Documentation du Bot Telegram avec Intégration Langflow

Cette documentation fournit une vue d'ensemble détaillée d'un bot Telegram intégré avec Langflow pour traiter les messages des utilisateurs et générer des réponses. Le bot utilise la bibliothèque python-telegram-bot pour interagir avec Telegram et la bibliothèque langflow pour traiter les entrées des utilisateurs via un fichier JSON Langflow prédéfini.

par Jordan Nonagni

Github: 0d4vid

RAG et Open Source LLM's

Un LLM (large language model), ou modèle de langage large, est un modèle d'intelligence artificielle qui est capable de générer du texte de manière autonome. Il s'agit d'un modèle basé sur l'apprentissage automatisé et l'analyse de grands ensembles de données textuelles pour produire des réponses et des contenus écrits. En ce qui concerne le RAG (Retrieval Augmented Generation), il s'agit d'une méthode qui combine des techniques de recherche et de génération pour améliorer la précision et la pertinence des réponses générées par le modèle de langage. Le RAG utilise une étape de recherche d'informations suivie d'une étape de génération de texte pour fournir des réponses plus précises et adaptées aux besoins des utilisateurs.

Ollama: Lancer des LLM en Local

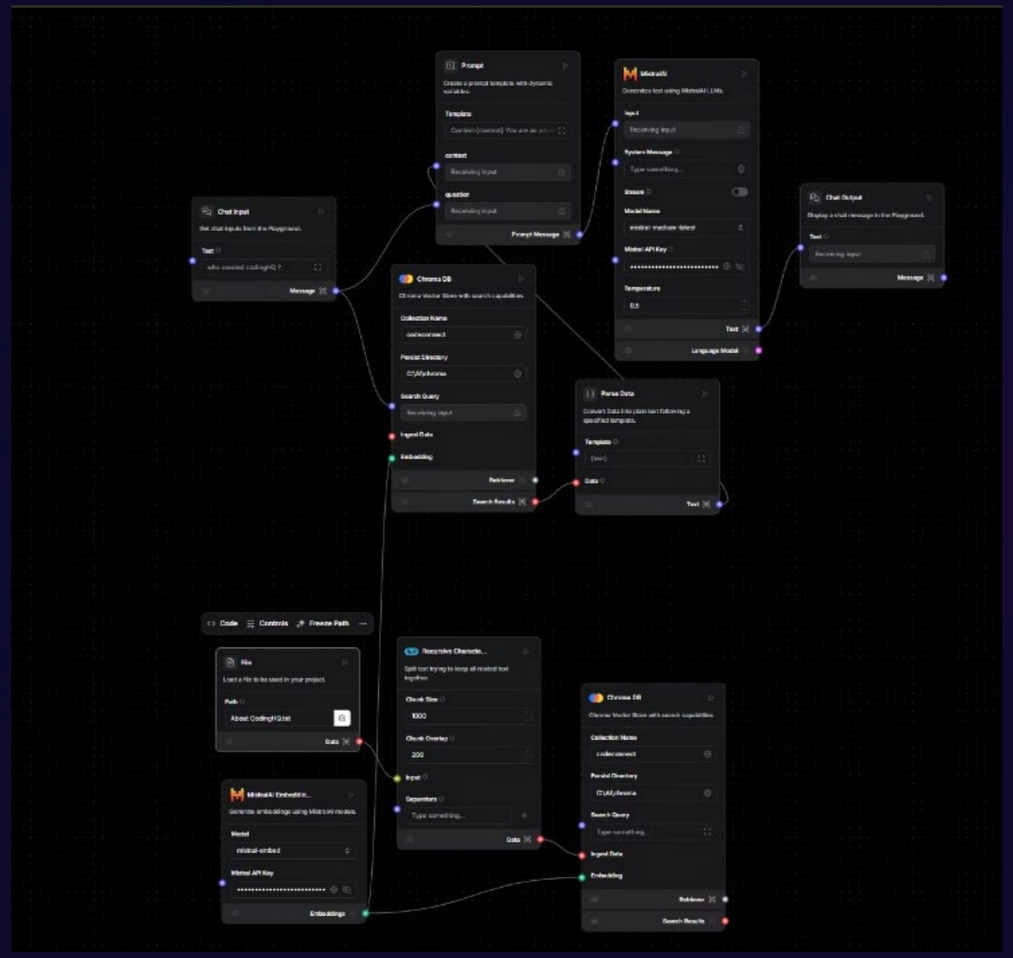
Ollama est un logiciel open source qui permet d'exécuter des modèles de langage volumineux (LLM) en local sur votre propre ordinateur.

Vous n'avez pas besoin d'un serveur coûteux pour exploiter la puissance des modèles de langage volumineux.

Flux Langflow

LangFlow est une bibliothèque opensource qui permet de construire des flux de conversation intelligents pour les chatbots et les assistants virtuels. Il offre des fonctionnalités telles que la gestion des intentions, des entités et des actions pour faciliter le traitement et la génération de réponses cohérentes. Grâce à LangFlow, il est possible de créer des conversations fluides et interactives avec les utilisateurs du bot Telegram.

Conception du flux sur langFlow (en local) Le flux Langflow est conçu en utilisant l'éditeur graphique de Langflow, qui permet de créer un flux conversationnel en y ajoutant des nœuds et en configurant les transitions entre eux. Chaque nœud représente une étape dans la conversation et peut être configuré pour effectuer des actions spécifiques, telles que l'exécution de requêtes API ou l'envoi de messages aux utilisateurs. Une fois que le flux est conçu, il peut être exporté au format JSON pour être intégré dans le Bot Telegram.

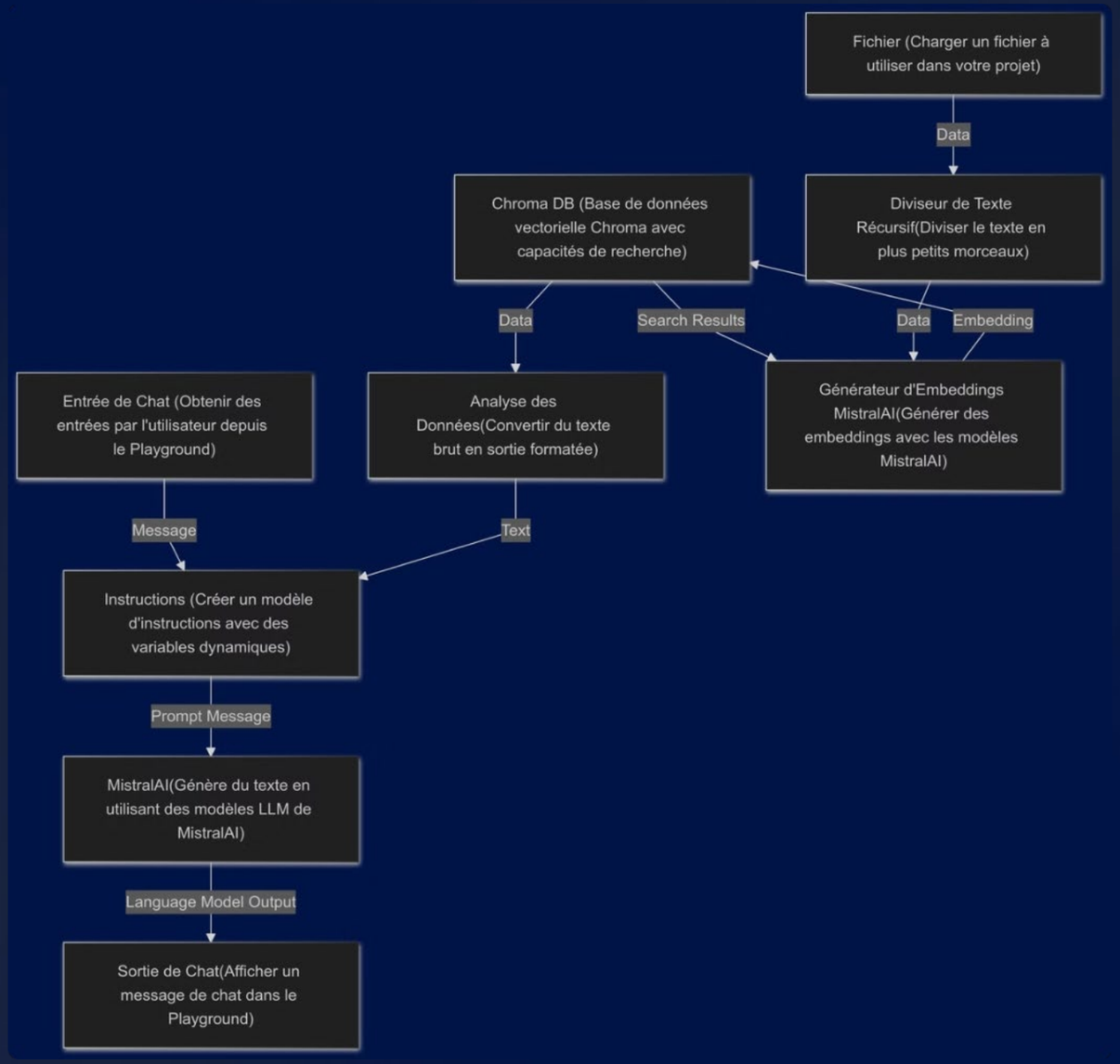


Flux pour CodingHQ-Bot

Ce design de flux sur Langflow est spécifiquement conçu pour le bot CodingHQ-Bot. Chaque nœud représente une fonctionnalité ou une action spécifique que le bot peut effectuer pour les utilisateurs du bot. Ces actions peuvent inclure l'envoi de solutions de code, la recherche de documentation ou même l'envoi de notifications aux utilisateurs. Le flux est optimisé pour une meilleure expérience utilisateur en fournissant des réponses rapides et précises.

diagramme explicatif du flux langflow

Le diagramme explicatif du flux Langflow offre une vision complète du design du flux pour le bot CodingHQ-Bot. Chaque nœud est soigneusement organisé et classé selon sa fonctionnalité. Les utilisateurs peuvent facilement naviguer à travers les différents nœuds pour accéder aux fonctionnalités souhaitées.



Prérequis et Installation

Prérequis

- Python 3.7 ou version ultérieure
- Token du Bot Telegram
- LangFlow
- Fichier JSON Langflow (CodinHQ.json)

Installation

1. Creer un bot telegram sur BotFather
2. Installer les dépendances requises : pip install python-telegram-bot langflow

Configuration du Bot

Token du Bot Telegram

Remplacez TELEGRAM_TOKEN par votre véritable token de bot Telegram.

Fichier JSON Langflow

Assurez-vous que le fichier codinHQ.json se trouve dans le même répertoire que le script ou fournissez le chemin correct vers le fichier.

Tweaks

Définissez les tweaks à utiliser avec le fichier JSON Langflow.

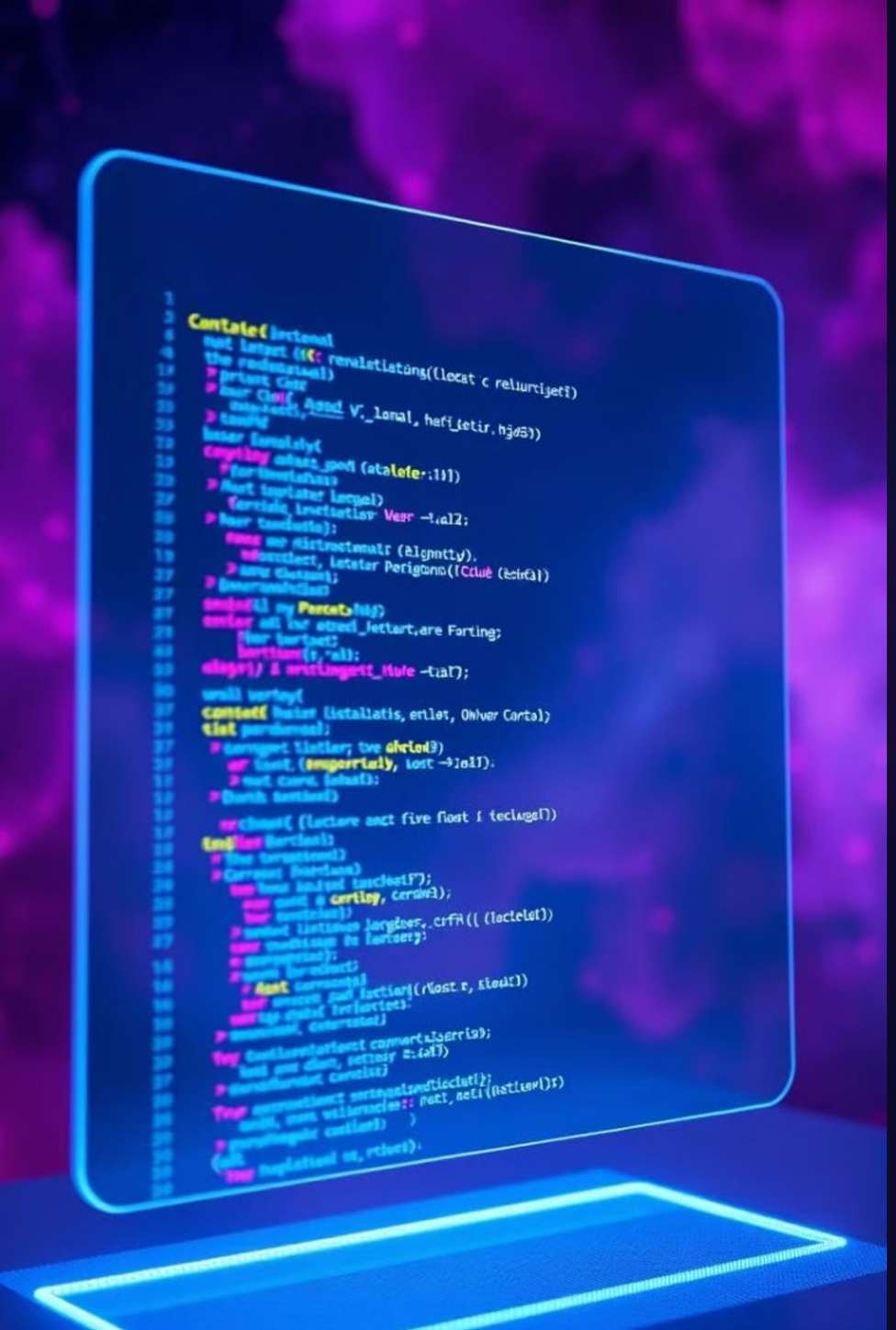
Etapes de creation d'un bot telegram

Pour creer un bot telegram il faut :

1. Se rendre sur le site BotFather (https://core.telegram.org/bots#botfather) et suivre les instructions pour créer un nouveau bot en fournissant un nom et en choisissant un nom d'utilisateur.
2. Récupérer le token du bot créé et le remplacer dans la section "Token du Bot Telegram" du fichier de configuration.
3. Compléter le fichier JSON Langflow en ajoutant les réponses et les actions souhaitées pour les interactions avec les utilisateurs.

Aperçu du Code : Imports

```
from telegram import Update
from telegram.ext import Application, CommandHandler, MessageHandler, ContextTypes, filters
from langflow.load import run_flow_from_json # Utiliser la version synchrone
import asyncio
from concurrent.futures import ThreadPoolExecutor
```



Aperçu du Code : Fonction Langflow Synchrone

```
def sync_get_langflow_response(user_message):
 try:
  result = run_flow_from_json(
   flow=FLOW_FILE,
   input_value=user_message,
   session_id="", # Fournir session_id si nécessaire
   fallback_to_env_vars=True,
   tweaks=TWEAKS
  )
  print("DEBUG: Result from Langflow:", result) # Pour le débogage
  # Extraction de la réponse textuelle
  if isinstance(result, list) and result:
   # Accéder à la structure imbriquée
   message_data = result[0].outputs[0].results['message'].data
   return message_data.get('text', "Je n'ai pas compris votre demande.")
  else:
   return "Je n'ai pas compris votre demande."
 except Exception as e:
  print(f"Erreur lors de l'exécution du flux Langflow : {e}")
  return "Désolé, une erreur est survenue."
```

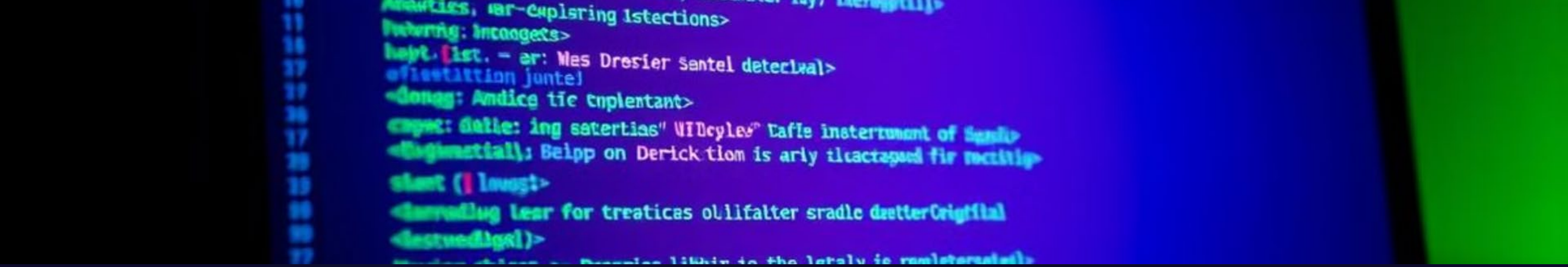
Gestion des Messages et des Commandes

Gestionnaire de Messages

La fonction handle_message traite les messages texte des utilisateurs et utilise la fonction get_langflow_response pour obtenir une réponse générée par Langflow.

Gestionnaire de Commande /Start

La fonction start gère la commande /start et envoie un message de bienvenue à l'utilisateur.



Fonction Principale

```
def main():
 # Créer une instance de l'application Telegram
 print("DEBUG: Starting Telegram bot...") # Pour le débogage
 application = Application.builder().token(TELEGRAM_TOKEN).build()
 # Ajouter des gestionnaires
 application.add_handler(CommandHandler("start", start))
 application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND, handle_message))
 # Démarrer le bot
 print("polling...")
 application.run_polling(poll_interval=3)
```

Conclusion

Dans la conclusion de ce bot, nous pouvons noter qu'il s'agit d'un bot Telegram qui utilise Langflow pour traiter les messages texte des utilisateurs et générer des réponses adaptées. Ce bot comprend une fonction de démarrage qui envoie un message de bienvenue et une fonction principale qui crée une instance de l'application Telegram, ajoute des gestionnaires pour les commandes et les messages texte, puis démarre le bot en utilisant le polling.

Cas d'utilisations

quel sont les autres cas d'application des flux langfCertains autres cas d'application des flux Langflow avec l'intégration de l'IA peuvent inclure la création de chatbots pour le service client, l'automatisation des réponses aux demandes de renseignements fréquentes, la création de chatbots pour le divertissement ou l'assistance à l'apprentissage des langues. Les possibilités sont vastes et dépendent des besoins spécifiques de chaque entreprise ou projet.