Développement d’un Système de Traitement de Commandes Vocales et Textuelles pour l'Identification et l'Optimisation des Trajets SNCF

1- Introduction

Dans un monde où la mobilité joue un rôle crucial dans la vie quotidienne, le traitement et la gestion efficace des ordres de voyage deviennent essentiels pour offrir des services rapides et fiables. Le projet s'inscrit dans cette optique, en combinant des techniques avancées de traitement du langage naturel (NLP) et d'optimisation pour répondre aux besoins des usagers des transports.

L'objectif principal de ce projet est de développer une solution capable de comprendre des commandes textuelles ou vocales formulées en français pour générer des itinéraires de voyage adaptés. Cette solution ne se limite pas à identifier les villes de départ et d'arrivée, mais elle est également capable de valider la cohérence des trajets, de gérer des étapes intermédiaires, et d'exploiter les données des gares SNCF pour fournir des résultats pertinents. En intégrant des modèles NLP comme SpaCy et BERT, cette approche permet d'analyser avec précision des phrases souvent complexes et variées.

Le projet se concentre également sur la création d'un pipeline robuste et modulaire, comprenant la reconnaissance d'entités nommées (NER), la normalisation des noms de lieux, et l'intégration d'algorithmes d'optimisation des trajets. Ces composants sont conçus pour offrir une solution extensible, pouvant

être enrichie par des fonctionnalités supplémentaires, telles que l'intégration de données multimodales ou l'optimisation des ressources énergétiques.

Dans ce rapport, nous présenterons en détail les étapes de conception, les choix méthodologiques, et les résultats obtenus au cours du développement de cette solution. Nous discuterons également des défis rencontrés, des performances mesurées, et des perspectives d'amélioration pour répondre aux exigences croissantes en matière de mobilité intelligente.

2- Analyse des besoins solutions apporter

1. Détection des ordres de voyage valides

Besoins :

Identifier si une phrase donnée représente un ordre de voyage valide ou non.

Discriminer les phrases liées aux trajets des phrases sans rapport, en tenant compte de la diversité linguistique.

Solutions apportées :

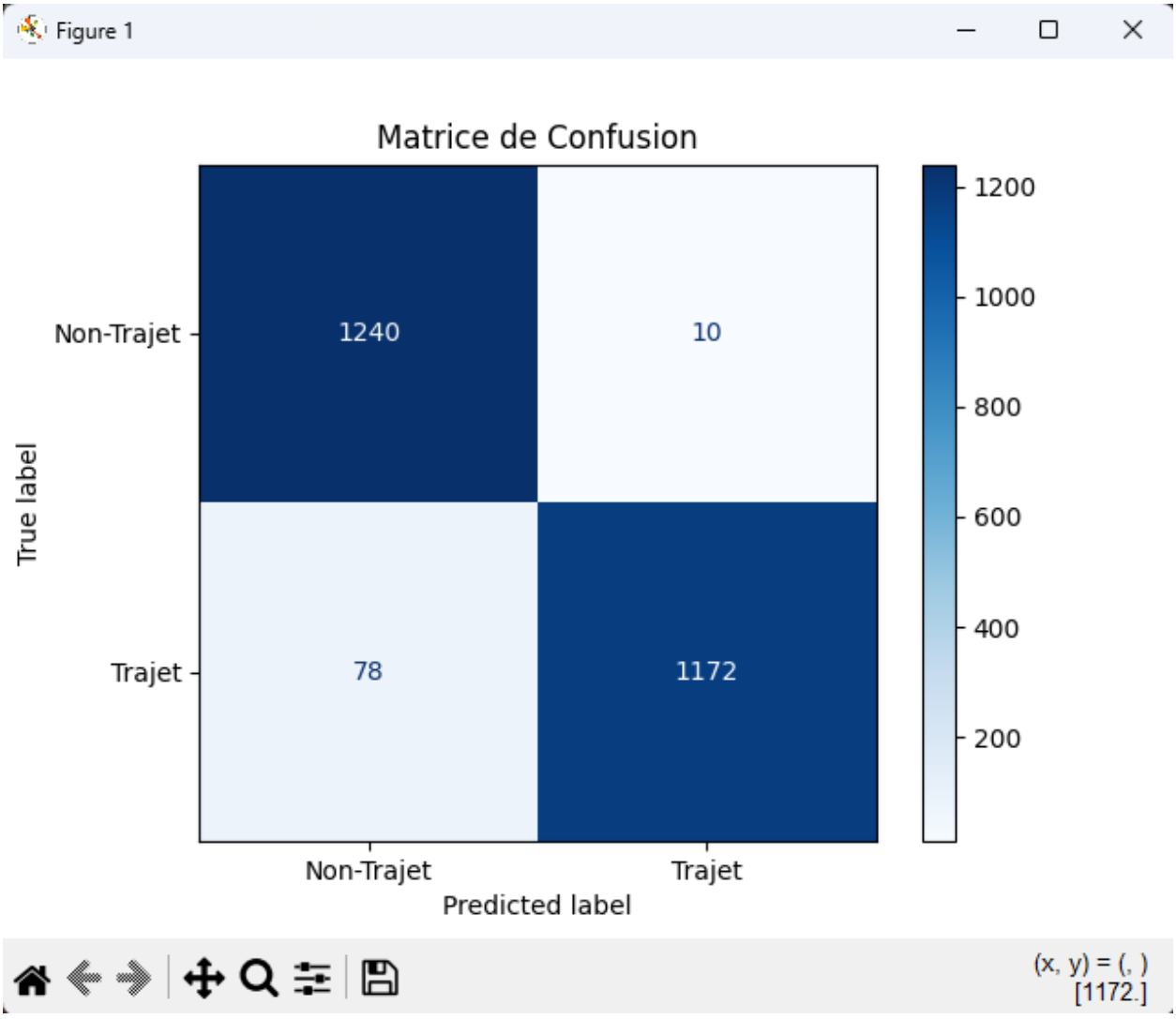
Développement d'un modèle BERT fine-tuné pour classifier les phrases en deux catégories : VALID_TRIP ou INVALID_TRIP.

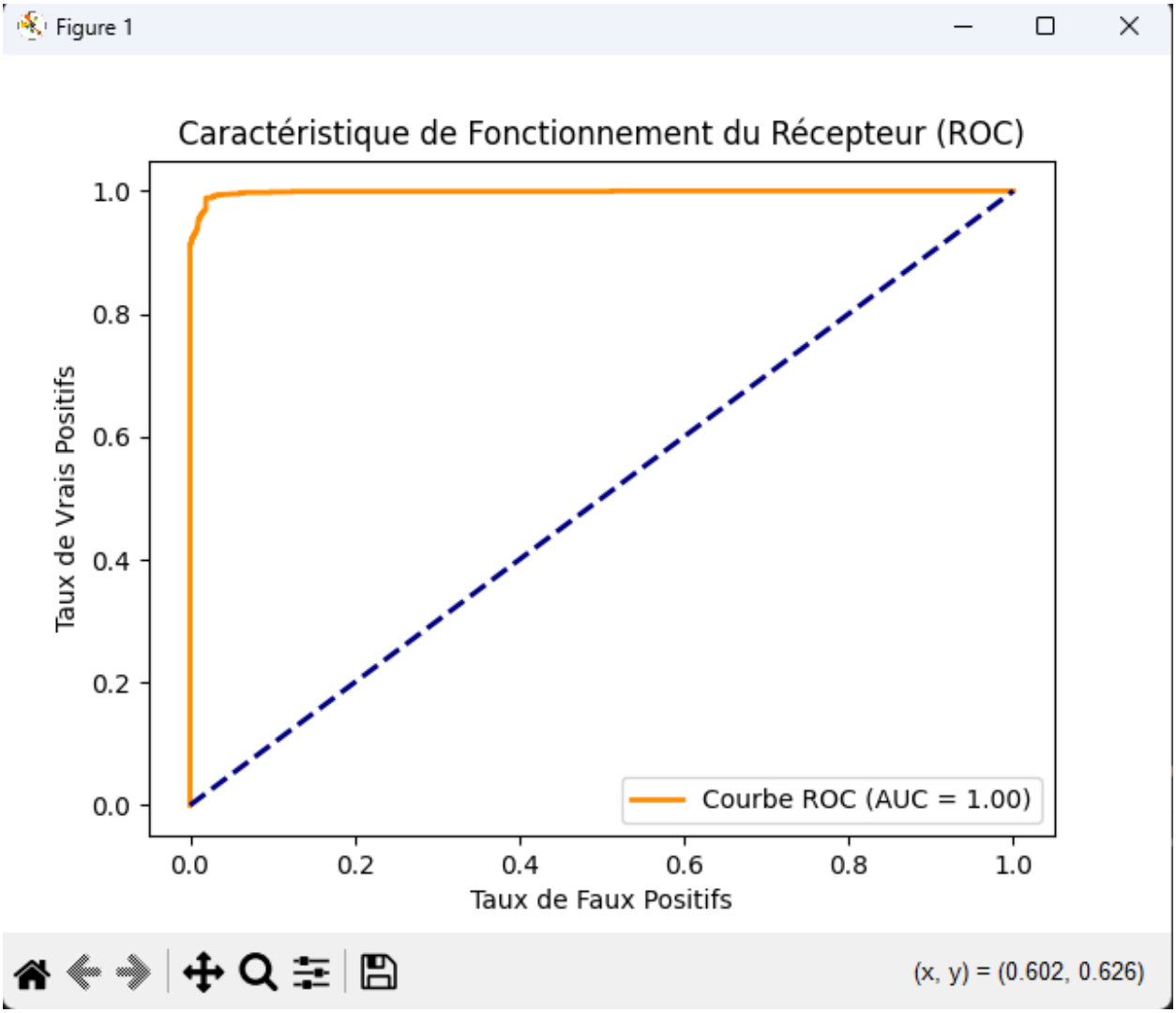
Implémentation de la fonction estTrajet() pour traiter les phrases en entrée et renvoyer une classification binaire.

Génération d'un ensemble de données d'entraînement varié à l'aide d'un script Python, en combinant des modèles syntaxiques diversifiés, des synonymes, et des villes aléatoires.

Test du modèle entraîné :

	precision	recall	f1-score	support
Non-Trajet	0.94	0.99	0.97	1250
Trajet	0.99	0.94	0.96	1250
accuracy			0.96	2500
macro avg	0.97	0.96	0.96	2500
weighted avg	0.97	0.96	0.96	2500





2. Extraction des villes de départ, d'arrivée, et des étapes intermédiaires

Besoins :

Identifier avec précision les lieux mentionnés dans une phrase, y compris la ville de départ, la ville d'arrivée, et les éventuelles étapes intermédiaires.

Gérer les structures grammaticales variées et complexes utilisées dans les ordres de voyage.

Solutions apportées :

Utilisation de SpaCy pour la reconnaissance des entités nommées (NER).

Développement de motifs personnalisés avec le module Matcher de SpaCy pour détecter les expressions liées aux trajets, comme "depuis", "à", "en passant par".

Extraction et normalisation des lieux détectés grâce à une correspondance avec les communes et gares disponibles dans les données SNCF.

3. Intégration des données SNCF

Besoins :

Valider les villes extraites en vérifiant leur correspondance avec les données officielles des gares SNCF.

Assurer une intégration fluide des informations nécessaires pour calculer des itinéraires optimaux.

Solutions apportées :

Chargement des données SNCF depuis un fichier CSV et création d'une structure de mappage entre les communes et les gares associées.

Vérification systématique des villes extraites pour s'assurer qu'elles figurent parmi les communes valides avant de poursuivre le traitement.

Possibilité d'utiliser les données pour générer des trajets optimisés entre les villes valides.

4. Gestion des contraintes

Besoins :

Rejeter les phrases mentionnant des véhicules non autorisés ou des modes de transport bannis.

Exclure les villes non françaises si elles ne sont pas présentes dans les données des gares SNCF.

Solutions apportées :

Implémentation de la fonction is_banned_vehicle() pour détecter et rejeter les phrases contenant des mots-clés associés à des véhicules interdits (par exemple, "moto", "scooter").

Normalisation et validation des villes mentionnées pour exclure celles qui ne figurent pas dans les données des communes françaises intégrées.

5. Génération de données d'entraînement pour le modèle NLP

Besoins :

Disposer d'un large corpus de phrases diversifiées pour entraîner le modèle NLP et le modèle de classification.

Inclure à la fois des phrases valides (ordres de voyage) et des phrases non valides.

Solutions apportées :

Développement d'un script Python pour générer dynamiquement des phrases variées en combinant des templates, des synonymes, des villes (françaises et internationales), et des modes de transport.

Création d'ensembles d'entraînement et de test distincts pour garantir une validation efficace du modèle.

Incorporation de phrases aléatoires avec des structures grammaticales complexes pour simuler la diversité des ordres de voyage réels.

6. Ré-entraînement du modèle SpaCy pour intégrer les communes SNCF

Besoins :

Adapter le modèle SpaCy pour inclure les noms de communes présents dans les données SNCF, afin d'améliorer la précision de la reconnaissance des lieux.

Ajuster le modèle pour gérer des phrases avec des erreurs de formatage, des abréviations ou des variations orthographiques.

Solutions apportées :

Ré-entraînement du modèle SpaCy en enrichissant le jeu de données avec les communes issues des données SNCF.

Création de nouvelles étiquettes pour les entités nommées afin de différencier les villes de départ, d'arrivée, et les étapes intermédiaires.

Tests approfondis pour évaluer la précision du modèle après l'intégration des nouvelles données.

3- Conclusion

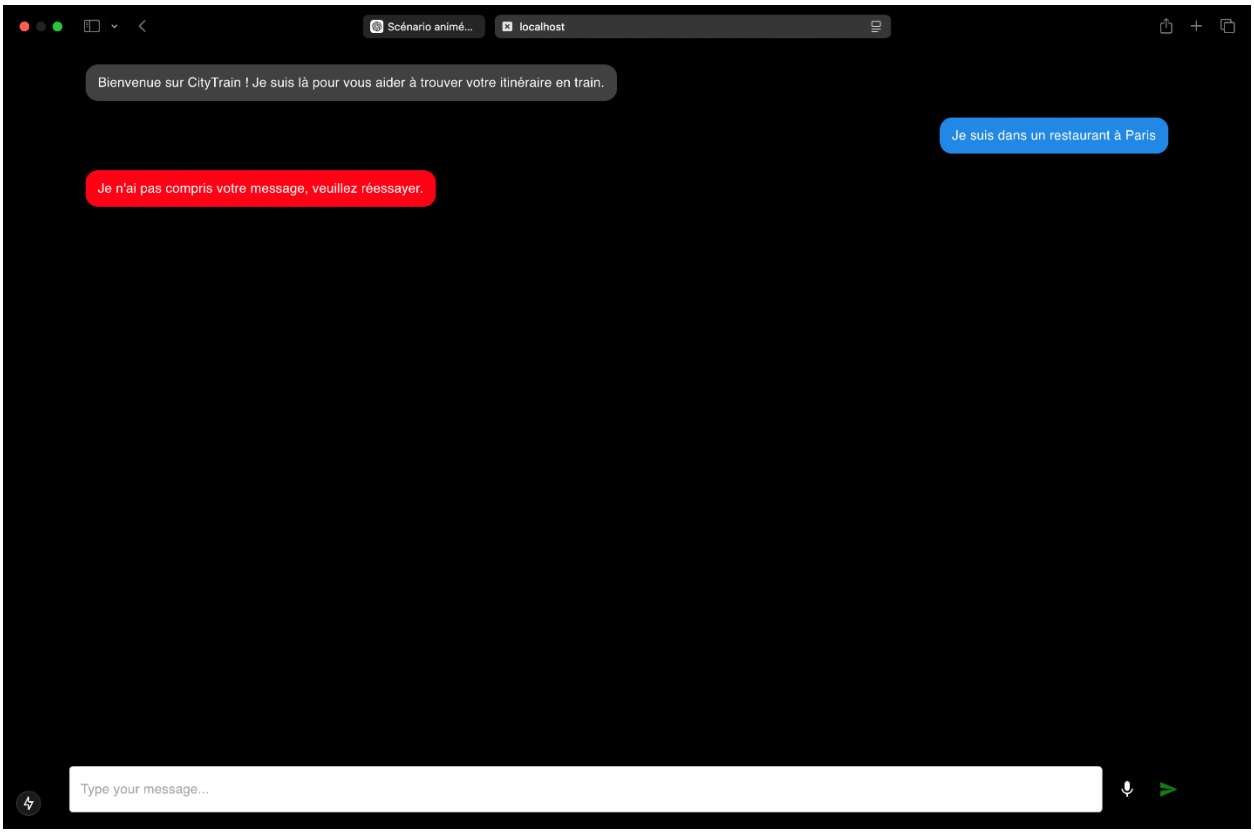
Le projet a démontré sa capacité à gérer une grande diversité de phrases formulées en français pour identifier et extraire des informations liées aux trajets. Grâce à l'intégration de techniques avancées de traitement du langage naturel (NLP) et de modèles entraînés sur des données spécifiques, la solution est capable de discriminer les ordres de voyage valides des phrases non pertinentes, tout en identifiant avec précision les villes de départ, d'arrivée, ainsi que les étapes intermédiaires.

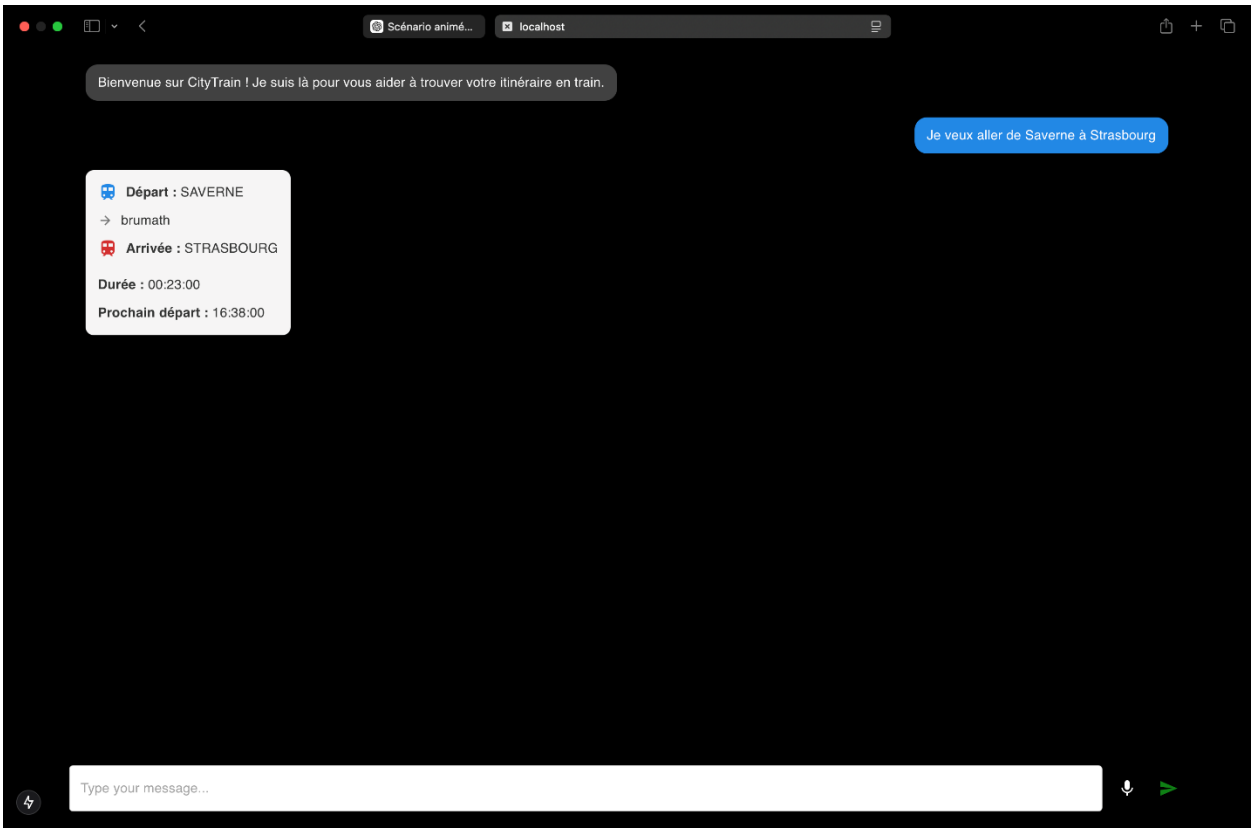
L'utilisation d'un modèle SpaCy ré-entraîné et d'un modèle BERT fine-tuné a permis de capturer les nuances linguistiques et de traiter efficacement des structures grammaticales variées. L'intégration des données des gares SNCF garantit la fiabilité des informations extraites et permet de valider les trajets

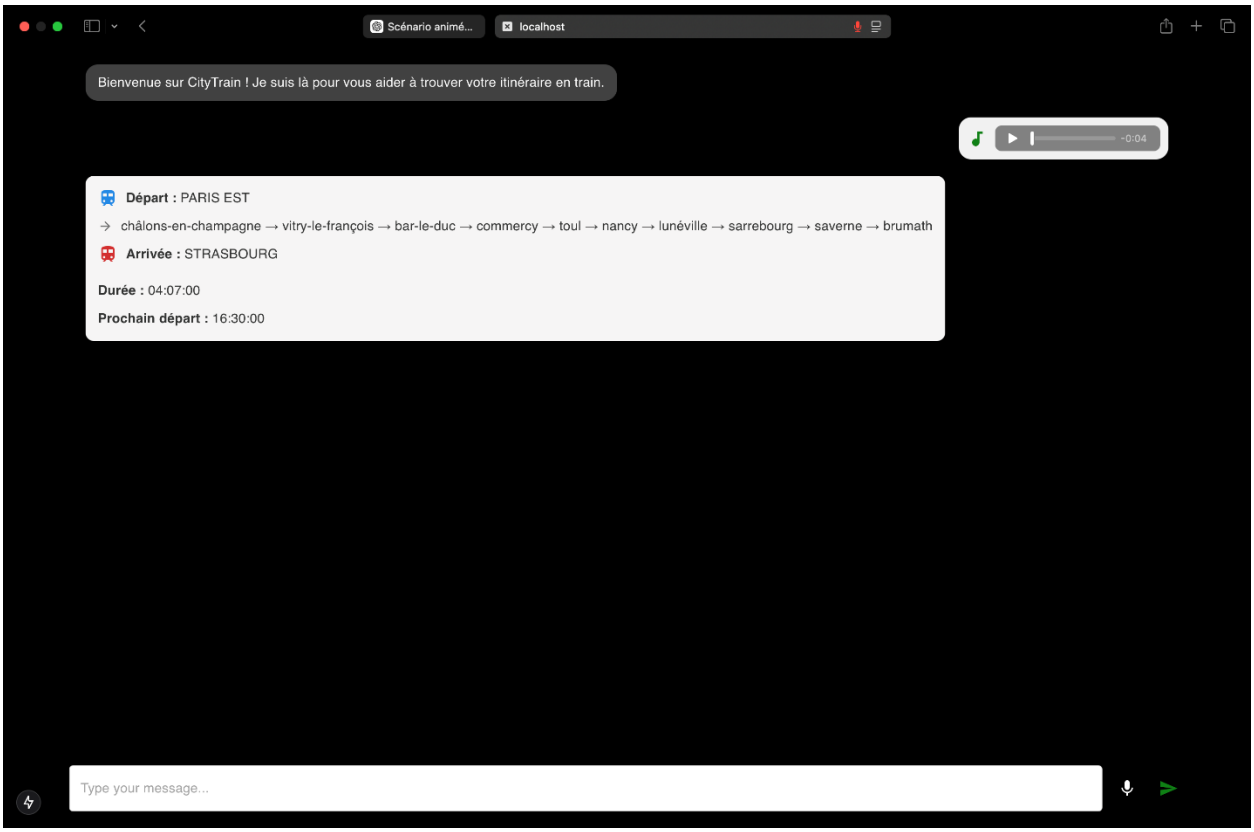
de manière robuste. De plus, la génération dynamique de phrases d’entraînement a renforcé la diversité et la généralisation du modèle, offrant ainsi une solution adaptable à des cas d'utilisation variés.

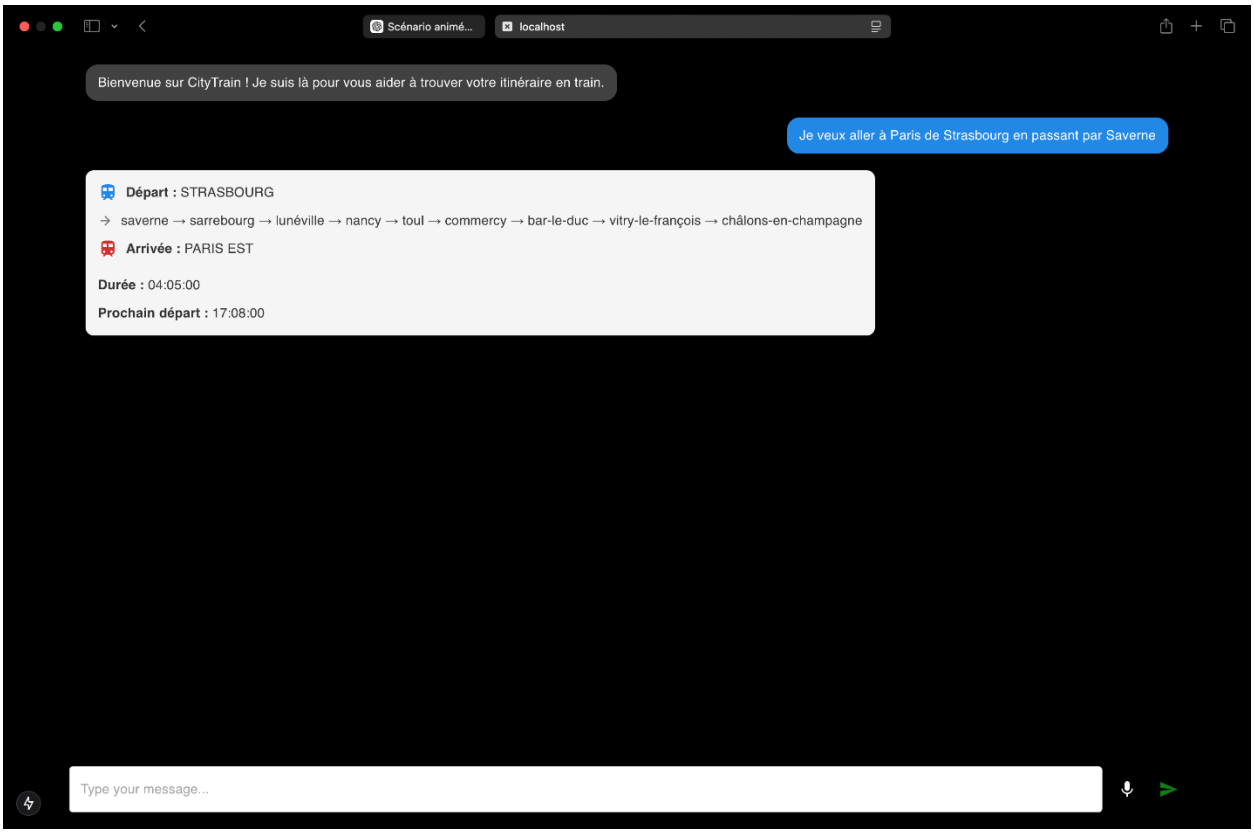
En perspective, des améliorations peuvent être envisagées, notamment au niveau de la variété des phrases sachant que qu’il prend déjà une grande variété de phrase complexe.

4- Image de l’application finale :




Scénario animé...
localhost
Bienvenue sur CityTrain ! Je suis là pour vous aider à trouver votre itinéraire en train.
Je veux aller de Saverne à Strasbourg
Départ : SAVERNE
brumath
Arrivée : STRASBOURG
Durée : 00:23:00
Prochain départ : 16:38:00
Type your message...


Scénario animé...
localhost
Bienvenue sur CityTrain ! Je suis là pour vous aider à trouver votre itinéraire en train.
-0:04
Départ : PARIS EST
→ châlons-en-champagne → vitry-le-françois → bar-le-duc → commercy → toul → nancy → lunéville → sarrebourg → saverne → brumath
Arrivée : STRASBOURG
Durée : 04:07:00
Prochain départ : 16:30:00
Type your message...


Scénario animé...
localhost
Bienvenue sur CityTrain ! Je suis là pour vous aider à trouver votre itinéraire en train.
Je veux aller à Paris de Strasbourg en passant par Saverne
Départ : STRASBOURG
→ saverne → sarrebourg → lunéville → nancy → toul → commercy → bar-le-duc → vitry-le-françois → châlons-en-champagne
Arrivée : PARIS EST
Durée : 04:05:00
Prochain départ : 17:08:00
Type your message...