Titre et Synthèse Exécutive

MedBot : Système de Classification d'Intention pour la Santé

MedBot : Système de Classification d'Intention pour la Santé

- Application d'un **Réseau Neuronal Dense (PyTorch)** sur Features **TF-IDF**.

- Automatisation de l'assistance de premier niveau garantissant une classification rapide et fiable

Auteur : Antonine Pelicier // Steve Calixte

Github : https://github.com/apelicier // https://github.com/Poincare008

LinkedIn : https://www.linkedin.com/in/antonine-pelicier **//**

Rôle: Développeur AI and Data scientist

Enjeu Métier et Proposition de Valeur

Efficacité Opérationnelle et Fiabilité

Efficacité Opérationnelle et Fiabilité

- **Problématique :** Désengorgement des services de santé dû au volume élevé de requêtes routinières.
- **Solution Ciblée :** Implémenter un système de **NLU (Natural Language Understanding)** pour la catégorisation en temps réel.
- **Proposition de Valeur :** Réponse instantanée et fiabilité de la classification des intentions.
- **KPI Clé :** Précision de Classification d'Intention >90% | Taux d'automatisation des réponses initiales.

Architecture Modulaire du Pipeline

Architecture Modulaire du Pipeline Data-Centrique

Architecture Modulaire du Pipeline Data-Centrique

- **Diagramme de Flux :** INPUT (Texte) →NLP Pipeline→Modèle PyTorch →OUTPUT (Réponse).
- **Étapes Clés du Pré-traitement :**

1. Acquisition de données structurées (healthcare_intents.json).
2. Tokenization et Suppression des Stop Words.
3. Normalisation par **Lemmatisation (NLTK)** pour minimiser la variance lexicale.

Pré-traitement et Feature Engineering (NLP)

Représentation Sémantique par TF-IDF

Représentation Sémantique par TF-IDF

- **Méthode :** Utilisation du **TF-IDF** (Term Frequency-Inverse Document Frequency) pour la vectorisation
- **Raisonnement :** Crée une **représentation vectorielle discriminante** en pondérant l'importance des termes rares du vocabulaire.
- **Sortie :** Matrice d'entrée X appartient a l’emsemble R pour le réseau.
- **Artefact :** L'instance du vectoriseur est sauvegardée dans le fichier **.pkl** (meta_tfidf.pkl).

Architecture du **ChatbotModel** (PyTorch)

Architecture du Modèle de Deep Learning

Architecture du ChatbotModel (PyTorch)

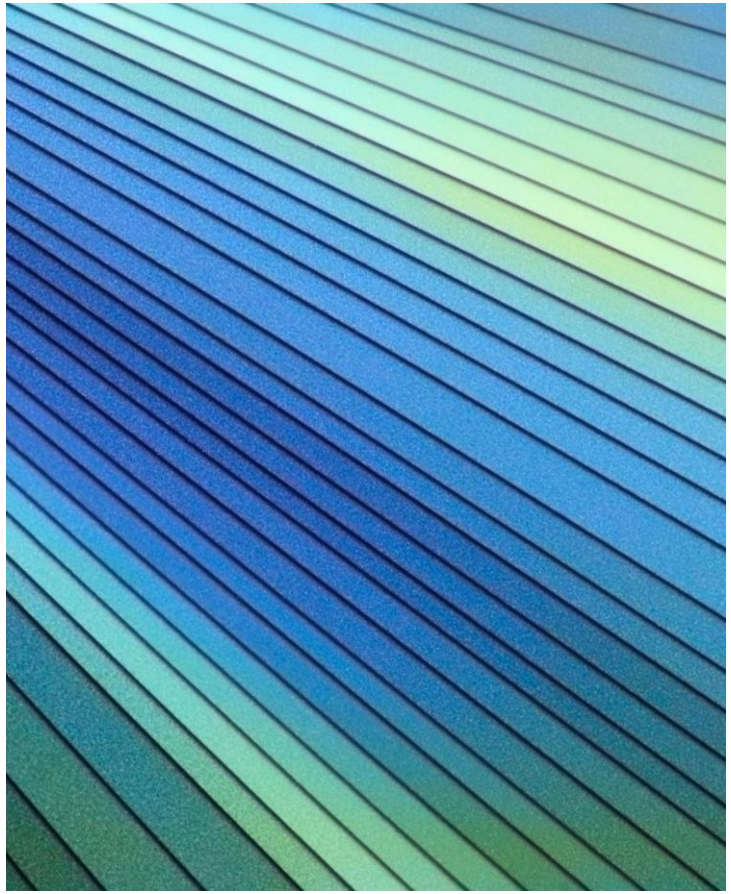
Cadre : MLP (Multi-Layer Perceptron) implémenté via nn.Module

Topologie : Réduction progressive des couches :

- Input → 512 → 256 → 128 → 64 → Output

Optimisation & Régularisation :

- **Batch Normalization** pour la stabilité et la vitesse.
- **Dropout** (0.4) pour prévenir le surapprentissage.



Stratégie d'Entraînement et Validation

Stratégie d'Entraînement et de Validation Rigoureuse

Stratégie d'Entraînement et Validation Rigoureuse

- **Hyperparamètres Clés :**
  - **Loss :** nn.CrossEntropyLoss.
  - **Optimiseur :** optim.Adam.
  - **Gestion du Learning Rate :** StepLR Scheduler.

- **Validation :** Partitionnement Train/Test (80/20) avec **Stratification** pour garantir une distribution équilibrée des classes.
- **Référence Code :** Logique d'entraînement dans le fichier ***Chatbot.ipynb***

Performances et Métriques

Résultats et Métriques de Performance (Test Set)

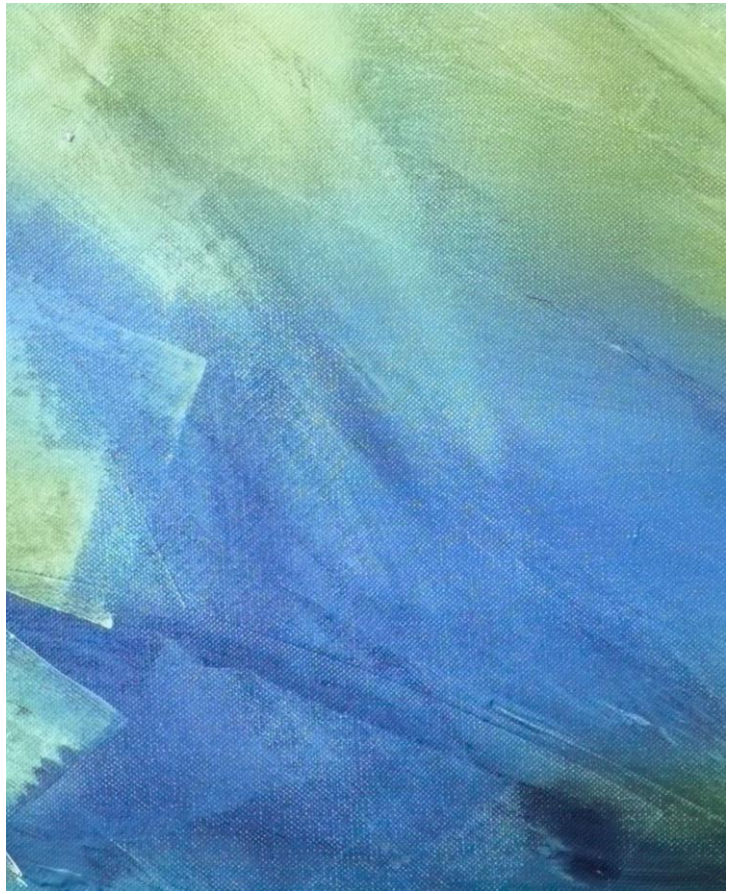
Résultats et Métriques de Performance (Test Set)

Convergence : Graphique de la Courbe de Perte
→Convergence rapide et stable sur **50 époques**.

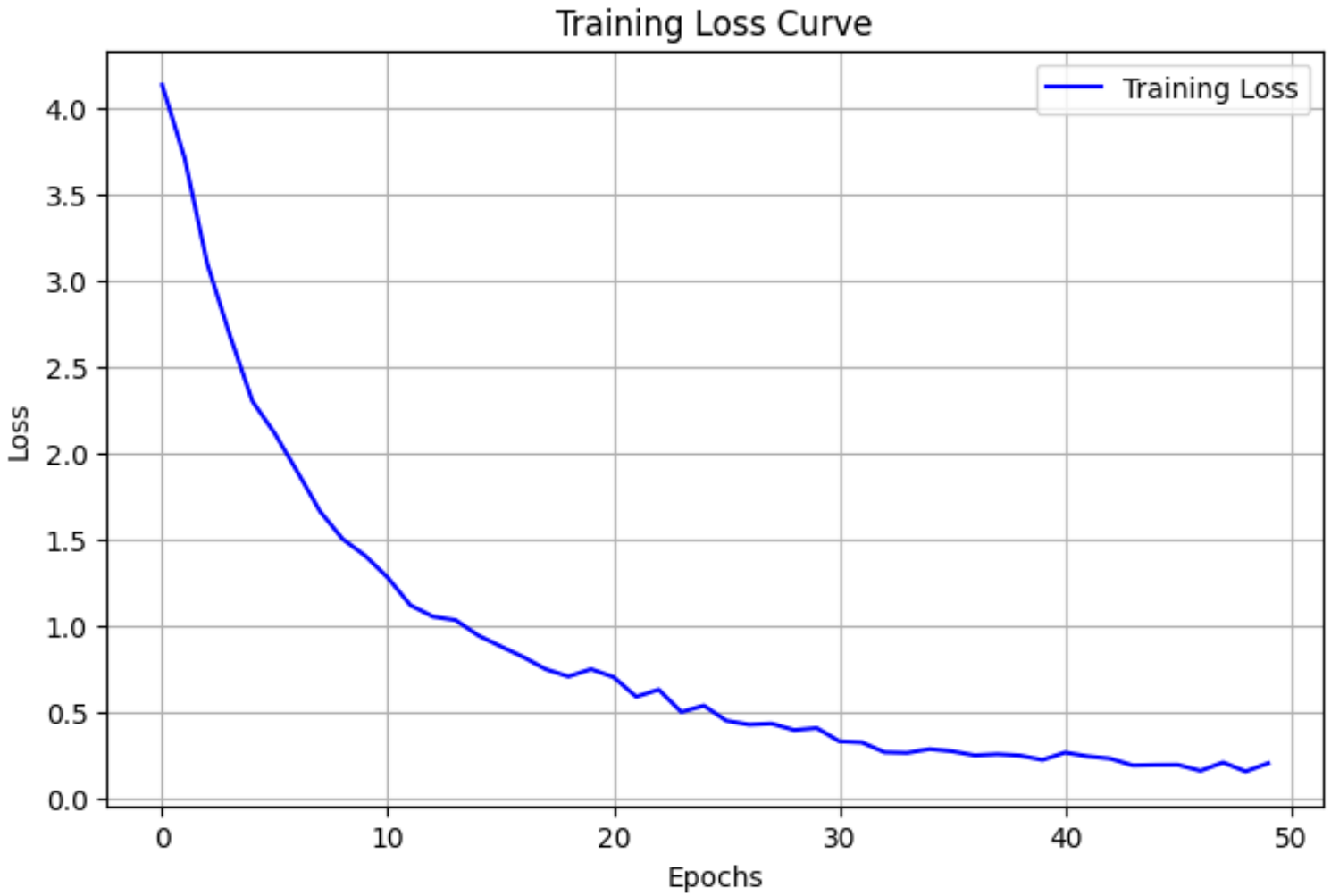
Précision Globale : Accuracy sur l'Ensemble de Test

Analyse Détaillée : Rapport de Classification (F1-Score, Précision, Rappel) élevé sur l'ensemble des intentions, validant la robustesse.

Artefact : Poids entraînés sauvegardés dans le fichier **.pth** (chat_model_tfidf.pth).



Graphique de la Courbe de Perte



Logique de Décision et Résilience

Résilience et Logique de Décision (MLOps)

Résilience et Logique de Décision (MLOps)

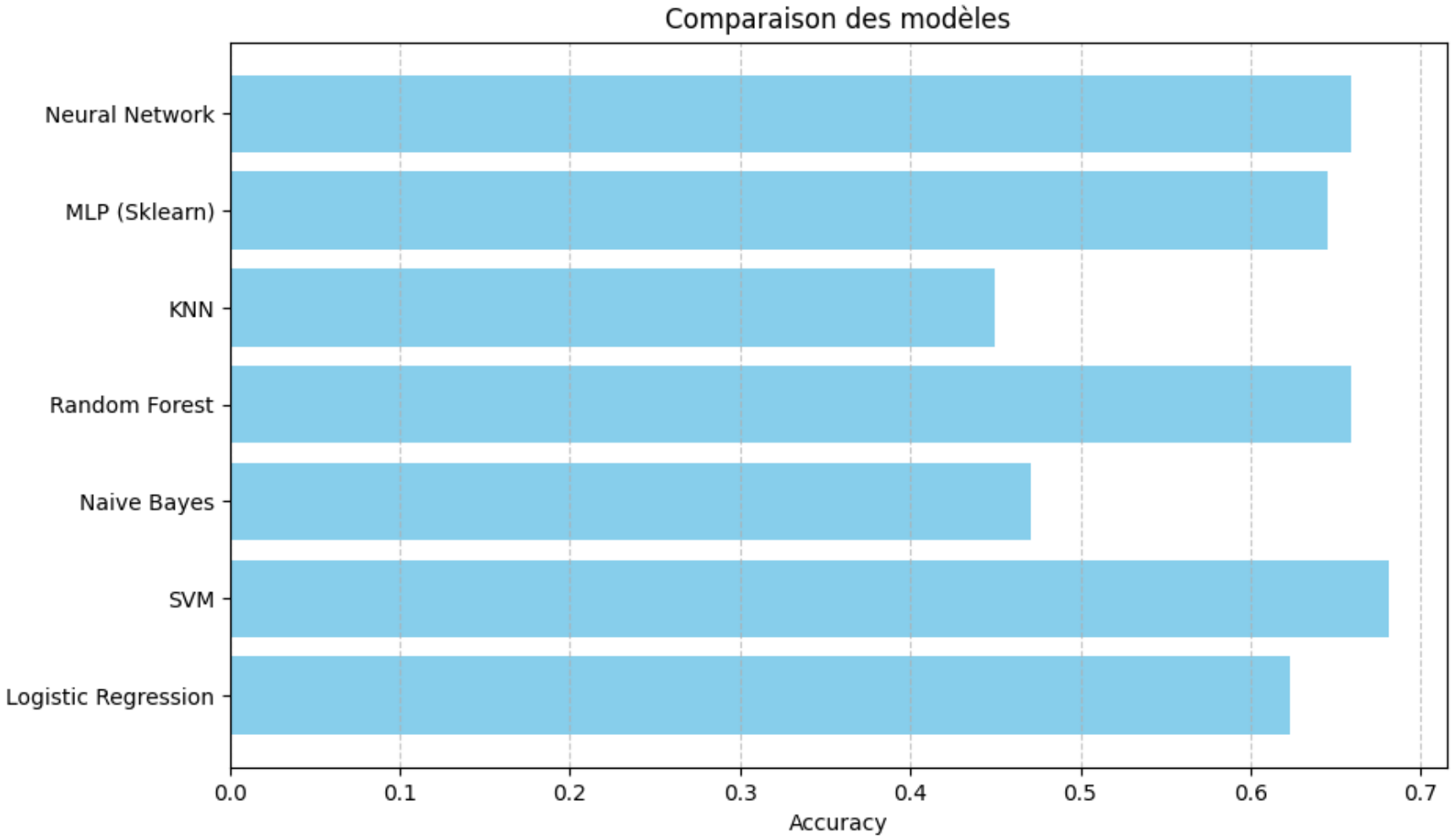
- **Mécanisme :** Probabilités calculées par **Softmax** → Classification basée sur la Confiance Maximale.
- **Seuil de Confiance :** Implémentation d'un **Seuil de 0.65** pour garantir la fiabilité.
- **ournalisation de l'Incertitude :** Requêtes sous le seuil stockées dans uncertain_inputs.log pour la **supervision humaine et le ré-entraînement (MLOps)**.

Choix du meilleur modèle

Après avoir comparé plusieurs modèles sur la base de leur taux de précision (Accuracy), nous avons décidé d'entraîner nos données à l'aide de deux modèles : un réseau de neurones (Neural Network) et une machine à vecteurs de support (SVM).

- Le SVM a été retenu car il a obtenu la meilleure précision globale parmi les modèles testés.
- Le Neural Network a été conservé (performances proches du SVM, modèle initialement prévu dans la conception de notre chatbot).

Choix du meilleur modèle



Choix du meilleur modèle

```
 Résumé de la comparaison des modèles:
Logistic Regression         → 0.6232
SVM                         → 0.6812
Naive Bayes                 → 0.4710
Random Forest               → 0.6594
KNN                         → 0.4493
MLP (Sklearn)               → 0.6449
Neural Network              → 0.6594

 Best model: SVM with accuracy = 0.6812
```

Déploiement et Expérience Utilisateur

Déploiement de l'Interface Utilisateur (UX)

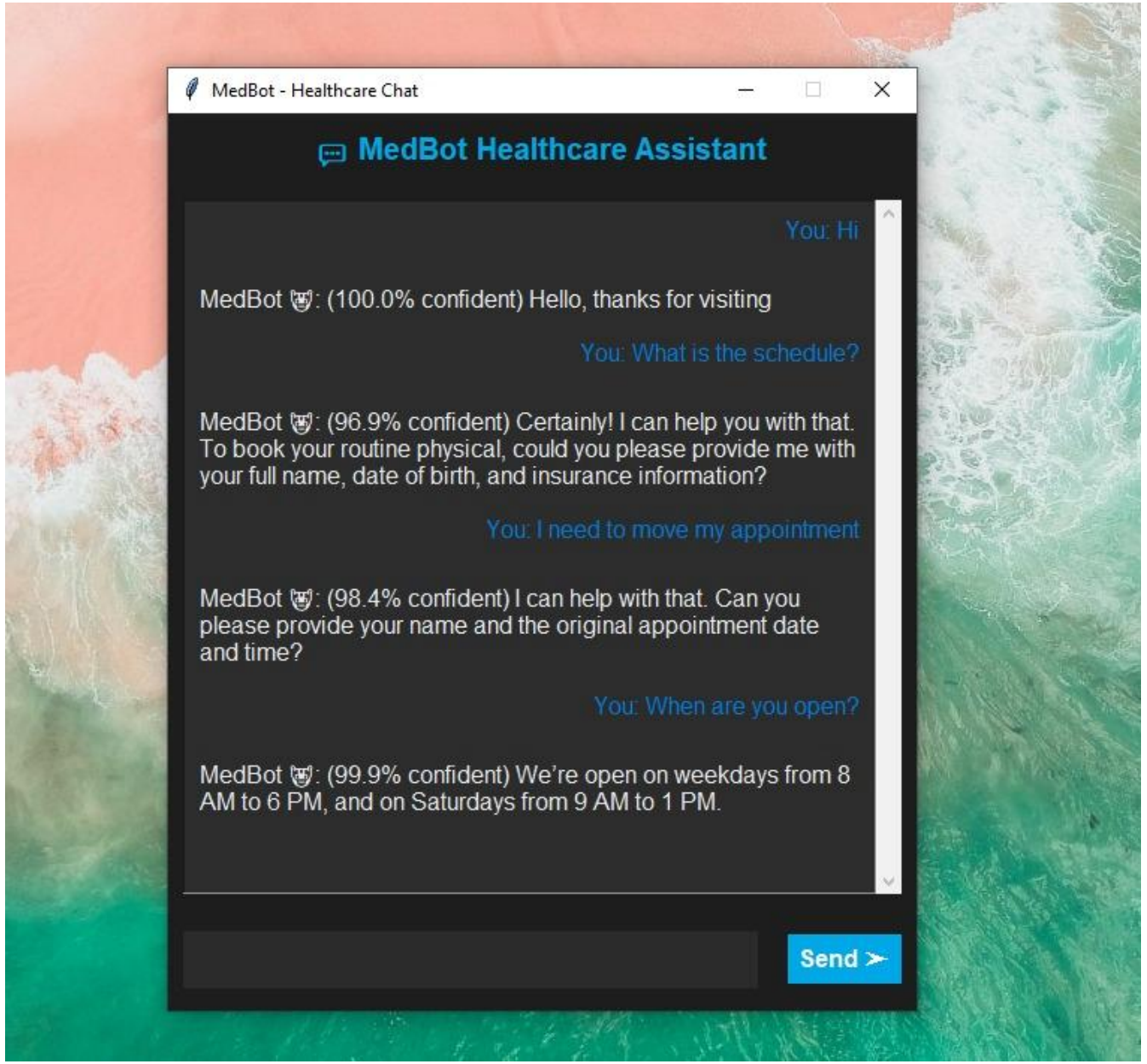
L'Interface Utilisateur (UX)

Interface : Application cliente **ModernChatApp** développée en **Tkinter** (*stand-alone*).

Performance Asynchrone : Utilisation du **Threading Python** pour l'exécution non-bloquante de la prédiction du modèle.

Avantage UX : Le GUI reste réactif (affichage "typing...") pendant que la prédiction s'exécute en arrière-plan.

L'Interface Utilisateur (UX)



Conclusion et Feuille de Route (Roadmap)

Conclusion et Feuille de Route Stratégique

Conclusion et Feuille de Route Stratégique

- **Synthèse :** Validation du système de classification d'intention Deep Learning haute performance.
- **Roadmap Technique : Migration vers des Word Embeddings (Word2Vec, FastText) pour une capture sémantiqu**e plus riche (remplacement de TF-IDF).
- **Roadmap Produit :** Déploiement en **Micro-service** (Flask/Streamlit) et intégration de la logique de *Tool-Use* (appel d'API pour actions concrètes).

Limite de notre ChatBot

- **Dataset**
- **Limites dans le Traitement du Langage Naturel (NLP):** C'est la limitation la plus significative, car elle est inhérente à la méthode TF-IDF.
- **Limites du Modèle (Architecture MLP):** Bien que le MLP soit efficace, il a des contraintes par rapport aux architectures séquence-à-séquence.
- **Limites Opérationnelles et Évolutives**
- **Limites du Modèle SVM:** C’est un modèle de Machine Learning classique et puissant. Ses limites sont principalement liées à sa **nature non-Deep Learning** et à sa performance face à des problèmes complexes ou des jeux de données très volumineux.

Possibilité d’amélioration

- **Disponible dans au moins 3 langues(Créole/Français/Anglais)**
- **Amélioration du Traitement du Langage Naturel (NLP):** Le point de blocage actuel est le **TF-IDF (Bag-of-Words)**. La première amélioration est de remplacer cette technique par une méthode qui capture la sémantique et le contexte.
- **Migration** vers des **Word Embeddings** afin d'enrichir la représentation des données pour le modèle PyTorch.

Source d’inspiration

1

https://chatbotsmagazine.com/contextual-chat-bots-with-tensorflow-4391749d0077

2

https://youtu.be/a040VmmO-AY?si=9IJ82qXV6B5TRzzv

3

https://github.com/patrickloeber/pytorch-chatbot

4

https://youtu.be/RpWeNzfSUHw?si=KF1174rAoIEus9nb

Thank you