

Document préparé par

Ameur Kais

[HTTPS://WWW.LINKEDIN.COM/IN/KAISAMEUR/](https://www.linkedin.com/in/kaisameur/)

**Mise en place d'un chatbot pour l'assistance à
l'optimisation des réseaux radio**

Blank...

TABLE DES MATIÈRES

Introduction Générale	2
1 Contexte général	4
Introduction	5
I Objectifs et solution proposée	5
I.1 Objectifs	5
I.2 Solution proposée	6
II Intelligence artificielle et Chatbots	8
II.1 Intelligence Artificielle : Machine Learning et Traite- ment Automatique du Langage Naturel	8
II.2 Les Chatbots	9
II.3 FrameWork Rasa	11
III Optimisation KPIs réseaux radio	15
IV Outils et technologies	16
Conclusion	16
2 Préparation des données et conception	17
Introduction	18
I Source des données	18
I.1 Présentation des données	19
I.2 Intégration des données sous mongoDB	23

I.3	Chargement des données par Rasa	25
II	Préparation des données	27
II.1	Suppression des caractéristiques inutiles	27
II.2	Nétoyage des données	28
II.3	Transformation des données	29
III	Conception des modules	30
III.1	Conception du module : Accès aux informations réseaux (Module 1)	30
III.2	Conception du module : Diagnostique et recomman- dations (Module 2)	31
III.3	Conception du module : Analyse et prédiction de per- formance (Module 3)	32
	Conclusion	33
3	Déploiement et évaluation de la solution	35
	Introduction	36
I	Connexion à la base de données MongoDB de Rasa	37
I.1	Installation des objets nécessaires	37
I.2	Configuration de la base de donnée	38
II	Interfaçage avec une page web	39
III	Test et résultats du chatbot	40
III.1	résultats du chatbot	40
III.2	Évaluation des performances du chatbot	45
	Conclusion	47
	Conclusion Générale	48
	Bibliographie	49
	Annexe	50
I	LTE KPI-Accessability	50
II	Initiation à Rasa	52

LISTE DES FIGURES

1.1	Architecture de Rasa [1]	12
1.2	Processus d'optimisation des KPIs réseaux radio	15
2.1	Source des données	19
2.2	Fichier des Features	20
2.3	Fichier d'informations des sites radio	21
2.4	Fichier de trafic des sites radio	22
2.5	Fichier des KPIs des sites	23
2.6	Intégration des données sous mongoDB	24
2.7	Insertion des données du mongoDB	25
2.8	Extraction des données du mongoDB	25
2.9	Suppression des caractéristiques inutiles	27
2.10	Traitement des valeurs nuls	28
2.11	Transformation de données	29
2.12	Processus de traitement du module 1 : accès informations réseaux	31
2.13	Processus de traitement du module 2 : diagnostique et recom- mandations	32
2.14	Processus de traitement du module 3 : Analyse et prédiction de performance	33

LISTE DES FIGURES

3.1	Execution de commande d'installation de bibliothèque "py-mongo"	37
3.2	Configuration du endpoints	38
3.3	Credentials.yml	39
3.4	Configuration de credentials.yml	40
3.5	Informations sur les sites radio	41
3.6	Informations sur les Features radio	41
3.7	Présentation des sites avec des KPIs dégradés	42
3.8	Recommandation pour les sites avec des KPIs dégradés	43
3.9	Détection des sites radio <i>Top Worst</i>	44
3.10	Prédiction du trafic des sites radio	44
3.11	Histogramme des intents	45
3.12	Matrice de confusion des <i>intents</i>	46
3.13	Matrice de confusion de flux de conversation	47

LISTE DES TABLEAUX

1.1 Comparaison de quelques Frameworks 7

2.1 Informations sur toutes les collections de la table 26

INTRODUCTION GÉNÉRALE

L'optimisation des réseaux radio constitue un enjeu clé dans le domaine des télécommunications afin de garantir des communications sans fil efficaces et de haute qualité. Avec l'évolution rapide des technologies et la prolifération des appareils connectés, les réseaux radio sont devenus de plus en plus complexes, nécessitant des solutions innovantes pour les optimiser et assurer une expérience utilisateur optimale.

Dans le cadre de ce projet, nous nous concentrons sur l'implémentation d'une solution basée sur un chatbot de type Rasa pour l'assistance à l'optimisation des réseaux radio. Rasa est un framework open-source qui permet de créer des chatbots intelligents et conversationnels en utilisant des techniques d'intelligence artificielle et du Traitement Automatique du Langage Naturel (TALN).

Notre solution chatbot, reposant sur le framework Rasa, vise à offrir aux ingénieurs et techniciens une interface interactive et conviviale pour l'assistance à l'optimisation de leurs réseaux radio. Grâce à l'utilisation du TALN, le chatbot est capable de comprendre le langage naturel des utilisateurs et d'interagir avec eux de manière conversationnelle.

Le chatbot de notre solution exploite une base de connaissances approfondie sur l'optimisation des réseaux radio, alimentée par des données techniques, des meilleures pratiques et des modèles prédictifs. Il peut répondre

aux questions des utilisateurs, leur fournir des informations détaillées sur les problèmes de réseau, les guider à travers les procédures d'optimisation et même proposer des recommandations spécifiques adaptées à leur contexte aussi que de prédire les KPIs réseaux.

La flexibilité et la modularité offertes par le framework Rasa permettent d'intégrer facilement le chatbot dans les systèmes existants de Tunisie Télécom. Le chatbot peut être accessible via réseau Intranet et peut avoir accès aux bases de données, aux outils d'analyse et à d'autres sources de données pertinentes pour fournir des informations en temps réel et des suggestions d'optimisation précises.

Ce rapport présente une étude complète sur l'optimisation des performances des réseaux radio en utilisant une solution basée sur l'intelligence artificielle et les chatbots. En utilisant la méthodologie CRISP-DM (Cross-Industry Standard Process for Data Mining), nous avons abordé chaque étape de manière systématique et structurée. Dans le chapitre "Contexte général", nous examinons l'organisme d'accueil et les objectifs du projet, en mettant en évidence l'importance de l'intelligence artificielle, y compris l'apprentissage automatique et le traitement automatique du langage naturel, ainsi que les avantages des chatbots. Le chapitre "Préparation des données et conception" aborde la collecte et la préparation des données, ainsi que la conception des modules du système, tels que l'accès aux informations réseau, le diagnostic et les recommandations, et l'analyse et la prédiction des performances. Dans le chapitre "Déploiement et évaluation de la solution", nous présentons la configuration du framework Rasa, la connexion à la base de données MongoDB, ainsi que des tests et des résultats du chatbot.

Enfin, en combinant l'intelligence artificielle, le traitement du langage naturel et l'expertise en télécommunications, notre solution offre un outil puissant pour améliorer les performances des réseaux radio, résoudre les problèmes plus rapidement et offrir une expérience utilisateur optimale.

CHAPITRE 1

CONTEXTE GÉNÉRAL

Introduction	5
I Objectifs et solution proposée	5
I.1 Objectifs	5
I.2 Solution proposée	6
II Intelligence artificielle et Chatbots	8
II.1 Intelligence Artificielle : Machine Learning et Traite- ment Automatique du Langage Naturel	8
II.2 Les Chatbots	9
II.3 FrameWork Rasa	11
III Optimisation KPIs réseaux radio	15
IV Outils et technologies	16
Conclusion	16

Introduction

L'objectif de cette partie chapitre est d'explorer les concepts fondamentaux de l'optimisation radio dans les réseaux de télécommunication, ainsi que les principes de base des chatbots.

En comprenant ces notions essentielles, nous serons en mesure de concevoir une solution efficace et pertinente pour notre projet.

I Objectifs et solution proposée

I.1 Objectifs

De nombreuses entreprises de télécommunications ont mis en place des chatbots qui interagissent avec les clients pour améliorer le service, simplifier les processus et améliorer l'expérience utilisateur.

Le défi du projet est de créer un chatbot fiable et précis qui peut communiquer naturellement avec les professionnels de Tunisie Télécom et offrir des recommandations sur mesure pour l'optimisation des KPI radio qui constitue un défi majeur pour les opérateurs de télécommunications afin d'offrir des services de haute qualité aux utilisateurs finaux.

Ci après, nous présentons les objectifs de la solution :

- Objectif 1 : développer un module intitulé *Accès aux informations réseaux* pour faciliter l'accès à la base de connaissances et aux meilleures pratiques pour optimiser les réseaux radiophoniques.
- Objectif 2 : développer un module intitulé *Diagnostic et recommandations* pour aider à diagnostiquer les causes de dégradation des KPI radio.
- Objectif 3 : développer un module intitulé *Analyse et prédiction de performance* pour assurer l'analyse et prédiction des performance des

réseaux à l'aide d'algorithmes d'apprentissage automatique.

I.2 Solution proposée

Pour l'assistance à l'optimisation des réseaux radio, la solution proposée est d'utiliser un chatbot intelligent basé sur le framework Rasa. Notre assistant virtuel sera configuré pour communiquer avec des experts grâce à une interface de clavardage.

Ainsi, le chatbot sera configuré pour répondre à trois types de besoins, à savoir :

- Module d'assistance par des informations sur l'architecture et paramètres des réseaux radio existants
- Module d'assistance au diagnostic des problèmes de performances réseaux et recommandations des meilleures solutions et actions à entreprendre
- Module d'assistance par la classification des noeuds des réseaux radio selon leur rentabilité et qualité de service ainsi que la prédiction de quelques KPIs comme le trafic

Le tableau 1.1 présente certains des frameworks les plus populaires pour le développement de chatbots. Il inclut des informations sur la gratuité logiciel, personnalisation, le déploiement sur site, l'interopérabilité et la qualité de la documentation.

Feature	Rasa	Dialogflow	IBM Watson Assistant
Open Source	Oui	Non	Non
Personnalisation	Élevé	Limité	Limité
Déploiement sur site	Oui	Non	Oui

Facilité d'accès aux logiciels et à l'information	Élevé	Limité	Limité
Aide communautaire	Forum et documentation communautaires solides et actifs	Soutien et documentation communautaires limités	Soutien et documentation communautaires limités

Table 1.1: Comparaison de quelques Frameworks

Rasa est largement reconnu comme l'un des meilleurs choix pour le développement de chatbots, notamment dans le cadre des projets nécessitant une personnalisation et une flexibilité approfondies. Le tableau comparatif [1.1](#) ci-dessus met en évidence les principales raisons qui soutiennent la sélection de Rasa en tant que solution privilégiée pour le développement d'un chatbot open-source.

En effet, RASA est un Framework d'IA ouvert qui peut être utilisé pour développer des chatbots fonctionnant avec l'IA. Il est flexible, évolutif et transparent et fournit une documentation claire avec le soutien de la communauté. RASA suit une architecture de micro service modulaire et extensible qui s'insère bien dans un scénario de développement logiciel typique.

RASA peut être facilement intégré avec des services de messagerie populaires via des canaux API REST tels que Whatsapp, Slack, Telegram, Facebook Messenger et Rocket.Chat. Il peut également être hébergé sur un site dédié.

RASA utilise un TALN avancé appelé DIET pour comprendre et répondre aux différentes entrées de manière contextuelle, ce qui en fait un assistant de niveau 3. De plus, RASA offre des capacités de personnalisation en termes de

composants et de politiques de pipeline, offrant aux entreprises la possibilité d'adapter l'assistant virtuel à leurs besoins spécifiques[1].

II Intelligence artificielle et Chatbots

II.1 Intelligence Artificielle : Machine Learning et Traitement Automatique du Langage Naturel

L'intelligence artificielle (IA) est la création des systèmes informatiques capables d'effectuer des tâches qui exigent habituellement l'intelligence humaine, comme la résolution de problèmes, l'apprentissage et le jugement.

L'utilisation de l'intelligence artificielle peut transformer totalement notre vie personnelle et professionnelle. Des systèmes d'intelligence artificielle peuvent être créés pour analyser des quantités massives de données, repérer des tendances et faire des prédictions ou des jugements à partir de ces données à l'aide des algorithmes sophistiqués, de l'apprentissage automatique et de l'apprentissage profond.

Une branche de l'intelligence artificielle appelée "Machine Learning" (M.L.) se concentre sur la création de modèles statistiques et d'algorithmes qui permettent aux ordinateurs d'apprendre de leurs performances passées sans avoir à être explicitement programmé. En plus de l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage renforcé sont inclus dans les diverses catégories d'algorithmes d'apprentissage automatique.

L'apprentissage automatique est de plus en plus crucial dans une variété d'industries et d'applications, des soins de santé et de la finance aux voitures autonomes et aux systèmes de recommandation, grâce à l'augmentation exponentielle de grandes données et des améliorations de la puissance de traitement. Les séries chronologiques désignent une séquence de points de données classés chronologiquement, enregistrant des observations ou des mesures enregistrées à intervalles réguliers au fil du temps. La prévision de séries

chronologiques consiste à prédire les valeurs futures d'une série chronologique en se fondant sur des observations antérieures.

Un sous-domaine de l'intelligence artificielle connu comme "Traitement Automatique du Langage Naturel" (TALN) est concerné par la façon dont les ordinateurs et le langage humain interagissent. Il consiste à créer des algorithmes et des modèles qui donnent aux ordinateurs la capacité de déchiffrer, de comprendre et de produire du texte ou de la voix dans un langage naturel.

Le TALN est un domaine interdisciplinaire qui utilise des méthodes de psychologie cognitive, d'informatique, de linguistique et de mathématiques. La classification des textes, l'analyse des sentiments, la reconnaissance des entités nommées, la traduction automatique et la reconnaissance vocale sont quelques-unes des principales tâches du TALN.

Le TALN a plusieurs utilisations, y compris dans les chatbots, les moteurs de recherche, les logiciels de traductions linguistiques, et les assistants virtuels tels que Siri et Alexa.

II.2 Les Chatbots

Les chatbots, également connus sous le nom d'agents conversationnels, sont des programmes informatiques conçus pour simuler des conversations humaines. Grâce à l'intelligence artificielle et au traitement du langage naturel, ils peuvent comprendre et répondre aux requêtes des utilisateurs de manière conversationnelle. Les chatbots se sont révélés être une technologie prometteuse dans divers domaines tels que la santé, l'éducation, le service clientèle et bien d'autres. Ce chapitre explore les concepts fondamentaux des chatbots, leurs applications pratiques, les défis auxquels ils font face et leur potentiel futur.

On distingue généralement trois types de chatbots : les chatbots basés sur des règles, les chatbots basés sur l'intelligence artificielle et les chatbots hybrides. Les chatbots basés sur des règles utilisent des ensembles prédéfinis de règles et de réponses pour interagir avec les utilisateurs. Les chatbots

basés sur l'intelligence artificielle utilisent des techniques d'apprentissage automatique pour comprendre et générer des réponses. Les chatbots hybrides combinent les deux approches pour obtenir un meilleur équilibre entre flexibilité et précision.

Les chatbots comprennent plusieurs composants clés. L'interface utilisateur permet aux utilisateurs de communiquer avec le chatbot de manière conviviale, via des messages texte ou des interfaces vocales. Le moteur de traitement du langage naturel est responsable de la compréhension des requêtes des utilisateurs et de la génération de réponses appropriées. Enfin, la base de connaissances est une ressource qui stocke les informations et les réponses préalablement configurées ou apprises par le chatbot.

Les chatbots ont connu une adoption généralisée dans de nombreux domaines, offrant des solutions variées et adaptées aux besoins spécifiques. Dans le secteur de la santé, ils sont utilisés pour le suivi des patients, facilitant la collecte d'informations sur leurs symptômes, les rappels de médicaments et la fourniture de conseils médicaux de base. Dans le domaine de l'éducation, les chatbots permettent de créer des environnements d'apprentissage virtuels interactifs, offrant aux étudiants une assistance dans la compréhension des concepts, la réponse à leurs questions et l'accès à des ressources supplémentaires.

Les chatbots sont également utilisés dans le service clientèle, en répondant automatiquement aux questions fréquentes, en offrant une assistance en temps réel et en orientant les clients vers les ressources appropriées.

Dans le secteur du commerce électronique, ils facilitent la recherche de produits, l'achat et le suivi des commandes, tandis que dans le tourisme, ils fournissent des informations sur les destinations, les réservations et des recommandations personnalisées.

Enfin, les chatbots sont utilisés dans le support technique, permettant aux utilisateurs de résoudre rapidement des problèmes techniques courants sans intervention humaine. Leur polyvalence et leur capacité à automatiser les tâches routinières contribuent à simplifier les processus, à améliorer

l'efficacité et à offrir une expérience utilisateur améliorée dans divers domaines.

Les chatbots font face à plusieurs défis et considérations. La conception centrée sur l'utilisateur est essentielle pour créer des chatbots conviviaux et capables de fournir des réponses pertinentes et personnalisées. Le traitement du langage naturel représente un défi majeur, nécessitant des avancées dans l'intelligence artificielle et l'apprentissage automatique. La confidentialité et la sécurité des données sont des préoccupations majeures, exigeant des mesures de protection robustes. Les considérations éthiques doivent être prises en compte, incluant la responsabilité, la transparence, l'équité et la prévention des biais. En surmontant ces défis, les chatbots peuvent évoluer vers des interactions plus intelligentes et personnalisées.

L'intégration de technologies émergentes comme la réalité virtuelle et la réalité augmentée permettra des expériences conversationnelles encore plus immersives et interactives. Les chatbots deviendront de plus en plus intelligents, capables de s'adapter aux préférences et aux besoins des utilisateurs, offrant ainsi des expériences personnalisées. Une évolution importante sera le développement de chatbots empathiques, capables de reconnaître et de répondre aux émotions des utilisateurs, créant ainsi des interactions plus humaines et empathiques. La sécurité et la confidentialité des données resteront une priorité, nécessitant des mesures de protection renforcées pour assurer la confiance des utilisateurs.

II.3 Framework Rasa

L'architecture de Rasa se compose de quatre composantes principales [1] comme le montre la figure 1.1 :

- Rasa Natural Language Understanding (NLU) composante est en charge de déchiffrer les entrées des utilisateurs et de collecter des données structurées à partir de lui. Il traite les entrées des utilisateurs en utilisant

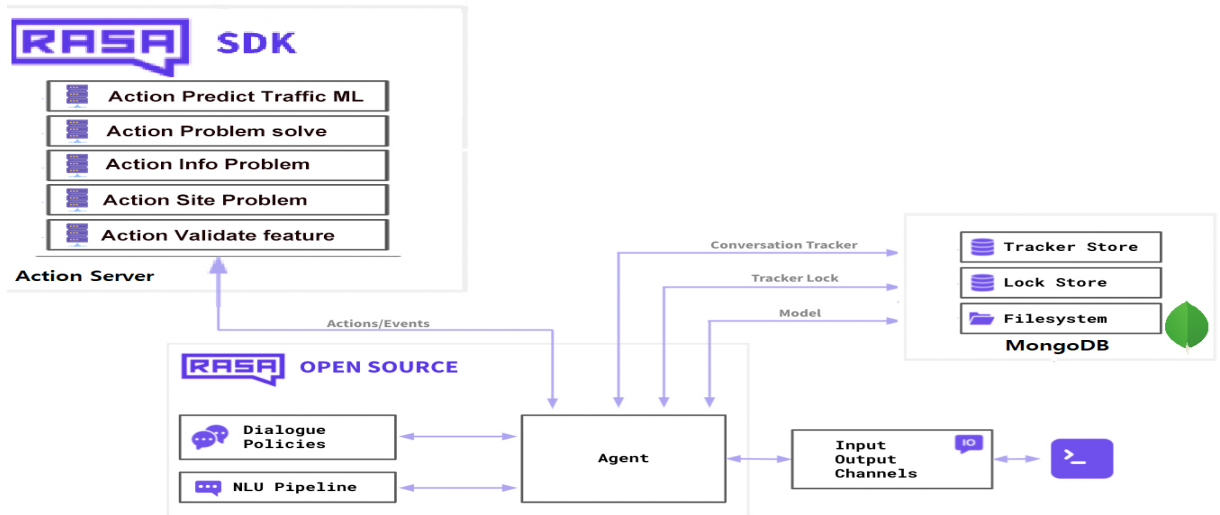


Figure 1.1: Architecture de Rasa [1]

des modèles de machine learning pour extraire les entités et l'intention. Il est composé de :

1. **Tokenizer** : La tokenisation dans NLP est le processus de briser un flux de données textuelles en mots, termes, symboles ou autres éléments importants appelés tokens comme `WhitespaceTokenizer` qui sépare le texte basé sur les espaces.
2. **Featurizer** : est un élément qui convertit des données d'entrée non traitées en fonctionnalités numériques qu'un modèle de machine learning peut utiliser. Il est essentiel pour analyser les entrées des utilisateurs et obtenir des données pertinentes pour la catégorisation des entités et des intentions. Citons un exemple des Featurizers utilisés, `RegexFeaturizer` qui extrait le texte en utilisant les expressions régulières, `CountVectorsFeaturizer` qui convertit le texte en un sac de mots représentation en comptant la fréquence de chaque mot et d'autres.
3. **Classifier** : En utilisant des caractéristiques numériques produites

par le générateur de fonctionnalités, un classificateur est un modèle de machine learning qui prédit le but et les entités du message d'un utilisateur. Il apprend à partir de données de formation étiquetées et utilise des algorithmes tels que SVM, softmax et des transformateurs pour générer des prédictions sur des données fraîches tels que DIET Classifier qui est capable de gérer la catégorisation des tâches de reconnaissance d'intention et d'entité simultanément en utilisant l'apprentissage multitâche qui se maine sur 2 Featurizer, dense featurizer qui transforme le texte en vecteurs denses de longueur fixe capables de capturer des informations sémantiques complexes pour classer les intentions et reconnaître les entités et sparse featurizer qui génère des vecteurs de caractéristiques épars en codant la présence ou l'absence de mots ou de n-grammes spécifiques, qui peut être utilisés pour la classification des intentions et les tâches de reconnaissance des entités..

- Rasa Core [1] est responsable de gérer le déroulement de la conversation et de sélectionner la marche à suivre appropriée. Basé sur l'entrée des utilisateurs et le contexte de conversation, il utilise des algorithmes d'apprentissage automatique pour prédire l'action future.
- Rasa action server [1] qui exécute des actions personnalisées pour un assistant conversationnel Rasa Open Source. Il contient aussi les NLUs ajoutés spécifiquement pour notre base de données tels que :
 - La correspondance la plus étroite avec un mot saisi est trouvée à partir d'une liste de termes acceptables par autocorrection lors de l'exécution d'une opération d'autocorrection. Le mot ayant le plus haut score de similarité au-dessus d'un seuil prédéterminé est choisi à l'aide d'une métrique de similarité basée sur des bigrammes pour comparer les scores de similarité de différents mots. Différents modes de recherche sont pris en charge par la fonction,

qui renvoie le mot corrigé ou "none" si aucune correspondance acceptable n'est découverte.

- La fonction *difflib.get_close_matches* utilise l'algorithme Ratcliff/Obershelp afin de calculer la similitude entre les chaînes. Cet algorithme utilise une combinaison de techniques d'appariement de séquences et de sous-séquences communes plus longues pour déterminer le score de similarité. Il compare le mot d'entrée à chaque mot candidat, en assignant une valeur en fonction du nombre et de la longueur des sous-chaînes correspondantes. La fonction retourne ensuite une liste des mots les mieux adaptés au-delà d'une limite définie, offrant une approche flexible pour l'appariement approximatif de chaînes et les tâches correctrices d'orthographe.
- Le nltk SentimentIntensityAnalyzer est un puissant outil d'analyse des émotions dans le traitement du langage naturel. Il fournit une technique rapide et précise pour déterminer la polarité globale de sentiment d'une phrase ou d'un document en attribuant des scores de sentiment au texte en évaluant l'intensité des termes émotionnels positifs, négatifs ou neutres.
- *Rasa input and output channels* [1] sont l'interface entre le modèle conversationnel IA Rasa et les plates-formes de messagerie externe (ex : Slack, Facebook Messenger, Twilio). Ils nous permettent de recevoir des messages des utilisateurs et d'envoyer des réponses à l'utilisateur par l'intermédiaire d'une plate-forme spécifique.

Rasa est un cadre polyvalent qui, en fonction du cas d'utilisation et de l'environnement de déploiement, peut être appliqué de diverses manières. Il peut être configuré pour utiliser différents modèles de machine learning et couplé à d'autres programmes et services, y compris la messagerie et les systèmes de bases de données. Lors de ce projet, nous avons utilisé la version

3.9.13 de Python et la version 22.11.1 de Anaconda.

III Optimisation KPIs réseaux radio

L'optimisation des KPIs et des paramètres des réseaux radio, tels que l'accessibilité, la retentabilité, l'intégrité et la disponibilité, est une étape cruciale pour garantir des performances optimales et une expérience utilisateur satisfaisante. Les opérateurs de télécommunications mettent en œuvre différentes techniques et stratégies pour améliorer ces aspects clés du réseau.

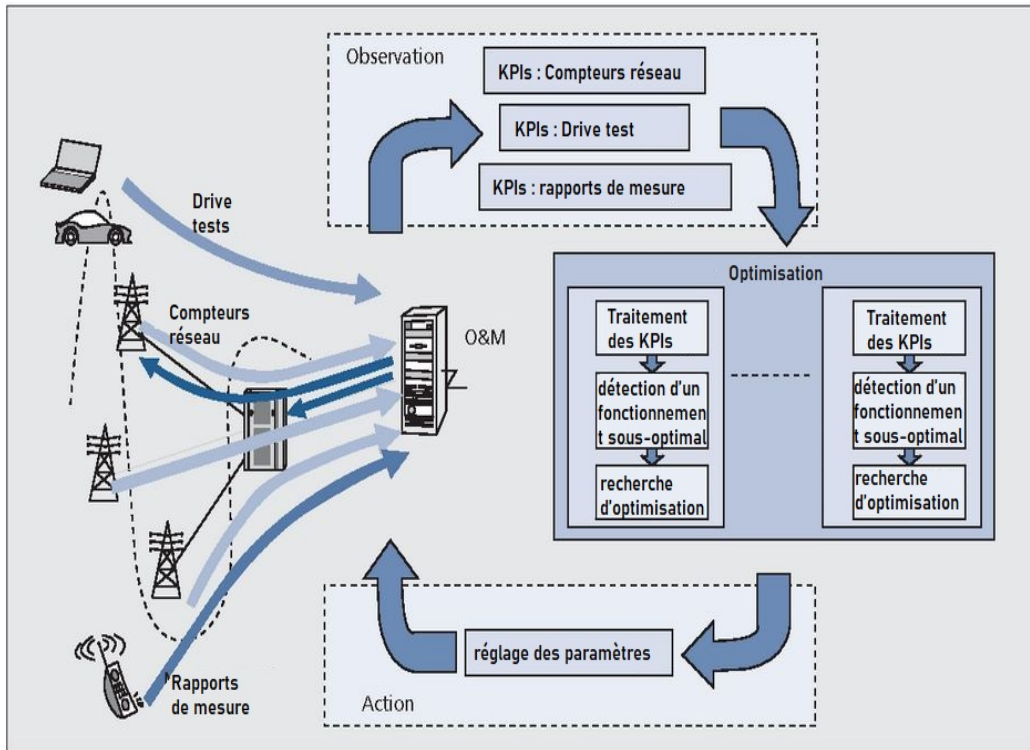


Figure 1.2: Processus d'optimisation des KPIs réseaux radio

L'optimisation des KPIs et des paramètres des réseaux radio est un processus continu qui implique une analyse régulière des performances, des

ajustements des configurations et une évaluation des nouvelles technologies et techniques comme indiqué dans la figure 1.2.

IV Outils et technologies

Notre projet a été mis en utilisant le Framework Rasa qui est une plateforme open-source pour construire des applications conversationnelles d'IA, comme les chatbots. Il fournit un ensemble complet d'outils pour développer et former des modèles conversationnels personnalisés, y compris le TALN et des algorithmes d'apprentissage automatique (ML). Rasa est conçu pour permettre aux développeurs de construire facilement des systèmes d'IA conversationnels hautement interactifs, personnalisés et capables de gérer des tâches complexes.

Avec le *FrameWork* RASA nous avons utilisés les outils Python[2], Visual Studio[3] et MongoDB[4].

Conclusion

Ce chapitre introductif a posé les bases nécessaires pour comprendre le contexte du projet d'assistance à l'optimisation radio par un chatbot. En combinant les connaissances de l'optimisation radio avec les capacités des chatbots, nous pourrons développer une solution innovante qui automatisera certaines tâches, améliorera l'efficacité opérationnelle et contribuera à l'optimisation des performances des réseaux.

Dans les chapitres suivants, nous approfondirons davantage les aspects techniques, en mettant l'accent sur la conception et le développement du chatbot Rasa spécialisé dans l'assistance à l'optimisation radio.

CHAPITRE 2

PRÉPARATION DES DONNÉES ET CONCEPTION

Introduction	18
I Source des données	18
I.1 Présentation des données	19
I.2 Intégration des données sous mongoDB	23
I.3 Chargement des données par Rasa	25
II Préparation des données	27
II.1 Suppression des caractéristiques inutiles	27
II.2 Néttoyage des données	28
II.3 Transformation des données	29
III Conception des modules	30
III.1 Conception du module : Accès aux informations réseaux (Module 1)	30
III.2 Conception du module : Diagnostique et recomman- dations (Module 2)	31
III.3 Conception du module : Analyse et prédiction de per- formance (Module 3)	32
Conclusion	33

Introduction

Le développement d'un chatbot performant et capable de fournir des réponses précises et pertinentes nécessite une préparation minutieuse des données ainsi qu'une conception soigneusement réfléchie. Dans ce chapitre, nous abordons l'importance cruciale de la préparation des données et de la conception dans le processus de développement d'un chatbot basé sur la plateforme Rasa.

Une conception solide garantit que le chatbot peut gérer les interactions complexes, maintenir un contexte cohérent et fournir des réponses adaptées aux besoins spécifiques des utilisateurs.

En utilisant les bonnes pratiques de préparation des données et de conception, nous pouvons développer un chatbot Rasa puissant et adapté aux besoins spécifiques de notre projet. Ce chapitre fournit les bases nécessaires pour créer une base solide pour notre chatbot.

I Source des données

Les fichiers Excel sont créés à partir des données extraites et agrégées de la base de données ENM/OSS (Enhanced Network Management / Operations Support System), qui est une plateforme utilisée par Tunisie Télécom pour gérer et contrôler efficacement les réseaux [2.1](#). Au cours de cette procédure, les données pertinentes sont recueillies à partir de différents sources exportées dans des fichiers Excel/CSV.

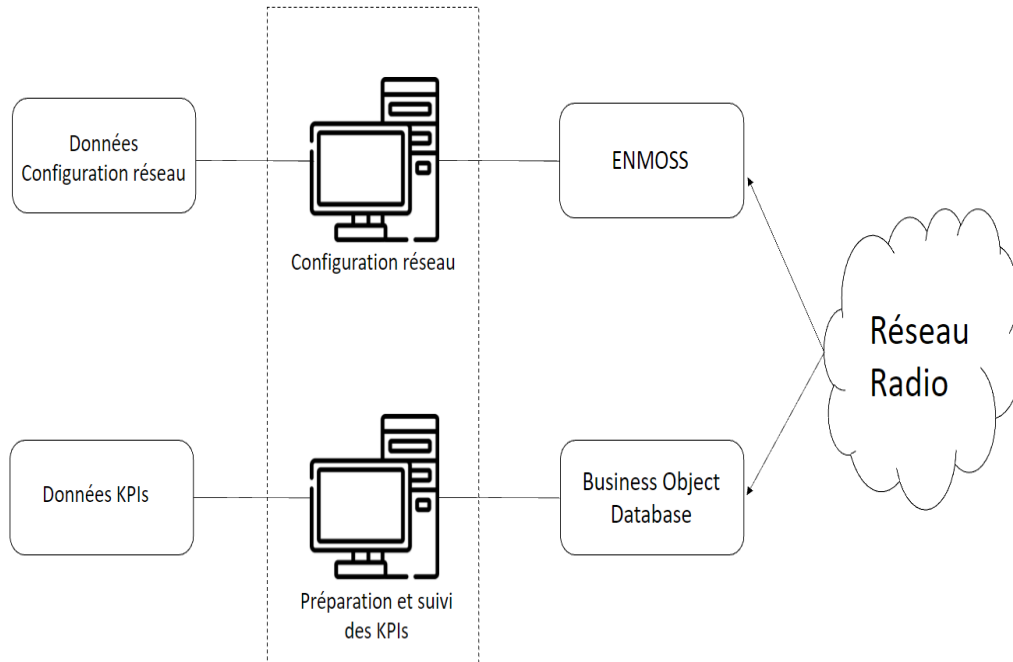


Figure 2.1: Source des données

I.1 Présentation des données

Ci-après, nous présentons les différents fichiers de données utilisées :

- Les features [2.2](#) :

Les features, sont des fonctionnalités spécifiques d'un réseau radio qui peuvent être activées ou désactivées en fonction des besoins. Les features sont souvent liées aux normes et aux technologies spécifiques utilisées dans le réseau, comme la 2G, la 3G, la 4G ou la 5G.

CHAPITRE 2. PRÉPARATION DES DONNÉES ET CONCEPTION

	A	B	C	D	E	F	G	H
	Feature Name	Feature Identity	Value Package Name	Value Package Identity	License Control MO DU radio Node, OptionalFeatureLicense=	License Control MO Baseband-based Node, FeatureState=	featureState Recommended Value	Comments
1								
2	13-18 Cell Support	FAJ 121 4242	Large eNodeB	FAJ 801 0403	Support18Cells	CXC4011917	SCENARIO_DEPENDENT	Prerequisites: 6 Cell Support (FAJ 121 1821) 7-12 Cell Support (FAJ 121 3020) Depends on customer needs.
3	19-24 Cell Support	FAJ 121 4426	Large eNodeB	FAJ 801 0403	Support24Cells	CXC4011974	SCENARIO_DEPENDENT	Prerequisites: 6 Cell Support (FAJ 121 1821) 7-12 Cell Support (FAJ 121 3020) 13-18 Cell Support (FAJ 121 4242) Depends on customer needs.
4	256 QAM Downlink	FAJ 121 4422	LTE Base Package	FAJ 801 0400	DI256Qam	CXC4011969	ACTIVATED	
5	3CC DL Carrier Aggregation Extension	FAJ 121 3084	Carrier Aggregation	FAJ 801 0405	ThreeDCCarrierAggregation	CXC4011714	ACTIVATED	Recommended activated if any of the potential Pcells in the node have 2 or more Scell candidates.

Figure 2.2: Fichier des Features

les features et leurs paramètres radio jouent un rôle crucial dans l'optimisation des performances et des capacités des réseaux de radio. Ils permettent aux opérateurs de configurer et de contrôler différents aspects du réseau pour assurer une expérience utilisateur de qualité et une connectivité fiable. L'optimisation de ces features est une tâche complexe qui nécessite une expertise approfondie et l'utilisation d'outils spécialisés pour obtenir les meilleurs résultats.

- La base de données des réseaux physique 2.3 :

La base de données des réseaux physiques, également connue sous le nom de base de données des sites, est une composante essentielle de la gestion des réseaux de télécommunication. Elle contient des informations détaillées sur les sites d'infrastructure physique utilisés dans le déploiement des réseaux de communication.

CHAPITRE 2. PRÉPARATION DES DONNÉES ET CONCEPTION

	A	B	C	D	E	F	G	H	I	J	K	L
	Site	Identifiant	Site_Code	Gouvernorat	Délégation	BSC	LAC	Secteur	Longitude	Latitude	Nombre de cellules	HBA(m)
1												
2	HBIRA_GSM	33211	MAH3321	Mahdia		BMOKEVO	132	33HBIGA	10.2292	35.1704	1	27.5
3	HBIRA_GSM	33212	MAH3321	Mahdia		BMOKEVO	132	33HBIGB	10.2292	35.1704	1	27.5
4	HBIRA_GSM	33213	MAH3321	Mahdia		BMOKEVO	132	33HBIGC	10.2292	35.1704	1	27.5
5	MHERZA_EST_GSM	33661	MAH3366	Mahdia		BMOKEVO	132	33MHEGA	10.2558	35.396	1	47
6	MHERZA_EST_GSM	33662	MAH3366	Mahdia		BMOKEVO	132	33MHEGB	10.2558	35.396	1	47
7	MHERZA_EST_GSM	33663	MAH3366	Mahdia		BMOKEVO	132	33MHEGC	10.2558	35.396	1	47
8	OULED_CHAM2_GSM	33591	MAH3359	Mahdia		BMOKEVO	132	33CK2GA	10.2726	35.5134	1	47
9	OULED_CHAM2_GSM	33592	MAH3359	Mahdia		BMOKEVO	132	33CK2GB	10.2726	35.5134	1	47
10	OULED_CHAM2_GSM	33593	MAH3359	Mahdia		BMOKEVO	132	33CK2GC	10.2726	35.5134	1	47
11	MZL_HACHED_GSM	33491	MAH3349	Mahdia		BMOKEVO	132	33ZHDGA	10.2839	35.1856	1	27
12	MZL_HACHED_GSM	33492	MAH3349	Mahdia		BMOKEVO	132	33ZHDGB	10.2839	35.1856	1	27
13	MZL_HACHED_GSM	33493	MAH3349	Mahdia		BMOKEVO	132	33ZHDGC	10.2839	35.1856	1	27
14	OULED_CHAME_GSM	33151	MAH3315	Mahdia		BMOKEVO	132	33CKHGA	10.3161	35.4909	1	27.5
15	OULED_CHAME_GSM	33152	MAH3315	Mahdia		BMOKEVO	132	33CKHGB	10.3161	35.4909	1	27.5

Figure 2.3: Fichier d'informations des sites radio

La base de données des réseaux physiques est utilisée par les équipes d'ingénierie et optimisation des réseaux. Elle permet de maintenir une vue d'ensemble complète des sites physiques, de leurs caractéristiques et de leur infrastructure, ce qui facilite la prise de décisions éclairées concernant le déploiement, l'expansion et la maintenance des réseaux de communication.

- Les données trafics des cellules et sites radio 2.4 :

CHAPITRE 2. PRÉPARATION DES DONNÉES ET CONCEPTION

	A	B	C	D	E
1	Date	Hour	ERBS Id	EUtranCell	Traffic PS (Gb)
2	3/1/2023	0	4G_Ain_Boumorra	LKA008O	1.667046919
3	3/1/2023	0	4G_Ain_Boumorra	LKA008P	0.406803214
4	3/1/2023	0	4G_Ain_Boumorra	LKA008Q	0.163470628
5	3/1/2023	0	4G_Ain_Boumorra	LKA008X	1.397840679
6	3/1/2023	0	4G_Ain_Boumorra	LKA008Y	0.05142088
7	3/1/2023	0	4G_Ain_Boumorra	LKA008Z	1.166798756
8	3/1/2023	0	4G_BOUHAJLA_REP	LKA050O	0.073622912
9	3/1/2023	0	4G_BOUHAJLA_REP	LKA050P	0.360123813
10	3/1/2023	0	4G_BOUHAJLA_REP	LKA050Q	0.056221732
11	3/1/2023	0	4G_BOUHAJLA_REP	LKA050X	0.316361547
12	3/1/2023	0	4G_BOUHAJLA_REP	LKA050Y	0.589673989
13	3/1/2023	0	4G_BOUHAJLA_REP	LKA050Z	0.582597801
14	3/1/2023	0	4G_Chogafia	LKA038O	0.653006602
15	3/1/2023	0	4G_Chogafia	LKA038P	1.581438701
16	3/1/2023	0	4G_Chogafia	LKA038Q	1.096505439
17	3/1/2023	0	4G_Chogafia	LKA038X	0.235453132
18	3/1/2023	0	4G_Chogafia	LKA038Y	0.698308577
19	3/1/2023	0	4G_Chogafia	LKA038Z	0.693115639
20	3/1/2023	0	4G_Chrichira	LKA095O	0.982588972
21	3/1/2023	0	4G_Chrichira	LKA095P	0.437202863
22	3/1/2023	0	4G_Chrichira	LKA095Q	1.876372728

Figure 2.4: Fichier de trafic des sites radio

Les données de trafic des cellules et des sites radio font référence aux mesures et aux statistiques recueillies sur l'utilisation et la performance du réseau dans chaque cellule et site radio d'un réseau de télécommunication. Ces données fournissent des informations précieuses sur le volume de trafic, la demande des utilisateurs, la capacité du réseau et la charge des ressources.

- Les KPIs des sites radio [2.5](#) :

Les KPIs (Key Performance Indicators) des sites radio sont des indicateurs clés de performance utilisés pour évaluer la qualité et l'efficacité des sites de communication dans un réseau radio. Ils permettent de mesurer les performances opérationnelles, techniques et commerciales des sites afin de garantir une expérience utilisateur optimale.

CHAPITRE 2. PRÉPARATION DES DONNÉES ET CONCEPTION

	A	B	C	D	E	F	G	H	I	J
1	Date	ERBS Id	EUTranCell	Disponibilité_4G	Call Drop Rate	RRC Setup Succ Rate	E-RAB Estab Succ Rate	S1 Sig Succ Rate	Call Drop Rate (sans mme)	DL Throughput (Mbps)
2	3/1/2023	4G_Ain_Boumorra	LKA008O	100	0.230001	100	99.0792	99.98832	0.306162	2.127120694
3	3/1/2023	4G_Ain_Boumorra	LKA008P	100	0.155763	99.97101	99.72746	99.971	0.19084	1.839921383
4	3/1/2023	4G_Ain_Boumorra	LKA008Q	100	0.179395	100	99.86275	99.9788	0.201845	3.431495798
5	3/1/2023	4G_Ain_Boumorra	LKA008X	100	0.10687	99.97855	99.72886	99.93562	0.130817	21.75674122
6	3/1/2023	4G_Ain_Boumorra	LKA008Y	100	0.064935	99.94454	99.942	100	0.067522	34.39102278
7	3/1/2023	4G_Ain_Boumorra	LKA008Z	100	0.208732	99.93111	99.98353	99.98621	0.253485	28.81189943
8	3/1/2023	4G_BOUHAJLA_REP	LKA050O	100	0	100	100	99.86207	0	11.93169083
9	3/1/2023	4G_BOUHAJLA_REP	LKA050P	100	0.234329	99.95591	99.78179	99.92649	0.290206	3.732013648
10	3/1/2023	4G_BOUHAJLA_REP	LKA050Q	100	0.034807	99.97371	99.89091	99.92111	0.044903	9.592443745
11	3/1/2023	4G_BOUHAJLA_REP	LKA050X	100	0.07734	99.96761	99.96305	100	0.089445	58.1011644
12	3/1/2023	4G_BOUHAJLA_REP	LKA050Y	100	0.093844	100	99.98196	99.98395	0.104581	30.82363999
13	3/1/2023	4G_BOUHAJLA_REP	LKA050Z	100	0.044135	99.98282	99.96799	99.95704	0.057045	37.52280483
14	3/1/2023	4G_Chogafia	LKA038O	100	0.092571	100	99.60416	100	0.127918	4.19626074
15	3/1/2023	4G_Chogafia	LKA038P	100	0.121108	99.96139	99.29172	99.98896	0.146325	0.496977702
16	3/1/2023	4G_Chogafia	LKA038Q	100	0.128018	99.97114	99.33315	99.97835	0.148699	1.737935086
17	3/1/2023	4G_Chogafia	LKA038X	100	0.098717	99.96488	99.70301	100	0.126422	33.07727945
18	3/1/2023	4G_Chogafia	LKA038Y	100	0.854531	99.88739	99.83625	99.97495	1.255474	13.07491508
19	3/1/2023	4G_Chogafia	LKA038Z	100	0.116822	99.89368	99.86955	99.97871	0.13624	15.36089154
20	3/1/2023	4G_Chrichira	LKA095O	100	0.020855	99.98748	99.59554	99.99374	0.024948	1.98375893
21	3/1/2023	4G_Chrichira	LKA095P	100	0.438808	99.03344	100	99.03207	0.4544	40.77430004

Figure 2.5: Fichier des KPIs des sites

Ces KPIs des sites radio sont surveillés et analysés en continu par les équipes d'exploitation et de maintenance des réseaux de télécommunication. Ils aident à identifier les problèmes de performance, à prendre des mesures correctives

- La base de connaissance et bonne pratique sur d'optimisation des KPIs des sites radio voir annexe I.

La base de connaissance sur l'optimisation des KPIs des sites et les meilleures pratiques regroupe les informations, les techniques et les recommandations permettant d'améliorer les performances et la qualité des sites de communication dans un réseau radio. Elle offre des conseils et des stratégies pour optimiser les KPIs clés, afin de garantir une expérience utilisateur optimale et d'atteindre les objectifs opérationnels.

I.2 Intégration des données sous mongoDB

Nous avons utilisé Python et la librairie Pandas 2.6 pour convertir des données fournies en DataFrame, qui est une structure de données de type tableau. En-

CHAPITRE 2. PRÉPARATION DES DONNÉES ET CONCEPTION

suite, nous nous avons connecté à la base de données MongoDB en utilisant PyMongo et nous avons créé une nouvelle collection dans laquelle les données seraient stockées. Enfin, nous avons utilisé le DataFrame pour insérer les données dans la collection MongoDB, en veillant à ce que les données soient correctement formatées et mises en correspondance avec les champs appropriés.

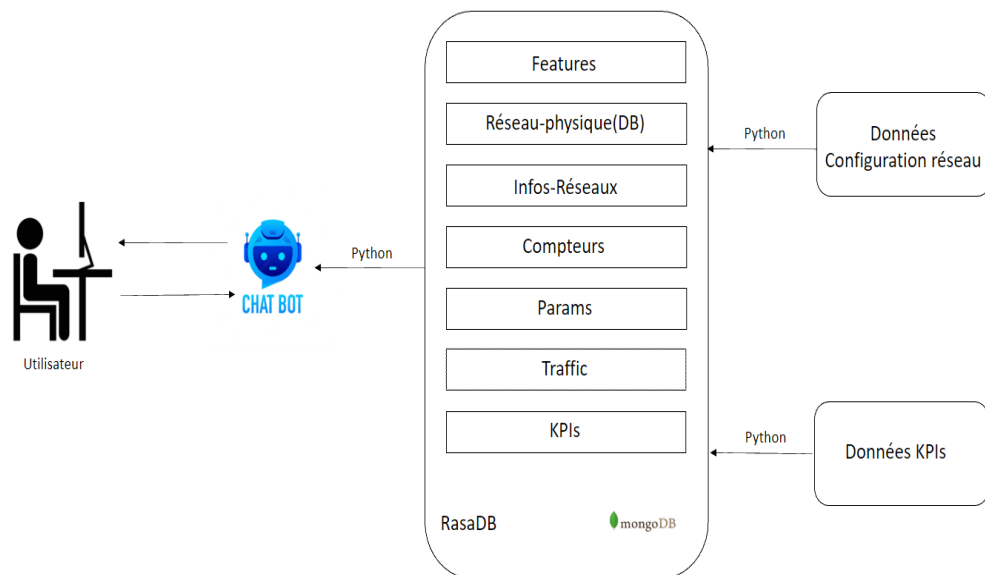


Figure 2.6: Intégration des données sous mongoDB

Dans l'ensemble, ce processus démontré dans la figure 2.6 est réalisé à l'aide de code dans la figure 2.7, nous'a permis de stocker et de gérer efficacement les données des réseaux radio, ce qui rend plus facile à analyser et à les manipuler à l'avenir.

```
#Library importation
import pymongo
import pandas as pd
#Excel Importation into a dataframe
df= pd.read_excel('L_Acc_Analysis_Report (1).xlsx', sheet_name='Report 3')
# Establishing a connection to the MongoDB database is
client = pymongo.MongoClient("mongodb://localhost:27017/")
#Database to insert data to
db = client["rasa"]
#Collection to add data to
collection = db["Compteurs"]
# Convert the DataFrame to a list of dictionaries
records = df.to_dict('records')
# Insert the records into the collection
collection.insert_many(records)
```

Figure 2.7: Insertion des données du mongoDB

I.3 Chargement des données par Rasa

Les données communiquées par le client sont bien sûr disponibles sur une base de données MongoDB avec des différentes collections "params", "Conteurs", "Features" tel que montrées dans la figure 2.8.

```
import pymongo
# retrieve all documents from the collection
client = pymongo.MongoClient("mongodb://localhost:27017/")
db = client["rasa"]
collection = db["Chap2"]
data = list(collection.find())
# create a Pandas DataFrame from the list of dictionaries
Xdf = pd.DataFrame(data)
```

Figure 2.8: Extraction des données du mongoDB

CHAPITRE 2. PRÉPARATION DES DONNÉES ET CONCEPTION

Le table 2.1 présente les informations concernant toutes les collections figurant dans notre tableau.

Nom de la collection	Nombre de lignes et colonnes	Description
Features	(149, 5)	Cette collection présente la définition de chaque fonctionnalité
Infos-Optimisation	(13, 6)	Cette collection indique les étapes de résolution des problèmes techniques
Compteurs	(10, 9)	Cette collection indique les valeurs des les conteurs pour les vérifier
Réseau-physique(DB)	(2, 19)	Cette collection montre les informations au sujet des réseaux physiques de chaque site.
Traffic	(1079949, 5)	Cette collection nous informe sur le trafic dans chaque cellule des sites
KPIs	(45056, 7)	Cette collection nous informe sur le Kpis dans chaque cellule des sites

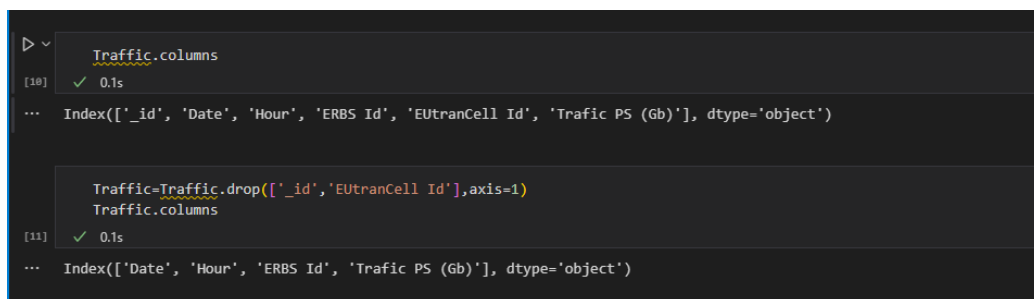
Table 2.1: Informations sur toutes les collections de la table

II Préparation des données

Le processus de préparation des données est une étape essentielle pour l'exploration des données. Il s'agit de nettoyer, de transformer et de fusionner les données afin de créer un ensemble de données uniforme et de grande qualité. Le processus nécessite une attention particulière aux détails et une compréhension en profondeur des données et de leur contexte.

II.1 Suppression des caractéristiques inutiles

La suppression de caractéristiques inutiles consiste à supprimer des attributs de données redondants, non pertinents ou bruyants de l'ensemble de données. Nous simplifions les données et par conséquent, nous améliorons la précision analytique. Ce procédé est exprimé par la figure 2.9.



```

Traffic.columns
[10] ✓ 0.1s
... Index(['_id', 'Date', 'Hour', 'ERBS Id', 'EUTranCell Id', 'Trafic PS (Gb)'], dtype='object')

Traffic=Traffic.drop(['_id', 'EUTranCell Id'], axis=1)
Traffic.columns
[11] ✓ 0.1s
... Index(['Date', 'Hour', 'ERBS Id', 'Trafic PS (Gb)'], dtype='object')
```

Figure 2.9: Suppression des caractéristiques inutiles

II.2 Néttoyage des données

Au cours du processus de nettoyage des données, nous repérons et corrigeons ou supprimons des données inexactes, incomplètes ou incohérentes de l'ensemble de données. En premier lieu, nous vérifions le nombre de valeurs non définies dans chaque colonne, puis nous nettoignons les enregistrements avec une valeur "Nan" par l'utilisation de lissage par moyenne comme indiqué par la figure 2.10.

```
result.isna().sum()

...
_id                0
Date               0
ERBS Id            0
EUTRANCell Id      0
Disponibilite_4G   0
Call Drop Rate     277
RRC Setup Succ Rate 274
E-RAB Estab Succ Rate 274
S1 Sig Succ Rate   274
Call Drop Rate (sans mme) 279
DL Throughput (Mbps) 282
dtype: int64
```

```
result["Call Drop Rate"] = result["Call Drop Rate"].fillna(result["Call Drop Rate"].mean())
result["RRC Setup Succ Rate"] = result["RRC Setup Succ Rate"].fillna(result["RRC Setup Succ Rate"].mean())
result["E-RAB Estab Succ Rate"] = result["E-RAB Estab Succ Rate"].fillna(result["E-RAB Estab Succ Rate"].mean())
result["S1 Sig Succ Rate"] = result["S1 Sig Succ Rate"].fillna(result["S1 Sig Succ Rate"].mean())
result["Call Drop Rate (sans mme)"] = result["Call Drop Rate (sans mme)"].fillna(result["Call Drop Rate (sans mme)"].mean())
result["DL Throughput (Mbps)"] = result["DL Throughput (Mbps)"].fillna(result["DL Throughput (Mbps)"].mean())
result.isna().sum()

...
_id                0
Date               0
ERBS Id            0
EUTRANCell Id      0
Disponibilite_4G   0
Call Drop Rate     0
RRC Setup Succ Rate 0
E-RAB Estab Succ Rate 0
S1 Sig Succ Rate   0
Call Drop Rate (sans mme) 0
DL Throughput (Mbps) 0
dtype: int64
```

Figure 2.10: Traitement des valeurs nuls

II.3 Transformation des données

Nous convertissons les données d'un format à un autre pour les rendre plus adaptées à l'analyse comme le montre la figure 2.11.

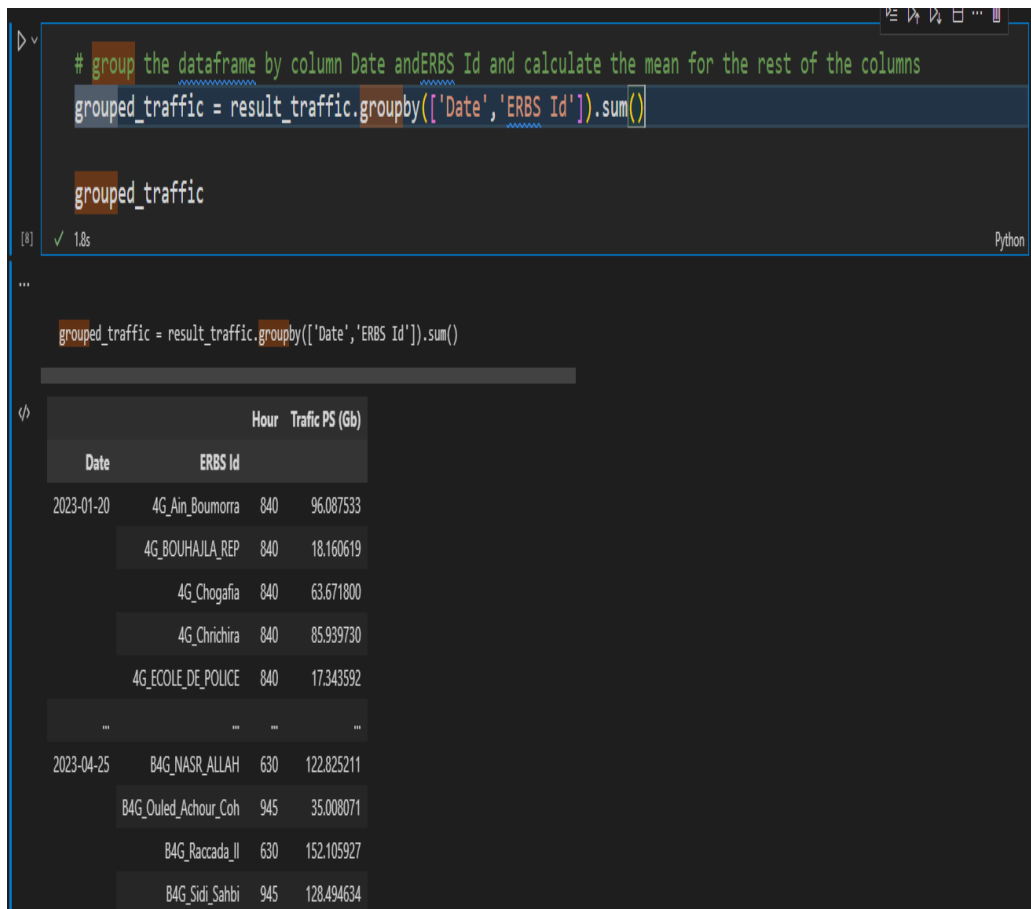


Figure 2.11: Transformation de données

III Conception des modules

Nous avons développé différents modules intégrés au Framework Rasa pour assurer le bon fonctionnement de notre chatbot.

Nous avons configuré et entraîné le NLU de Rasa pour améliorer sa précision et sa capacité à interpréter les requêtes des utilisateurs de manière précise et cohérente. Ces Modules sont :

1. Module 1 : Accès aux informations réseaux
2. Module 2 : Diagnostique et recommandations
3. Module 3 : Analyse et prédiction de performance

III.1 Conception du module : Accès aux informations réseaux (Module 1)

Pour ce module, notre objectif est de faciliter l'accès à la base de connaissances et aux informations des sites radio, par conséquent, nous avons défini des questions types afin que nous puissions interagir avec notre assistant [2.12](#).

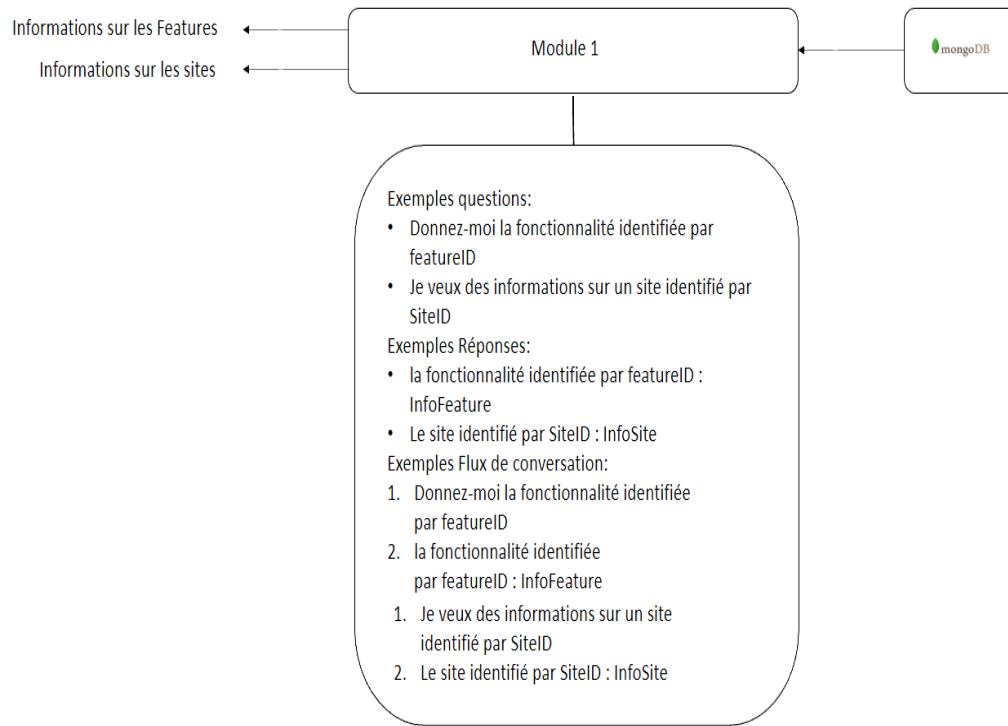


Figure 2.12: Processus de traitement du module 1 : accès informations réseaux

III.2 Conception du module : Diagnostique et recommandations (Module 2)

Dans module intitulé *Diagnostique et recommandations*, nous sommes intéressés pour aider aux tâches de diagnostic des causes de dégradation des KPIs radio donc pour que nous pouvons interagir avec notre chatbot illustrée dans la figure 2.13.

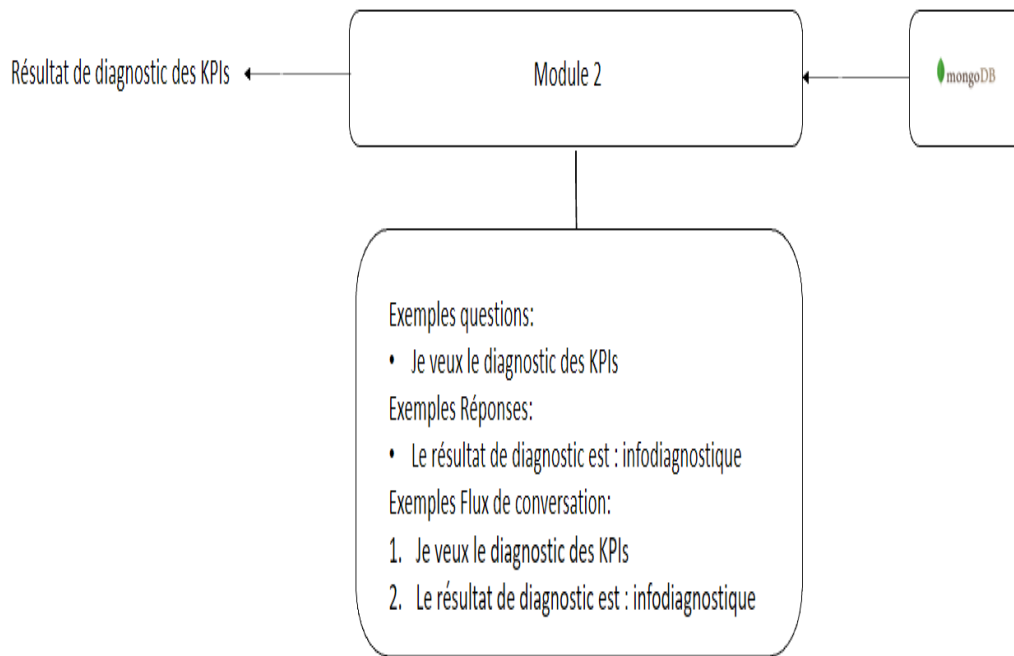


Figure 2.13: Processus de traitement du module 2 : diagnostique et recommandations

III.3 Conception du module : Analyse et prédiction de performance (Module 3)

Pour ce module intitulé *Analyse et prédiction de performance*, nous devons faire les analyses réseau sur demande basé sur des algorithmes de machine Learning [2.14](#)

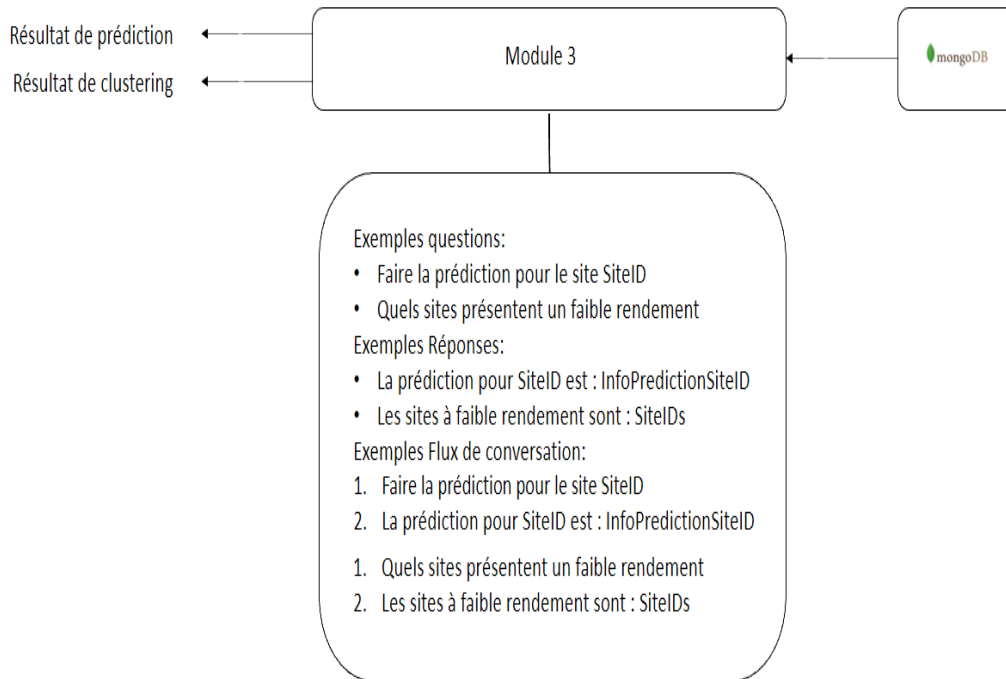


Figure 2.14: Processus de traitement du module 3 : Analyse et prédiction de performance

Les modèles utilisés dans le module *Analyse et prédiction de performance* sont choisis selon la nature des données disponibles, dans notre projet, notre variable cible est une variable de séries chronologiques qui correspond à la valeur du trafic de chaque site, nous sommes donc dans un problème de prédiction et de clustering.

Conclusion

La phase de compréhension et de préparation des données est une étape cruciale dans tout projet d'exploration de données. Au cours de cette phase, nous avons acquis une compréhension approfondie de notre ensemble de

CHAPITRE 2. PRÉPARATION DES DONNÉES ET CONCEPTION

données et l'avons préparé pour la phase de modélisation.

Nous avons nettoyé, transformé et intégré nos données pour créer un ensemble de données cohérent et de grande qualité. Notre ensemble de données est maintenant prêt pour la conception des trois modules qui nous aideront à atteindre nos objectifs de projet.

Dans le prochain chapitre, nous allons aborder la phase de déploiement pour développer les modules cibles.

CHAPITRE 3

DÉPLOIEMENT ET ÉVALUATION DE LA SOLUTION

Introduction	36
I Connexion à la base de données MongoDB de Rasa	37
I.1 Installation des objets nécessaires	37
I.2 Configuration de la base de donnée	38
II Interfaçage avec une page web	39
III Test et résultats du chatbot	40
III.1 résultats du chatbot	40
Module : <i>Accès aux informations réseaux</i>	40
Module : <i>Diagnostic et recommandations</i>	42
Module : <i>Analyse et prédiction</i>	43
III.2 Évaluation des performances du chatbot	45
Conclusion	47

Introduction

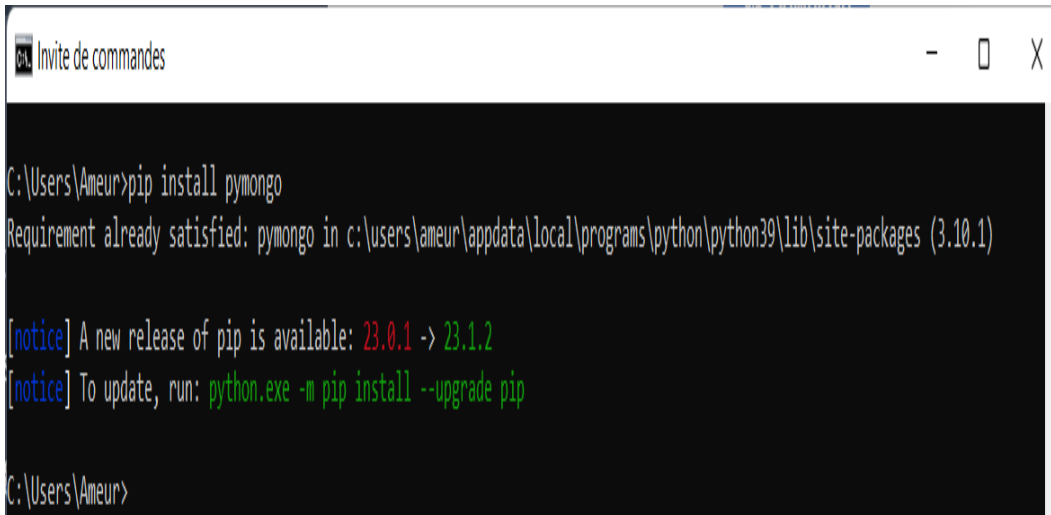
Ce chapitre est consacré à présenter l'architecture, la configuration et à l'évaluation du chatbot développé à l'aide du Framework Rasa.

Dans ce chapitre, nous explorerons en détail les différentes étapes et composants nécessaires pour mettre en place avec succès notre chatbot optimisé pour l'assistance à l'optimisation radio

I Connexion à la base de données MongoDB de Rasa

I.1 Installation des objets nécessaires

Pour connecter à une base de données MongoDB, nous devons installer MongoDB. Pour se connecter à une base de données MongoDB dans un chatbot Rasa, nous aurons besoin de mettre en place une bibliothèque d'adaptateur de base de données comme pymongo en utilisant la commande `pip install pymongo` comme indiqué par la figure 3.1. ainsi que la bibliothèque pandas pour gérer les données de format JSON en des dataframes en utilisant la commande `py -m pip install pandas`.



```
Invite de commandes

C:\Users\Ameur>pip install pymongo
Requirement already satisfied: pymongo in c:\users\ameur\appdata\local\programs\python\python39\lib\site-packages (3.10.1)

[notice] A new release of pip is available: 23.0.1 -> 23.1.2
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\Ameur>
```

Figure 3.1: Execution de commande d'installation de bibliothèque "pymongo"

I.2 Configuration de la base de donnée

Le fichier de configuration `endpoints.yml` 3.2 de Rasa liste les canaux de communication dont le chatbot peut servir pour communiquer avec ses utilisateurs. nous pouvons définir différents points d'entrée et de sortie dans le fichier. nous pouvons personnaliser les paramètres de chaque endpoint dans le fichier `endpoints.yml`, y compris les clés API ou les jetons nécessaires à la connexion au canal, l'URL du canal, etc [1].

Nous pouvons également spécifier quel endpoint devrait être le canal principal pour envoyer et recevoir des messages à l'intérieur du fichier.

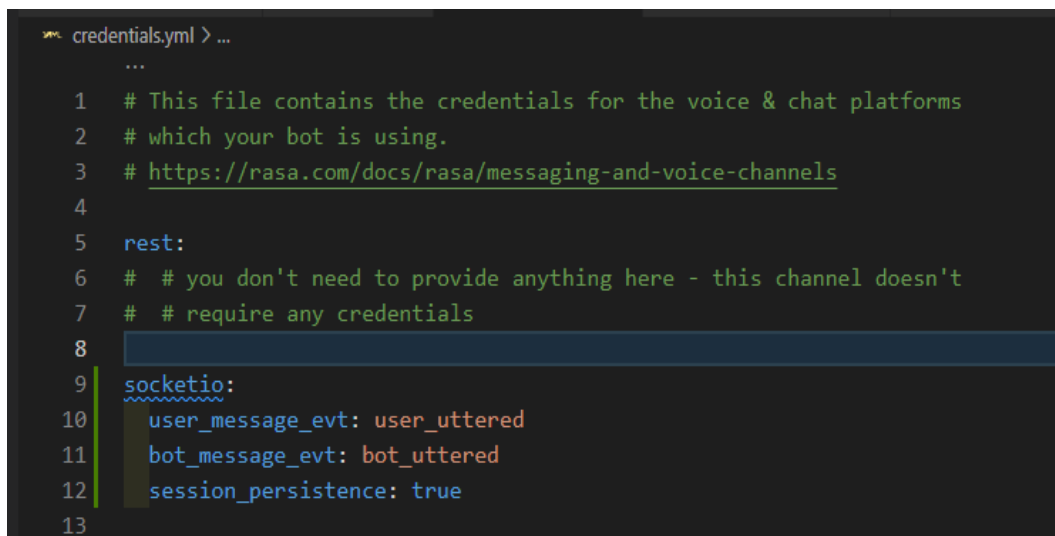
```
endpoints.yml > ...
19
20 # tracker_store:
21 #   type: SQL
22 #   dialect: "mysql+pymysql" # the dialect used to interact with the db
23 #   url: "localhost" # (optional) host of the sql db, e.g. "localhost"
24 #   db: "rasa" # path to your db
25 #   username: "root" # username used for authentication
26 #   password: "root" # password used for authentication
27
28 tracker_store:
29   type: mongod
30   url: mongodb://localhost:27017
31   db: rasa
32   username:
33   password:
34
35 # Event broker which all conversation events should be streamed to.
36 # https://rasa.com/docs/rasa/event-brokers
37
38 #event_broker:
39 #   url: localhost
40 #   username: username
41 #   password: password
42 #   queue: queue
43
```

Figure 3.2: Configuration du endpoints

II Interfaçage avec une page web

Pour se connecter à une page web nous avons suivi ces étapes :

1. Nous activons le canal Socket.IO dans notre projet Rasa en ajoutant la configuration suivante au fichier `credentials.yml` comme indiqué dans la figure 3.3

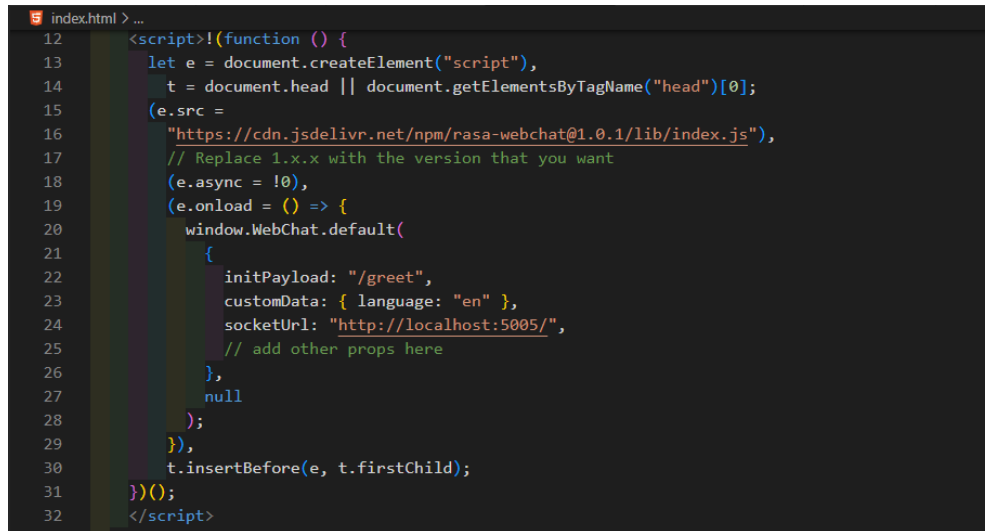


```
credentials.yml > ...  
...  
1  # This file contains the credentials for the voice & chat platforms  
2  # which your bot is using.  
3  # https://rasa.com/docs/rasa/messaging-and-voice-channels  
4  
5  rest:  
6  # # you don't need to provide anything here - this channel doesn't  
7  # # require any credentials  
8  
9  socketio:  
10     user_message_evt: user_uttered  
11     bot_message_evt: bot_uttered  
12     session_persistence: true  
13
```

Figure 3.3: Credentials.yml

2. Nous avons ajouté la bibliothèque client Socket.IO à notre client webchat en ajoutant la balise de script suivante au fichier `index.html` représenté dans la figure 3.4

CHAPITRE 3. DÉPLOIEMENT ET ÉVALUATION DE LA SOLUTION



```
12 <script>(function () {
13   let e = document.createElement("script"),
14       t = document.head || document.getElementsByTagName("head")[0];
15   (e.src =
16     "https://cdn.jsdelivr.net/npm/rasa-webchat@1.0.1/lib/index.js"),
17     // Replace 1.x.x with the version that you want
18     (e.async = !0),
19     (e.onload = () => {
20       window.WebChat.default(
21         {
22           initPayload: "/greet",
23           customData: { language: "en" },
24           socketUrl: "http://localhost:5005/",
25           // add other props here
26         },
27         null
28       );
29     }),
30     t.insertBefore(e, t.firstChild);
31   })();
32 </script>
```

Figure 3.4: Configuration de credentials.yml

3. nous démarrons le serveur Rasa dans l'invite de commande en utilisant **rasa run -m models --enable-api --cors "*" .**
4. Enfin, nous ouvrons le fichier index.html dans un navigateur web. Dans le cas d'erreur, nous devons désinstaller notre version de websockets en utilisant **pip uninstall websockets** et installer la version 10.0 en utilisant **pip install websockets==10.0**.

III Test et résultats du chatbot

III.1 résultats du chatbot

Module : *Accès aux informations réseaux*

Nous rappelons que le module *Accès aux informations réseaux* consiste à ce que le chatbot assiste les ingénieurs, techniciens et utilisateurs de Tunisie Télécom par fournir des informations sur les éléments du réseau existant selon des demandes de type : me fournir les informations relatives au site radio

CHAPITRE 3. DÉPLOIEMENT ET ÉVALUATION DE LA SOLUTION

au Nom "Nom site" 3.5 ou présenter des informations sur les paramètres ou Features radio 3.6.

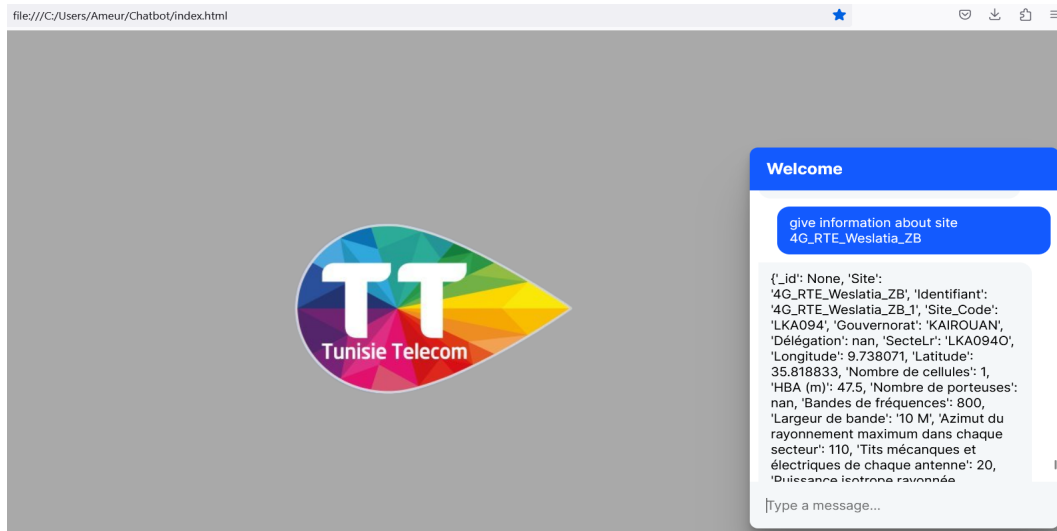


Figure 3.5: Informations sur les sites radio

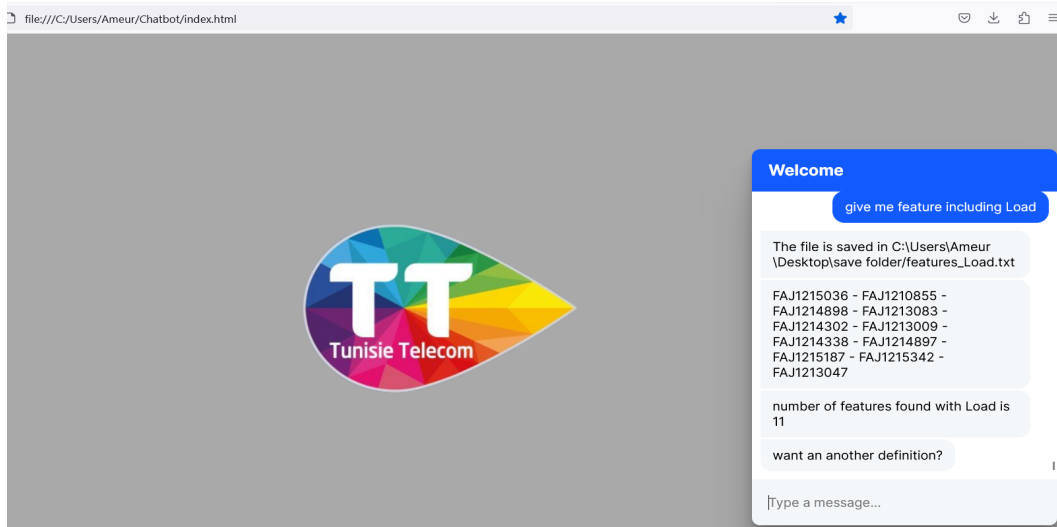


Figure 3.6: Informations sur les Features radio

CHAPITRE 3. DÉPLOIEMENT ET ÉVALUATION DE LA SOLUTION

Module : *Diagnostic et recommandations*

Le module intitulé *Diagnostic et recommandations* à implémenter consiste à ce que le chatbot assiste les ingénieurs d'optimisation des performances des réseaux radio de Tunisie Télécom par filtrer et présenter les sites radio à problématique ou dite avec des KPIs dégradés 3.7.

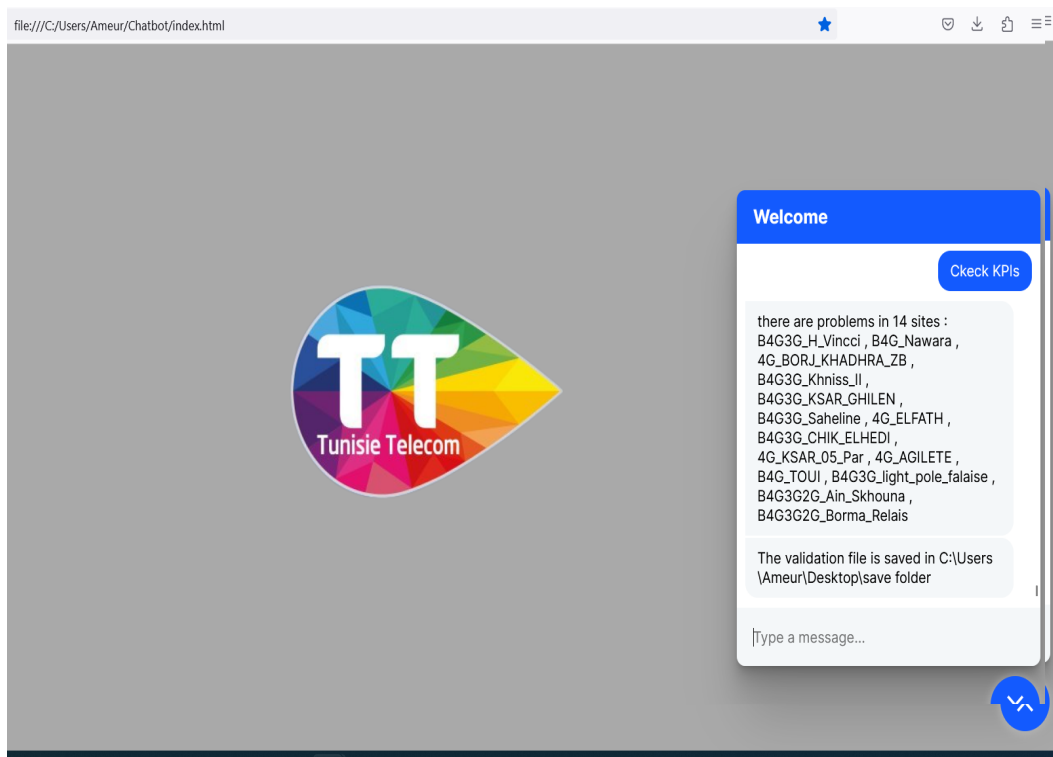


Figure 3.7: Présentation des sites avec des KPIs dégradés

Aussi, le chatbot permet de présenter les actions recommandées en vue de remédier et améliorer les KPIs dégradés 3.8 et ce selon des bonnes pratiques de l'optimisation radio (voir annexe partie I).

CHAPITRE 3. DÉPLOIEMENT ET ÉVALUATION DE LA SOLUTION

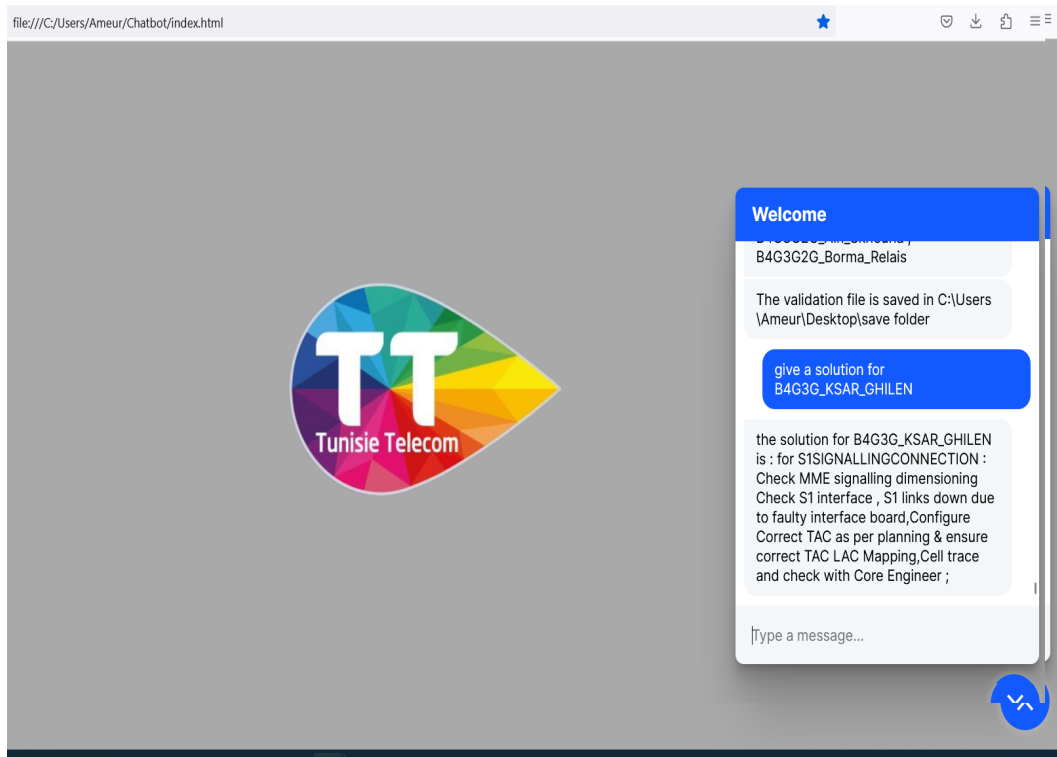


Figure 3.8: Recommandation pour les sites avec des KPIs dégradés

Module : *Analyse et prédiction*

Le module intitulé *Analyse et prédiction* à implémenter consiste à ce que le chatbot assiste les ingénieurs et techniciens de Tunisie Télécom par segmenter les sites pour détecter les sites dites *Top Worst* 3.9 et donner des prédictions sur les KPIs trafic radio en utilisant l'algorithme "LSTM" 3.10.

CHAPITRE 3. DÉPLOIEMENT ET ÉVALUATION DE LA SOLUTION

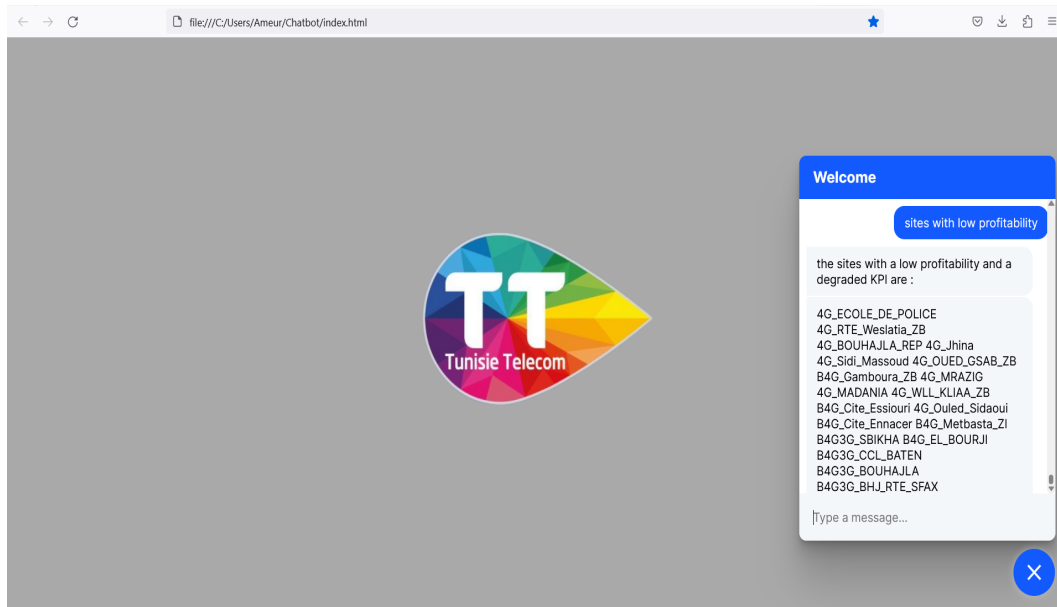


Figure 3.9: Détection des sites radio *Top Worst*

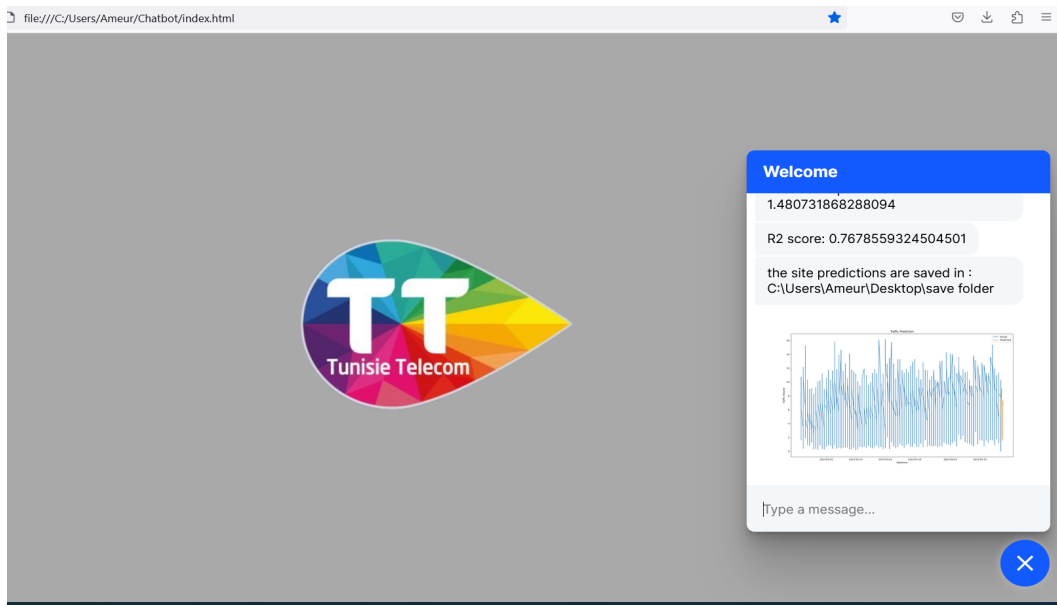


Figure 3.10: Prédiction du trafic des sites radio

III.2 Évaluation des performances du chatbot

Pour évaluer notre solution, nous avons utilisé la commande **rasa test core** qui nous a donné :

- L'histogramme des intents de la figure 3.11 qui nous montre la distribution des intentions prévues, l'identification des déséquilibres potentiels, les erreurs de classification et l'efficacité du modèle dans la reconnaissance des intentions fréquentes et peu fréquentes. Il fournit de l'information sur la performance de classification de l'intention du modèle et aide à identifier les domaines à améliorer.

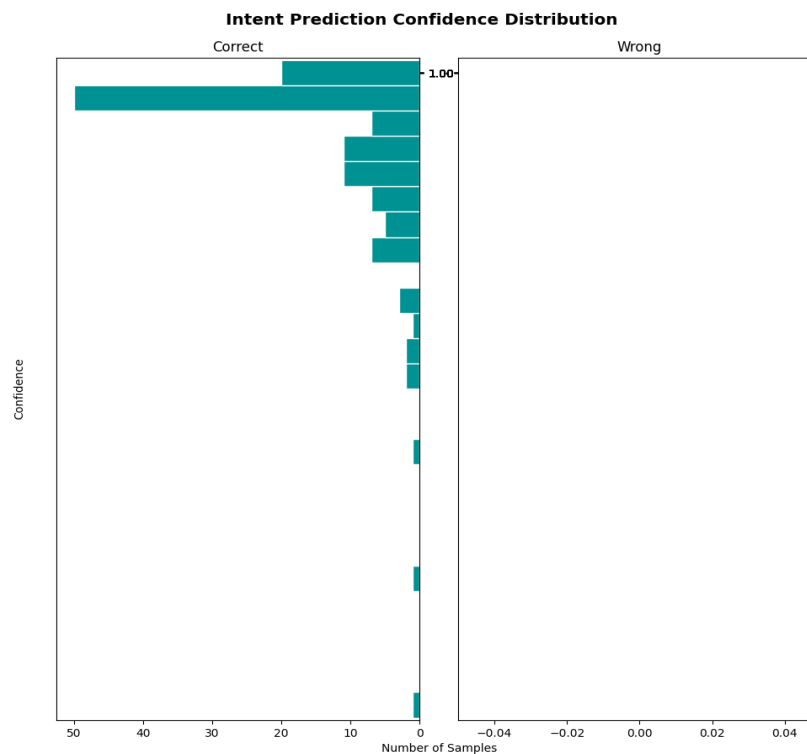


Figure 3.11: Histogramme des intents

- La matrice de confusion des intents de la figure 3.12 fournit des renseignements sur les erreurs de classification entre les intentions prévues,

CHAPITRE 3. DÉPLOIEMENT ET ÉVALUATION DE LA SOLUTION

permettant de cerner les tendances et d'évaluer l'exactitude du modèle de classification des intentions.

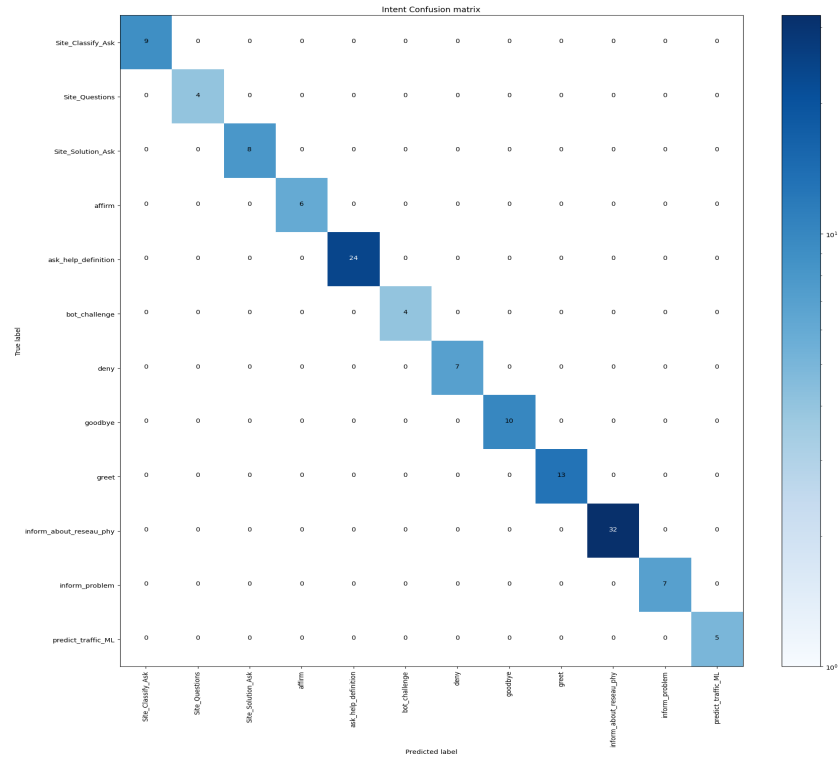


Figure 3.12: Matrice de confusion des *intents*

- La matrice de confusion des flux de conversation de la figure 3.13 fournit des renseignements sur les erreurs de classification entre les conversations prévues, permettant de cerner les tendances et d'évaluer l'exactitude du modèle de classification des conversations.

CHAPITRE 3. DÉPLOIEMENT ET ÉVALUATION DE LA SOLUTION

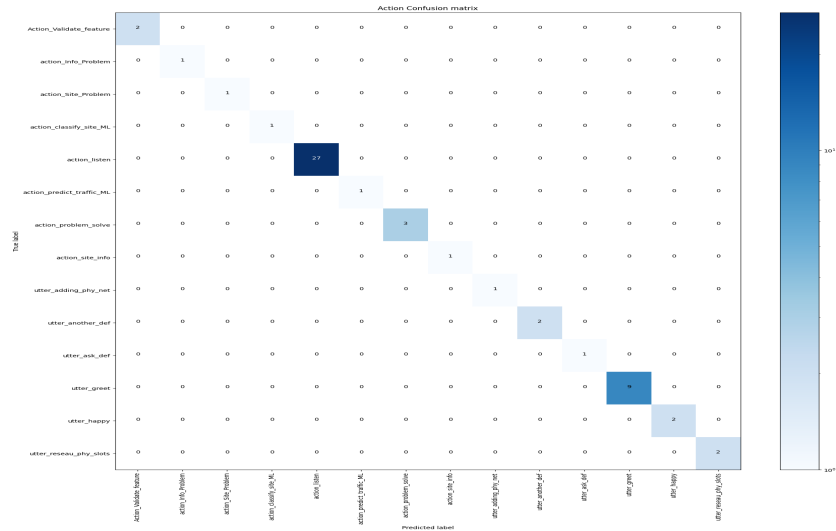


Figure 3.13: Matrice de confusion de flux de conversation

Toutes les matrices d'évaluation de performance, présentées, reflètent la bonne performance de notre solution en terme de erreurs de classification et détection des *Intents* et compréhension des flux de conversations.

Conclusion

Ce chapitre a abordé les aspects essentiels pour le déploiement d'une solution chatbot basée sur le Framework Rasa. Nous avons débuté en détaillant l'installation des objets nécessaires. Ensuite, nous avons procédé à la configuration de la base de données en vue de garantir une gestion optimale des données du chatbot.

Nous avons également exploré l'interfaçage avec une page web, ce qui a permis de créer une interface utilisateur conviviale et interactive pour le chatbot.

Enfin, nous avons présentés les résultats et réponses que peut le chatbot répondre et ce selon les objectifs fixés ainsi qu'une évaluation de ses performance.

CONCLUSION GÉNÉRALE

Nous avons abordé plusieurs aspects clés liés à l'optimisation des réseaux radio, ainsi qu'à la conception et à l'implémentation d'un chatbot efficace. Nous avons commencé par présenter le contexte du projet, en explorant les concepts fondamentaux de l'optimisation radio dans les réseaux de télécommunication. Nous avons également étudié les principes de base des chatbots et du framework Rasa, afin de comprendre comment les intégrer de manière optimale dans notre projet.

Nous avons ensuite procédé à l'architecture et à la configuration du framework Rasa, en mettant l'accent sur la connexion à la base de données MongoDB, l'installation des objets nécessaires et la configuration de la base. Cette étape cruciale nous a permis de mettre en place une infrastructure solide pour le développement de notre chatbot.

Ensuite, nous avons abordé l'interfaçage avec une page web, ce qui nous a permis de créer une interface utilisateur conviviale et interactive pour le chatbot. Cette interface a facilité l'interaction avec les utilisateurs et a contribué à une expérience utilisateur améliorée.

Nous avons également effectué des tests approfondis du chatbot et évalué ses performances ce qui nous a fourni une évaluation objective de son efficacité.

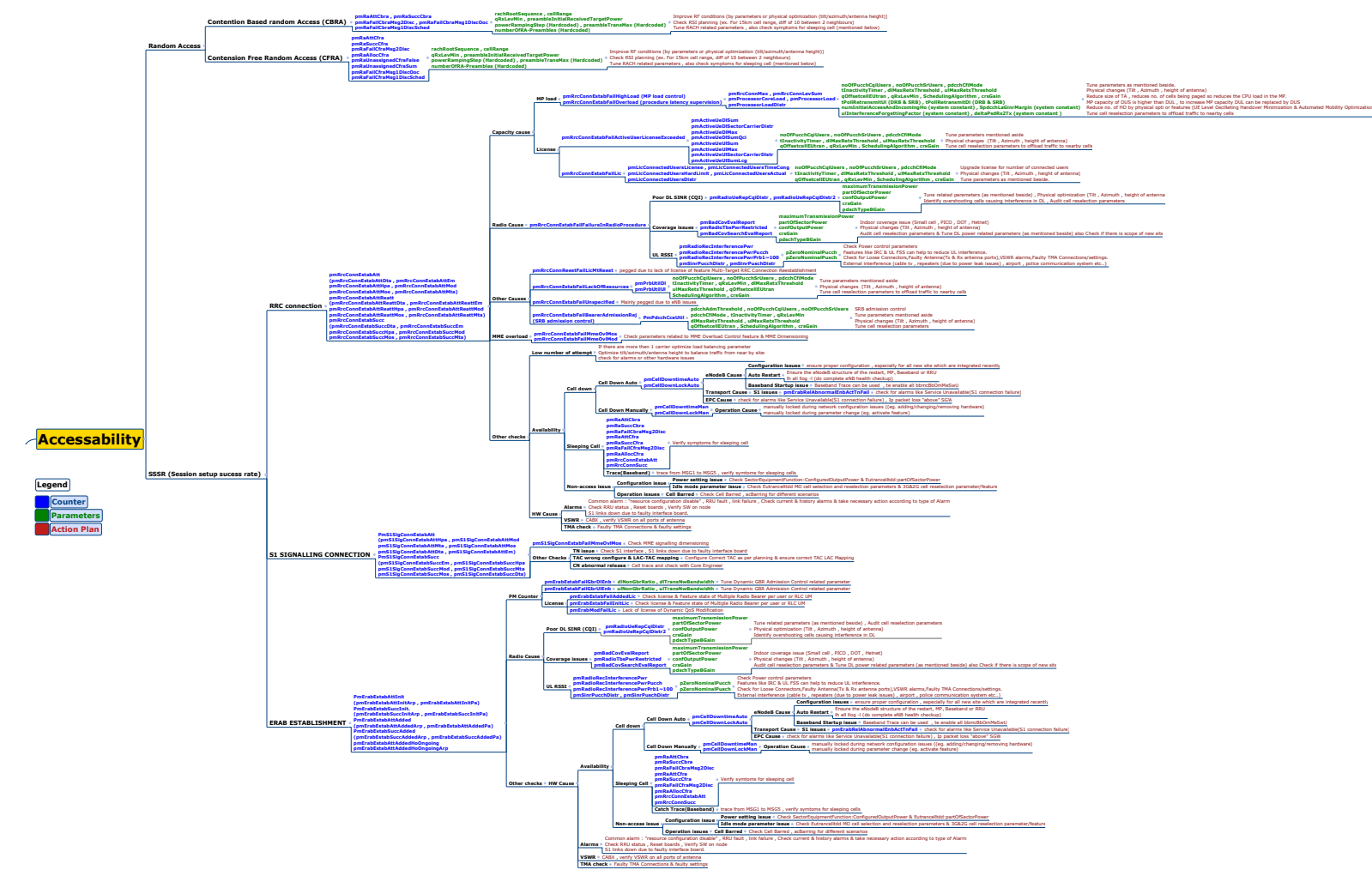
BIBLIOGRAPHIE

- [1] *FrameWork Rasa: Open source.* <https://rasa.com/>. Consulté le 04/03/2023.
- [2] *Python.* <https://www.bocasay.com/fr/python-meilleur-langage-de-programmation-de-lannee/>. Consulté le 03/03/2023.
- [3] *vscode.* <https://code.visualstudio.com>. Consulté le 03/03/2023.
- [4] *mongo.* <https://www.mongodb.com/docs/manual/>. Consulté le 03/03/2023.

ANNEXE

I LTE KPI-Accessability

ACCESSABILITY-L14B



II Initiation à Rasa

Pour télécharger et installer Rasa, nous devons suivre ces étapes :

1. Installation de python : Rasa est écrit en Python, ainsi Python doit être installé sur la machine doit avoir. En tapant la commande **python --version**, Nous pouvons voir si Python est installé. Nous utiliserons la version 3.9.12 qui peut-être installé à partir de [site officiel du python](#).
2. Installation de pip : Il nous faudra le gestionnaire de paquetage Python pip pour installer Rasa et toutes ses dépendances. En exécutant la commande **pip --version** sur le terminal, nous saurons si pip est installé ou non. Si nous n'avons pas encore installé pip, nous pouvons le faire en accédant au [site officiel pip](#) et en suivant les instructions d'installation.
3. Installation de Anaconda: nous allons télécharger le paquet d'installation à partir du [site officiel d'anaconda](#) puis configurer le chemin en ajoutant le bin au PATH et vérifier l'installation en utilisant la commande **conda --version**.
4. Création d'environnement: Nous utilisons la commande **conda create --name envname python==3.9.13**
5. Activation de l'environnement : Nous utilisons **conda activate envname**
6. Installation de Rasa : Nous pouvons utiliser la commande **pip install rasa==3.4.2** pour installer notre version de Rasa.
7. Vérifier l'installation : Nous pouvons utiliser la commande **rasa --version** pour confirmer l'installation. Le numéro de version que nous avons installé doit être imprimé en conséquence. En cas d'erreur, désinstaller websockets en utilisant la commande **pip uninstall websockets** et

installer la version 10.0 en utilisant la commande **`pip install websockets==10.0`**

En suivant ces instructions, nous pouvons commencer à construire notre premier bot après avoir installé Rasa :

1. Création d'un projet: En tapant la commande **`rasa init`** dans notre terminal, nous pouvons démarrer un nouveau projet Rasa. Les fichiers et répertoires par défaut requis pour un projet Rasa seront créés dans un nouveau répertoire.
2. Définition de notre flux de conversation : Nous devons spécifier comment notre bot va mener une discussion. Nous pouvons y parvenir en modifiant notre fichier **`stories.yml`** dans le répertoire de données de notre projet Rasa. En enregistrant les nombreuses intentions et réponses des utilisateurs dans ce fichier, nous pouvons définir les différentes directions qu'une discussion peut prendre.
3. Définition de notre domaine: Le domaine établit la gamme de choses et d'activités avec lesquelles notre bot peut s'engager. En modifiant notre fichier **`domain.yml`** situé dans le répertoire de domaines de notre projet Rasa, nous pouvons spécifier les intentions, les entités, les actions et les modèles que notre bot utilisera pendant la discussion.
4. Création des actions: Pour ajouter une action personnalisée à notre bot Rasa, la première étape est la spécification dans le fichier **`actions.py`** selon les informations fournies par le graphique. Les routines Python appelées actions personnalisées implémentent un comportement particulier.

Ensuite, nous ajoutons l'action personnalisée au fichier **`domain.yml`** dans le répertoire de domaines de notre projet Rasa, le nom de l'action doit être ajouté à la section des actions du fichier.

Puis nous appelons l'action personnalisée des histoires dans le fichier **stories.yml** dans le répertoire et nous devons exécuter la commande `rasa run actions` pour lancer le serveur des actions avant d'utiliser l'action personnalisée. L'exécution des actions personnalisées est gérée par le serveur d'actions, un composant distinct de données de notre projet Rasa

5. Entrainement de modèle: En saisissant la commande suivante dans notre terminal, nous pouvons créer notre modèle : **rasa train**. Cela sera utilisé pour former un modèle d'apprentissage automatique que notre bot utilisera pour comprendre les entrées des utilisateurs et produire des réponses en utilisant l'information dans nos `stories.yml` et `domain.yml` fichiers.
6. Démarrage de serveur des actions : Nous devons exécuter la commande **rasa run actions**. Pour lancer le serveur d'actions avant d'utiliser l'action personnalisée. L'exécution des actions personnalisées est gérée par le serveur d'actions, un composant distinct.
7. Test de modèle: En entrant la commande **rasa shell** dans notre terminal, nous pouvons tester notre modèle en saisissant les entrées utilisateur et visualiser les réponses générées. Nous pouvons tester notre bot dans un shell interactif après avoir tapé la commande **rasa interactive**.
8. Déploiement du bot: Une fois que nous serons satisfaits des fonctionnalités de notre bot, nous pouvons le déployer.

Pour déployer notre chatbot, nous devons suivre les étapes suivantes:

1. Installation des dépendances : Pour déployer notre bot Rasa, nous devons nous assurer que toutes les dépendances nécessaires sont présentes sur le serveur. Python, pip, et la bibliothèque Rasa sont inclus dans ceci.

2. Copie des fichiers du projet : Les fichiers de notre projet Rasa local doivent être copiés sur le serveur.
3. Entraînement de modèle sur le serveur : Formons le modèle sur le serveur en utilisant la même commande que nous avons utilisée localement : **rasa train**.
4. Démarrage de serveur des actions : Les actions personnalisées spécifiées dans notre domaine sont gérées par le serveur d'action, un composant distinct. Exécutez la commande suivante : **rasa run actions** pour lancer le serveur des actions.
5. Établissement d'un proxy inverse : Nous devons configurer un proxy inverse, comme NGINX ou Apache, afin de rendre le bot Rasa accessible via Internet. Les demandes seront reçues et transmises au robot Rasa opérant sur le serveur par le proxy inverse.
6. Test de déploiement : Nous pouvons communiquer avec le bot Rasa soit un client de chat, soit une interface de chat en ligne pour tester le déploiement.