

8INF974 – ATELIER PRATIQUE EN INTELLIGENCE ARTIFICIELLE II

Projet #1 – IA dans le monde moderne

Objectifs du projet

- Prendre en main l'environnement de travail à l'UQAC
- Intégrer l'intelligence artificielle dans un vrai logiciel
- Comprendre la relation entre la théorie vue en classe et l'implantation réelle
- Approfondir ses connaissances de manière autodidacte
- Acquérir de l'expertise dans le développement d'intelligences artificielles

Description de votre projet

Choix #1 – ChatBot UQAC

À l'aide d'une librairie moderne (LangGraph, LangChain, Python LangGraph, vous devez faire l'implémentation d'un chatbot en utilisant la technique *Retrieval Augmented Generation* (RAG). Pour des raisons de confidentialités, de protection des données et surtout de coûts, vous devez utiliser **un modèle téléchargé localement** pour réaliser votre projet. Vous pouvez trouver un modèle sur le site internet [GPT4All](https://www.gpt4all.io/) ou sur une autre plateforme similaire. Vous êtes libre de choisir votre modèle (Attention, certains modèles seront trop gros !!!). Vous exploiterez le manuel de gestion de l'UQAC avec l'url suivant : <https://www.uqac.ca/mgestion/>. Le manuel de gestion est très volumineux et comporte plus de 200 liens urls qui mènent à des pages HTML ou des documents PDF uniques. Vous devez implémenter dans votre code une technique de moissonnage (web scrapping) qui va aller chercher tous les liens pertinents et de mettre en mémoire l'information pertinente. Au niveau des pages html, le contenu pertinent à utiliser pour votre RAG sont mis dans des HTML tags <div> ayant comme nom de classe "entry-header" et "entry-content". Pour les fichiers PDF, vous devez télécharger temporairement le fichier localement (librairie tempfile) et faire l'extraction des données. Ensuite, vous devez persister localement les données à l'intérieur d'une base de données. Vous sauverez ainsi beaucoup de temps, car l'importation est coûteuse en temps en raison du volume.

Au niveau du chatbot, vous devez le créer de tel sorte que l'utilisateur demande une question à propos du guide de gestion et le bot fournira la réponse à l'écran. La réponse **doit donner la source** d'où provient l'information qu'il a donné sous la forme d'un lien URL. Par exemple, si la réponse provient de la « *Politique relative à la planification stratégique* » le chatbot doit mentionner : "source : <https://www.uqac.ca/mgestion/chapitre-2/reglement-sur-la-mission-et-les-valeurs-de-luqac/politique-relative-a-la-planification-institutionnelle/>". Enfin, vous devez implémenter un mécanisme de mémoire pour que le bot utilise les questions et les réponses précédentes pour répondre à de nouvelles questions. Pour le frontend de l'application, commencez simplement par utiliser la console ligne de commande de Python. Par la suite, si votre projet avance bien, utilisez un framework Web qui va gérer pour vous le CSS et le Javascript en arrière-plan. Streamlit est une bonne option si vous voulez l'implémenter.

Choix #2 – Système d'exploitation futuriste

Vous devez tenter de moderniser Windows à l'aide de la commande vocale et d'une librairie de *Robotic Process Automation* (RPA). Premièrement, vous devrez faire fonctionner un modèle de fondation pour l'audio (multimodal) à partir d'un PC Windows. Je vous suggère de prendre une version de Whisper locale (Faster-Whisper) ou un modèle similaire. Dans votre logiciel, l'utilisateur devrait simplement avoir à appuyer sur l'espace pour commencer à parler et dicter une commande. Évidemment, comme nous sommes dans un contexte académique, vous devrez penser à un sous-ensemble de commandes restreintes. Par exemple, « Prends une capture d'écran. » ou encore « Ouvre Notepad et écris le texte suivant. ». Votre second défi sera donc de faire correspondre les paroles transcrites à votre ensemble de commandes. Pour y parvenir, vous avez plusieurs options : A – exploitation d'un LLM pour trouver la commande la plus similaire au texte transcript, B-Scinder la commande et utiliser une distance sur les versions encodées des commandes afin de trouver la plus similaire ou C-exploiter des règles, regex combinés à un dictionnaire de synonyme. Ne sous-estimez pas la difficulté de cette étape, si vos commandes ne sont pas atomiques et généralisent (e.g. « Ouvre Excel et entre le calcul suivant : 3+4 ») le système peut devenir très complexe. Je vous suggère par exemple de structurer la commande : {"intent":"type_text","target":"notepad","text":"Bonjour"}. Vous pouvez aussi ajouter un module simple de

désambiguïsation afin de préciser si de l'incertitude survient (e.g. « Veux-tu ouvrir Edge? Tu serais mieux avec Brave »). Vous devriez aussi implémenter un module d'orchestration et de sécurité. En effet, certaines commandes pourraient s'avérer dangereuses et devraient être bloquées (e.g. CMD -r) et d'autres devraient exiger une confirmation si elles sont permises (e.g. effacer un fichier).

Enfin, vous devez exploiter une librairie de RPA. Je suggère de commencer avec pyautogui qui est très simple à faire fonctionner mais assez limité. Exploiter les fonctions clavier et souris ainsi que la capture d'écran. Sinon vous pouvez en choisir une autre telle que pywinauto. Par la suite, tentez d'exploiter un logiciel professionnel comme le RPA de UiPath (<https://www.uipath.com/developers>). Comparez les solutions et réaliser une petite démonstration.

Pour ce projet, on s'attend à l'équivalent de **30 heures de travail par personne** sur 5 semaines (2 et demi en semestre condensé d'été).

Rencontres avec le professeur

Vous devez être préparés pour vos rencontres avec le professeur. Celles-ci devraient se dérouler de cette façon :

1. Récapitulatif d'où vous en étiez précédemment
2. Présentation de vos travaux respectifs
3. Présentation de votre plan et des problèmes auxquels vous faites face
4. Apprentissage fait depuis la dernière rencontre (le cas échéant)

Rapport de projet

Pour vos rapports de projet, vous devriez être en mesure de succinctement :

1. Présenter les points clés de votre implémentation
2. Parler de ce que vous avez réussi et ce qui a échoué
3. Faire un bilan des bugs/défis/apprentissages (e.g., ce que vous feriez différemment)
4. Montrer des exemples et des captures d'écran
5. Montrer des bouts de code si pertinent