**Chatbot basé sur DialoGPT**

Introduction

**Contexte**

Dans un monde de plus en plus axé sur la technologie, l'interaction homme-machine joue un rôle essentiel dans de nombreux domaines, tels que le service client, l'éducation, et les systèmes d'assistance. Les chatbots, en particulier, sont devenus un outil populaire pour automatiser les interactions, améliorer l'accessibilité des services et offrir une assistance en temps réel.

Ce projet vise à tirer parti des avancées en intelligence artificielle pour créer un chatbot capable de répondre de manière pertinente et engageante aux utilisateurs. En exploitant un modèle de traitement du langage naturel (NLP) avancé tel que DialoGPT, le chatbot offre une expérience de conversation fluide et réaliste.

**Objectif**

L'objectif principal de ce projet est de développer un chatbot interactif basé sur l'IA qui :

* Fournit des réponses contextuelles et naturelles aux utilisateurs.
* Offre une interface utilisateur intuitive et moderne pour simplifier les interactions.
* Montre le potentiel des modèles de langage pré-entraînés dans les applications réelles.

Ce projet a également pour but d'améliorer la compréhension des technologies NLP et des frameworks web, tout en explorant les limites actuelles des systèmes conversationnels.

**Technologies utilisées**

Pour mener à bien ce projet, plusieurs technologies et outils modernes ont été employés :

* **Python** : Langage principal utilisé pour implémenter le backend du chatbot.
* **Flask** : Framework léger pour développer l'application web et gérer les communications entre le frontend et le backend.
* **DialoGPT** : Modèle de langage pré-entraîné de Microsoft, spécialisé dans les conversations basées sur les données de chat.
* **Torch** : Bibliothèque de calcul scientifique utilisée pour charger et exécuter le modèle DialoGPT.
* **Transformers** : Bibliothèque Python développée par Hugging Face pour travailler avec des modèles NLP avancés.
* **HTML/CSS** : Pour la structure et le style de l'interface utilisateur.
* **JavaScript (jQuery)** : Pour ajouter des fonctionnalités interactives, telles que l'envoi de messages en temps réel et la gestion des réponses.
* **GitHub** : Pour le suivi des versions et la collaboration.

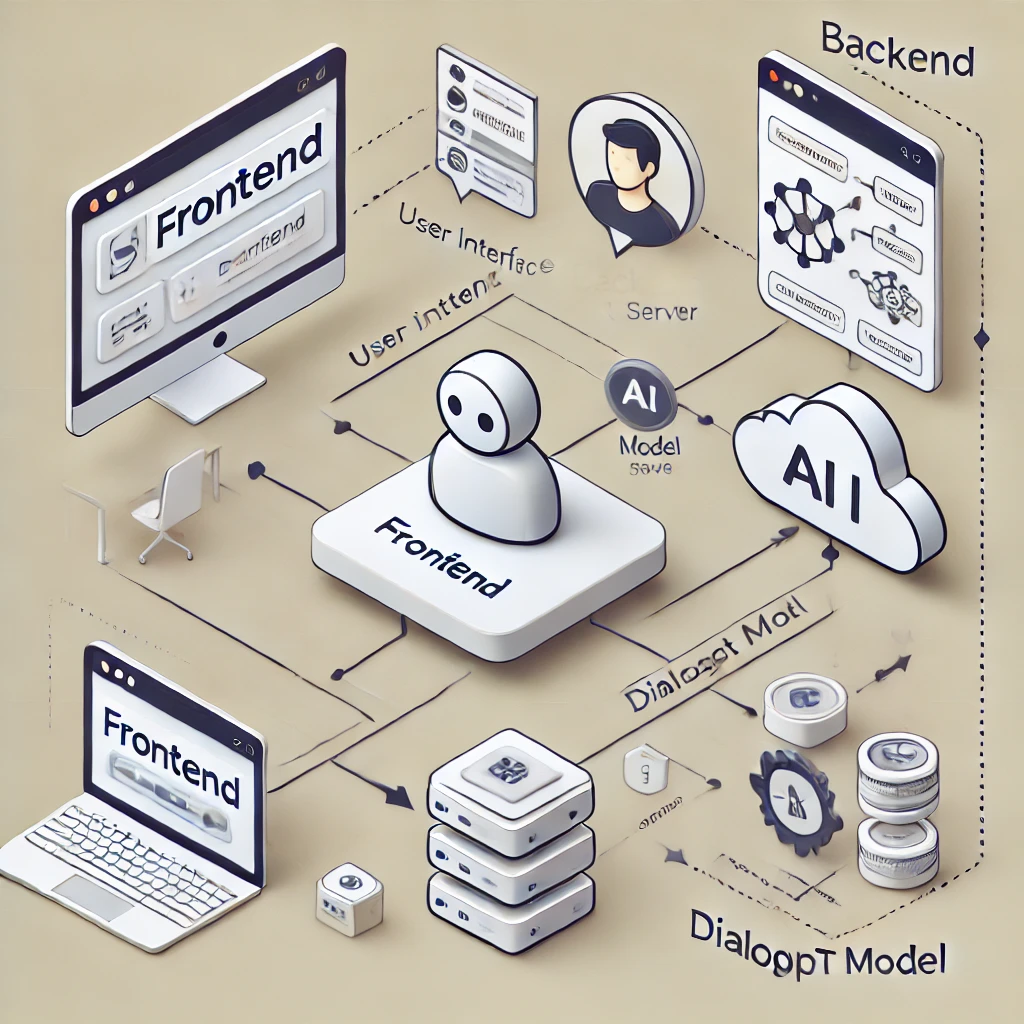
Ces outils permettent de créer un système robuste et extensible, tout en garantissant une expérience utilisateur optimale.

Conception et Architecture

**Architecture générale**

L'architecture du projet repose sur une communication fluide entre trois composants principaux :

1. **L'interface utilisateur (Frontend)** : Permet aux utilisateurs de poser des questions et de visualiser les réponses du chatbot.
2. **Le Backend (Flask)** : Sert de pont entre l'interface utilisateur et le modèle DialoGPT. Il reçoit les messages, les transmet au modèle IA, et retourne les réponses au frontend.
3. **Le Modèle DialoGPT** : Génère des réponses conversationnelles en utilisant un modèle de langage naturel avancé.



🡪 L'utilisateur saisit un message dans l'interface utilisateur.

🡪 Le frontend transmet le message au backend Flask via une requête POST.

🡪 Flask traite la requête et transmet le message au modèle DialoGPT.

🡪 DialoGPT génère une réponse, qui est renvoyée au backend Flask.

🡪 Le backend transmet la réponse au frontend, où elle est affichée.

**Composants principaux**

**Frontend**

L'interface utilisateur est conçue pour être intuitive et agréable, permettant des interactions fluides :

* **HTML** : Structure de base de la page.
* **CSS** : Stylisation pour rendre l'application attrayante et conviviale.
* **JavaScript (jQuery)** : Ajoute des fonctionnalités dynamiques telles que l'envoi de messages en temps réel et l'affichage des réponses sans rechargement de la page.

**Backend (Flask)**

* Le serveur Flask agit comme intermédiaire entre le frontend et le modèle DialoGPT.
* Il reçoit les messages utilisateur via des requêtes HTTP, traite ces données, et génère une réponse en utilisant le modèle.
* Flask est léger et adapté à ce type d'application où les interactions sont simples mais doivent être rapides.

**Modèle IA (DialoGPT)**

* **microsoft/DialoGPT-medium** : Ce modèle de langage, développé par Microsoft, est pré-entraîné sur des conversations et peut fournir des réponses naturelles et pertinentes.
* **Hugging Face Transformers** : Utilisé pour charger et manipuler le modèle DialoGPT.
* **Torch** : Bibliothèque utilisée pour les calculs liés au modèle.

**Implémentation**

**Configuration de l'environnement :**

Pour développer le chatbot, un environnement structuré a été mis en place:

1. **Installation des outils de base :**
   * Installer [Python](https://www.python.org/) (version 3.8 ou supérieure).
   * Installer les bibliothèques nécessaires :

pip install flask transformers torch jquery

1. **Mise en place de l’interface web :**
   * Créer des fichiers HTML, CSS, et JavaScript pour une interface utilisateur intuitive.
   * Configurer Flask pour exécuter un serveur local et rendre l’interface accessible.
2. **Téléchargement et configuration du modèle IA :**
   * Télécharger et configurer le modèle pré-entraîné DialoGPT-medium à l’aide de la bibliothèque transformers.

**Description des scripts principaux :**

**Backend Flask**

Le backend est le cœur du système, gérant la communication entre l’interface utilisateur et le modèle IA.

* Routes principales :
  + / : Charge l’interface utilisateur (fichier HTML principal).
  + /chat : Traite les messages envoyés par l’utilisateur et renvoie les réponses générées.

Exemple de code pour le backend Flask :

@app.route("/")

def home():

return render\_template("index.html")

@app.route("/chat", methods=["POST"])

def chat():

user\_message = request.json.get("message")

if not user\_message:

return jsonify({"response": "Je n'ai pas compris votre question. Veuillez réessayer."})

try:

response = chatbot.generate\_response(user\_message)

return jsonify({"response": response})

except Exception as e:

return jsonify({"response": f"Erreur : {str(e)}"})

* Gestion des erreurs :
  + Gestion des cas où aucun message utilisateur n’est envoyé.
  + Affichage d’un message d’erreur clair en cas de problème technique.

**Modèle IA**

Le modèle DialoGPT gère la logique de génération des réponses.

* Chargement et personnalisation :
  + Utilisation du modèle pré-entraîné microsoft/DialoGPT-medium.
  + Implémentation d’un système de suivi de l’historique pour maintenir la cohérence des conversations.

Exemple :

input\_ids = self.tokenizer.encode(user\_message + self.tokenizer.eos\_token, return\_tensors="pt")

if self.chat\_history\_ids is not None:

input\_ids = torch.cat([self.chat\_history\_ids, input\_ids], dim=-1)

self.chat\_history\_ids = self.model.generate(

input\_ids, max\_length=1000, temperature=0.7, top\_k=50, top\_p=0.9)

* Personnalisation des réponses conditionnelles : Certaines réponses sont explicitement définies pour les questions fréquentes.

if "what's your name" in user\_message.lower():

return "My name is DialoGPT!"

**Frontend**

Le frontend fournit une interface utilisateur interactive.

* Utilisation de jQuery pour AJAX :
  + Permet l’envoi et la réception asynchrones de messages sans recharger la page.

$("#send-btn").click(function () {

const userInput = $("#user-input").val().trim();

if (!userInput) return;

$.ajax({

url: "/chat",

method: "POST",

contentType: "application/json",

data: JSON.stringify({ message: userInput }),

success: function (response) {

$("#messages").append(`<div><strong>Chatbot:</strong> ${response.response}</div>`);

},

});

});

* Mise en forme des messages :
  + Les messages sont stylisés pour une expérience utilisateur agréable grâce à CSS :

#messages {

height: 300px;

overflow-y: scroll;

border: 1px solid #ddd;

padding: 10px;

background-color: #f9f9f9;

}

**Améliorations spécifiques :**

🡪 **Personnalisation des réponses conditionnelles :**

* Ajout de réponses directes à des questions fréquentes pour une meilleure expérience utilisateur.

🡪 **Optimisation des hyperparamètres :**

* Ajustement des paramètres comme temperature, top\_k, et top\_p pour obtenir des réponses plus cohérentes.

🡪 **Amélioration de l’interface utilisateur :**

* Design moderne et intuitif pour encourager les interactions utilisateur.

🡪 **Système de gestion de l’historique :**

* Permet au modèle de tenir compte du contexte précédent pour fournir des réponses plus pertinentes.

Résultats

Une image contenant texte, capture d’écran, logiciel, Icône d’ordinateur

Description générée automatiquement