervient ce nombre de « split ». Ce paramètre fixera le nombre de découpage à tester. Ainsi, suivant l'exemple ci-dessous, un modèle peut être plus performant (modèle A sur le « split1 ») sur un « split » particulier mais moins performant au global.
Ce paramètre est à maximiser en évitant de former des « split » contenant trop peu de données.

Figure 22 : Explication du fonctionnement du cross-validation (source : Machine Learnia)

Dans le cas du modèle « SGDClassifier », il a été défini grâce aux « validations curves » que les paramètres importants étaient "loss", "alpha" et "penalty". Le principe du « GridSearchCV » est de tester toutes les combinaisons possibles de paramètres d'un modèle (suivant les paramètres et leurs valeurs choisies) pour choisir les meilleurs paramètres pour le modèle. Cette méthode sous-entend que mettre la totalité des paramètres d'un modèle est judicieux pour ne pas se limiter aux paramètres prédominants uniquement. Cependant, le principal inconvénient de cette fonction est son temps d'exécution. Il est donc fortement déconseillé de tester plus de 3 paramètres lors de son utilisation.

8.2.10.Evaluation du modèle

Pour obtenir le score et définir la performance d'un modèle, plusieurs fonctions sont disponibles :

- ConfusionMatrix : En utilisant le module « ConfusionMatrixDisplay » de « sklearn.metrics », il est possible de créer une matrice de confusion. Celle-ci permet d'avoir un rendu visuel des prédictions du modèle. L'axe des abscisses représente les labels prédits et l'axe des ordonnées les véritables labels. Cette opération est à effectuer sur les données d'entraînement et sur les données de test. Chaque "bonne" classification sera en diagonale et représentée par un label prédit égal au véritable label. L'exemple ci-dessous ne comprend que 25 tags (classes) pour être plus visible. Une bonne classification pour les données d'entrainements et de tests sont observables. De plus, cet affichage permet de regarder les tags étant mal entraînés et mal prédits. À la suite de plus profondes analyses, les erreurs de classification sont dues à un « dataset » d'entraînement trop petit.

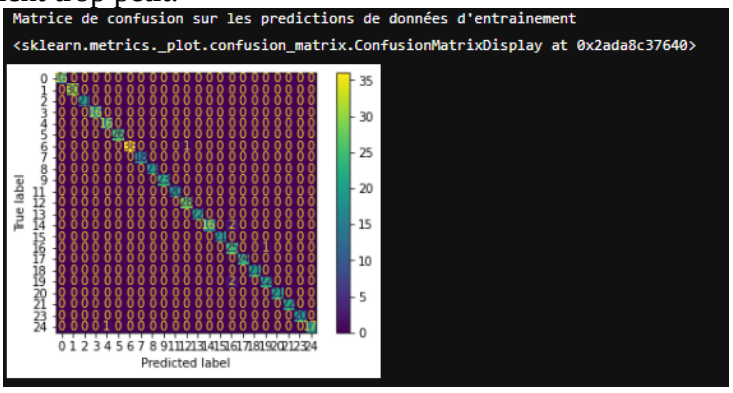


Figure 23 : Matrice de confusion réduit à 25 tags avec les données d'entrainements

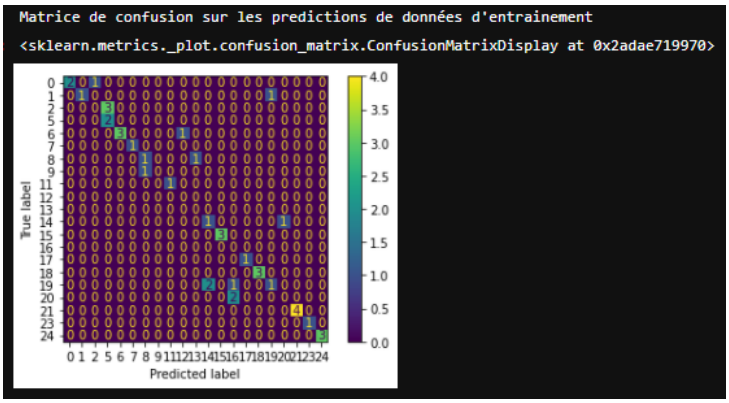


Figure 24 : Matrice de confusion réduit à 25 tags avec les données de tests

Cet affichage permet une analyse macroscopique de la performance d'un algorithme. Cependant, pour comparer deux d'entre eux, il est plus judicieux d'afficher le score des modèles avec la fonction « .best_score ».

Voici les deux meilleurs scores pour les modèles concernant ISEN et IA-MICROSOFT avec le SGDClassifier :

```
les meilleurs paramètres ISEN sont : {'alpha': 0.0001, 'loss': 'log', 'penalty': 'l2'} avec un score de 0.8625745950554133
les meilleurs paramètres IA MICROSOFT sont : {'alpha': 0.01, 'loss': 'modified_huber', 'penalty': 'l2'} avec un score de 0.7869645690111052
```

Figure 25 : Affichage des performances des modèles après GridSearchCV

Pour information, voici les performances des autres algorithmes testés :

LinearSVC

```
les meilleurs paramètres ISEN sont : {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'} avec un score de 0.724087591240876
les meilleurs paramètres IA MICROSOFT sont : {'C': 10, 'gamma': 0.1, 'kernel': 'rbf'} avec un score de 0.5924468922108576
```

Figure 26 : Affichage des performances des modèles après LinearSVC

Random Forest

```
les meilleurs paramètres ISEN sont : {'max_leaf_nodes': 19, 'n_estimators': 25} avec un score de 0.4613754181669603
les meilleurs paramètres IA MICROSOFT sont : {'max_leaf_nodes': 19, 'n_estimators': 17} avec un score de 0.4447785934170021
```

Figure 27 : Affichage des performances des modèles après Random Forest

8.2.11.Fonction predict

Cette dernière fonction regroupe et utilise les précédentes.
Celle-ci prend en entrée une phrase de l'utilisateur, et possède déjà en mémoire un modèle entraîné, l'encodeur créé par la vectorisation du « dataset » d'entrainement, le dictionnaire associant le numéro prédit par le modèle au tag ainsi que le dictionnaire associant le tag à la réponse à donner à l'utilisateur.
L'avantage de cette fonction est sa rapidité. En effet, en ne réentraînant pas le modèle, elle est mise en forme avec les fonctions « extraction_data_X_y », « collage » et « vectorizer » pour donner ensuite une prédiction.

```
phrase_test = "Quels sont les tarifs du cycle ingénieur?"
reponse = predict(phrase_test, model_sgd, encodeur_X_ISEN, num_tag_ISEN, dico_tag_responses_ISEN)
reponse
```

```
['Les trois années du ingénieur ont des frais de scolarité s'élevant à 8200 €/an. Au cours de l'année 2020-21, 86% des élèves de dernière année ont été exonérés des frais de scolarité puisque qu'ayant chois
i d'étudier en alternance. De plus, ils sont rémunérés par l'entreprise d'accueil. De plus, les élèves de l'ISEN sont éligibles aux bourses de l'enseignement supérieur relevant du ministère de l'éducation n
ationale et de l'enseignement supérieur et de la recherche.']
```

Figure 28 : Exemple d'utilisation de la fonction predict avant intégration

9.Partie Web

9.1.Flask

L'utilisation de Flask et sa mise en place sont ici expliquées et les différentes possibilités offertes évoquées. Après avoir installé Flask, l'importation du framework dans un fichier python nous permet de l'utiliser. Ce fichier sera considéré comme l'application Flask et sera donc celui à lancé pour faire fonctionner le serveur. Pour créer l'application, il suffit d'appeler une variable, comme visible sur la figure 29.

```
app = Flask(__name__)
```

Figure 29 : Lancement de Flask

Cette application sera automatiquement hébergée en local, par défaut sur l'adresse suivante : 127.0.0.1 :5000 .
Une fois l'application créée, une définition des routes qu'empruntera le serveur sera définie. La page de base est la route '/', elle permet par exemple de lancer automatiquement une fonction. De plus, dans le cadre d'un site web, ceci permet d'appeler la page html à l'aide de la fonction « render_template » comme observable dans la figure 30.

```
@app.route("/")
def index_get():
    return render_template("base.html")
```

Figure 30 : Appelle du fichier html

Le fonctionnement de Flask nécessite une organisation particulière. En effet, la création d'un dossier « templates » contenant tous les fichiers html, de même qu'un dossier « static » contenant les fichiers .css, .js ou les images utiles sur votre site sera nécessaire. La fonction « render template » ira chercher directement dans le dossier « templates » le fichier html et le lancera sur le serveur. Par la suite une écriture particulière, présentée par la figure 31, sera nécessaire dans le fichier html pour pouvoir récupérer les fichiers css, js et autres situé dans « static ».

```
href="{{ url_for('static', filename='style.css') }}">
```

Figure 31 : Apelle des fichiers du dossier «static»

Cette structure permet une organisation efficace tout en améliorant la rapidité du serveur. En effet, une autre méthode permet de garder tous les fichiers dans le même dossier. Toutefois après test, cette technique s'est révélée moins rapide lors de l'ouverture de l'application. L'organisation imposée par Flask a donc naturellement était l'option retenue.
À la suite de la présentation de Flask, son utilisation pour le chatbot va être à l'acmé de l'attention.

Flask permet le rajout de « route » pour faire tourner un programme, récupérer ou transférer des informations lors d'une requête, ce qui est l'objectif de ce projet. Dès lors, l'indication du type de requête attendue peut être mentionnée. Autre information importante : le nom du programme lancé lors de la boucle doit être le même que le nom de la « route », l'exemple de la figure 32 permet de mettre en exergue cette explication.

```
@app.route("/predictisen", methods=['GET', 'POST'])
def predictisen():
```

Figure 32: Nom de la route et de la fonction identique

Le framework contient une fonction permettant de récupérer les informations transmises en.json lors d'une requête, cela est fait à l'aide de la fonction « request.get_json.get(« Nom de l'information à récupérer ») ». Cette fonction est utilisée pour récupérer le message transmis par l'utilisateur lors de son envoi. Ce message récupéré, la fonction permettant de trouver une réponse à celui-ci est maintenant appelée, une fois récupérée, elle est mise dans un dictionnaire en réponse à « answer ». Pour retourner cette information au fichier JavaScript, un réencodage en .json est nécessaire. La fonction « jsonify » importée de Flask nous permet de faire ceci.
La fonction pour prédire est complète et peut être observée en figure 33, une deuxième fonction similaire sera nécessaire pour répondre selon le choix de l'école effectué par l'utilisateur.

```
def predictisen():
    text = request.get_json().get("message")


    response = get_response(text)
    message = {"answer":response}
    return jsonify(message)
```

Figure 33 : Fonction permettant de récupérer et renvoyer la réponse

L'application est prête à être lancée, ceci est réalisé dans le programme python à l'aide de la fonction présente en figure 34.

```
app.run(debug=True)
```

Figure 34 : Fonction permettant de lancer l'application

Une fois cette ligne incorporée à la fin du programme, il suffit de lancer ce script python. A partir du lancement du programme, différentes informations sur le serveur peuvent être récupérées telle que sa localisation, comme observable en figure 35.

```
D:\WPy64-3770\notebooks\Projet Chatbot>python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 276-205-277
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 35 : Lancement de l'application

9.2.HTML

La page html du projet doit rester simple et rapide, l'objectif étant de réaliser un support pour le chatbot. L'attention a donc été portée sur la visualisation de celui-ci et les améliorations possibles. Le contenu étant assez simple, le besoin de rajout d'un framework tel que bulma ou bootstrap ne s'est pas fait ressentir. La page est répartie en deux, la première partie comprend un « header » contenant le logo de l'ISEN, relié au site de l'ISEN Brest, pour décorer la page, un bouton pour télécharger le rapport et une icône renvoyant vers le github du projet, l'autre partie contenant une div « container » contenant la fenêtre du chatbot ainsi que son bouton pour le faire apparaitre, cette partie est elle-même séparée en quatre. Les 4 parties étant :
- Le haut du chatbot, contenant l'icône, le texte de présentation, et le bouton « réinitialiser »
- Un bloc blanc, où les futurs messages apparaîtront
- Le pied du chatbot, composé de la barre d'entrée et du bouton d'envoie
- La partie du bouton servant à faire apparaitre et disparaitre la petite fenêtre.

A la suite de l'utilisation de Flask, la page html est reliée aux fichiers .css et .js d'une façon propre au framework comme expliqué plus tôt.

9.3.CSS

Le fichier .css a pour but de rester simple et efficace comme pour son fichier html. L'objectif central est de styliser un maximum le fichier html tout en utilisant le minimum de classe. Dans un souci de simplification de modification des couleurs, des variables seront utilisées dans le fichier .css. La police utilisée sur la page est le Verdana, reconnue pour être la police la plus inclusive étant donné qu'elle est plus facilement lue par les personnes dyslexiques.

9.4.Rendu final de la page web

Une fois les fichiers combinés et le serveur lancé, le résultat semble satisfaisant, bien que non identique à la maquette. Le but recherché est obtenu, la page reste simple mais efficace et la fenêtre du chatbot possède les éléments attendus comme le montre la figure 36 ou l'annexe 2. La page web a été réalisé dans un objectif de responsivité optimale, sans point de rupture malgré l'absence de framework. Le chatbot, sur un ordinateur classique ou une tablette sera en bas à droite. Cependant, dans le cas d'un écran plus petit (comme celui d'un smartphone), il viendra se décaler sous le logo ISEN tout en gardant sa place à droite si possible et sa taille. Une amélioration potentielle serait l'ajout d'un point de rupture qui supposerait un affichage spécifiquement adapté pour passer sur un affichage téléphone comme imaginé dans la figure 37.

Figure 36 : Affichage finale du chatbot

Figure 37 : Affichage possible du chatbot sur mobile

9.5.JavaScript

Le fichier JavaScript permet le fonctionnement de la page web ainsi que les fonctions la reliant à la partie python du chatbot. Pour ne pas surcharger le html, le choix a été fait de repérer les éléments dans le fichier à l'aide de leur classe et non de leur rajouter un id. Pour les retrouver et pouvoir transformer la page web, la fonction document.querySelector(« nom de la classe ») a été utilisée comme le montre la figure 38.

```
this.args = {
    openButton: document.querySelector('.chatbox__button'),
    chatBox: document.querySelector('.chatbox__support'),
    sendButton: document.querySelector('.send__button'),
    refreshButton : document.querySelector('.refresh__button')
}
```

Figure 38 : Code utilisant la fonction ".querySelector()"

La première fonction utilisée est la fonction display, elle permet d'ajouter les évènements aux différents boutons et définir la touche « entrée » comme reliée à la même action que l'appui sur le bouton envoyer.

La deuxième fonction utilisée est la fonction appelée par le bouton pour afficher et masquer le chatbot, elle comprend deux utilités. Utilisée principalement pour afficher le chatbot, son utilité secondaire réside dans l'envoi dans un premier message proposant à l'utilisateur de se renseigner soit sur l'ISEN de Brest soit sur l'IA Microsoft.

Une fois que l'utilisateur a fait son choix, une fonction le sauvegardera et permettra de savoir quel algorithme utiliser pour chercher une réponse. Si l'utilisateur ne choisit pas d'école et envoie un message, un message lui sera renvoyé lui demandant de choisir une école avant de poser sa question.

Lors de l'appui sur le bouton « réinitialiser », la fonction « onRefreshButton » viendra supprimer l'historique de la conversation, que ce soit dans la mémoire du programme ou visuellement pour l'utilisateur, puis renverra le message demandant de choisir l'école sur laquelle il souhaite se renseigner. Selon les usages futurs, si l'historique souhaite être conservé, il suffirait de supprimer la boucle « while » de la figure 39.

```
while ( (i = this.messages.shift()) !== undefined ) {
}
```

Figure 39 : Boucle "while" permettant de supprimer l'historique

Une fonction permet l'ajout et l'affichage des nouveaux messages enregistrés. Les messages sont réaffichés et l'historique est conservé. Une fonction moins complexe aurait pu être effectuée sans l'historique sauvegardé, juste en affichant au fur et à mesure les messages. Cependant, dans le but d'améliorer l'algorithme et le set de données, pouvoir sauvegarder ce que l'utilisateur recherche, s'il le consent, à travers des cookies est intéressant.

Enfin, la fonction « onSendButton » intervient lorsque l'utilisateur appuie sur le bouton envoyer. Le programme commence par récupérer le message de l'utilisateur et vérifie que ce

n'est pas un message vide, il va ensuite utiliser la fonction « .push » pour sauvegarder le message envoyé. Ensuite, le programme, en fonction du choix de l'école, envoie le message via les routes flask à l'application et à l'algorithme qui va chercher une réponse à ce message. Une fois la réponse récupérée, le message est lui aussi enregistré comme réponse. Enfin, la fonction qui va afficher les messages est appelée, les messages sont affichés, et l'utilisateur est ramené au message le plus récent de la conversation.

10. Conclusion

Un chatbot fonctionnel répondant aux critères fixés. Voici l'aboutissement de ce projet résumé en une phrase. En effet, bien que perfectible, le chatbot est une réussite. Le projet a été mené à bien, avec les écarts de parcours inhérents à tout travail de fond. Ainsi, cette expérience a été valorisante tant pour les élèves qui ont acquis une expérience enrichissante pour leur avenir professionnel, que pour les deux écoles qui disposent désormais d'un chatbot prêt à être déployer.

L'intelligence artificielle présente au sein de ce chatbot a constitué un défi technologique important qui a nécessité une sincère motivation, une détermination incomparable et une implication conséquente des porteurs du projet. Les différents algorithmes possibles, étudiés au fur et à mesure de ce projet, ont été vecteurs d'une satisfaction considérable, d'autant plus qu'ils ont mené aux résultats attendus. Des étapes supplémentaires se sont ajoutées dans la partie IA comme l'augmentation des données, qui non planifiée s'est révélée être essentielle à la constitution d'un jeu de données assez ample permettant d'obtenir un algorithme contenant suffisamment d'informations. Une amélioration de ce jeu de donnée est d'ailleurs un des points clés dans la pérennisation de ce chatbot. Il faut maintenir les informations à jour et si possible les étendre. En effet, l'extension des données d'entrainement de l'algorithme est gage d'une performance accrue de ce dernier. Un autre axe d'amélioration peut se situer dans les réponses fournies à l'utilisateur. En effet, il n'y a pour l'instant qu'une réponse possible, mais avoir plusieurs réponses différentes, pour une même question, serait envisageable pour créer un chatbot plus « humain ».

Le site web a été l'autre grande partie de ce projet, elle a permis de découvrir de nouveaux outils comme le framework Flask, qui, bien que facile d'utilisation, a été complexe à comprendre. Le site web a été un défi constant dans la recherche d'optimisation de l'efficacité de sa responsivité. De plus, l'intégration d'algorithmes python avec le JavaScript a été également un concept enrichissant et important à découvrir. Plusieurs axes d'améliorations sont observables sur ce site web, le premier serait l'intégration sur un serveur distant, et non local. De plus, une version mobile pourrait être pensée, avec un point de rupture permettant une meilleure adaptation du site sur téléphone. Une version plein écran du chatbot pourrait également être envisagée.

D'autres améliorations sont également envisageables, la première serait d'intégrer un apprentissage par renforcement au chatbot. L'utilisateur pourrait indiquer quelle réponse il attendait si le chatbot n'a pas ou mal compris sa question. Une suggestion de question à poser pourrait également être affichée selon la question précédente de l'utilisateur, notamment à l'aide de l'élément « context » compris dans le jeu de donnée.

Les chatbots sont amenés à se développer et à s'établir comme norme dans un futur maintenant très proche. Ce projet a permis d'amener une compréhension sur leur fonctionnement et les différentes technologies qui y sont associées. Les entreprises investissent dans ces agents conversationnels mais jusqu'où cette idée peut-elle se développer? Quelle sera l'utilité des futurs chatbots et à quelle fin seront-ils utilisés?

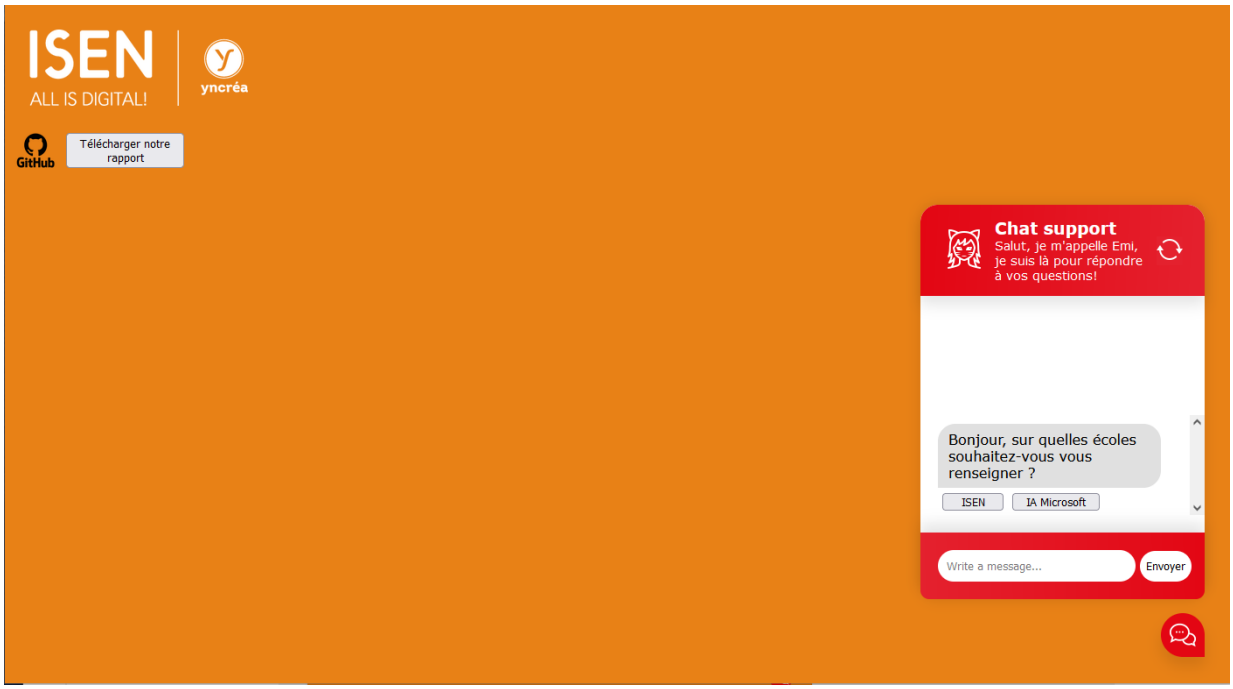
11.Webographie

https://www.youtube.com/c/MachineLearnia/videos
https://www.youtube.com/c/RanjanSharma/videos
https://www.youtube.com/c/AhmadBazzi/videos
https://dphi.tech/blog/tutorial-on-logistic-regression-using-python/
https://towardsdatascience.com/
https://maelfabien.github.io/papers/#
https://www.analyticsvidhya.com/blog/category/nlp/?utm_source=blog_navbar&utm_medium=_button
https://www.quennec.fr/trucs-astuces/langages/python
https://pythonnumericalmethods.berkeley.edu/notebooks/
https://stackoverflow.com/
https://pythonspot.com/save-a-dictionary-to-a-file/
https://moonbooks.org/Articles/Comment-sauvegarder-dans-un-fichier-un-modele-developpe-avec-scikit-learn-en-python-machine-learning-/
https://github.com/LueMar-R/chatbot_brief_simplon
https://pythonprogramming.net/tokenizing-words-sentences-nltk-tutorial/
https://scikit-learn.org/stable/index.html
https://intellipaat.com/community/9495/how-do-i-solve-overfitting-in-random-forest-of-python-sklearn
https://ichi.pro/fr/guide-du-debutant-sur-le-reglage-des-hyperparametres-de-foret-aleatoire-77596161963319
https://www.youtube.com/watch?v=-3UBmIXGHyc&t=651s&ab_channel=AIforyou-MorganGautherot
https://www.youtube.com/watch?v=a37BL0stIuM&t=1421s
https://www.youtube.com/watch?v=uD1gtcOxkWs&t=2043s
https://www.youtube.com/watch?v=y0CRhaWNpto
https://flask.palletsprojects.com/en/2.1.x/
http://www.xavierdupre.fr/app/ensae_teaching_cs/helpsphinx/notebooks/TD2A_eco_debuter_flask.html
https://github.com/Amaury-ISEN/Chatbot_Simplon
https://www.digitalocean.com/community/tutorials/how-to-make-a-web-application-using-flask-in-python-3-fr
https://www.oracle.com/fr/chatbots/what-is-a-chatbot/
https://www.monatourisme.fr/wp-content/uploads/2018/03/Note.pdf
https://www.flaticon.com/fr/
https://acubepi.github.io/IAP-chatbot/
https://blog.hubspot.fr/marketing/meilleurs-chatbots

12.Annexes

12.1.Annexe 1 : Maquette du site web

12.2.Annexe 2 : Site actuel



12.3.Annexe 3 : Serveur Discord utilisé pour le projet