Université Paris Nanterre

Licence MIASHS première année

Rapport de projet informatique

Protocole de mise au point d'un modèle IA de prédiction des prix immobiliers

Projet réalisé du 10 mars 2025 au 23 mai 2025

Membres du groupe

Tchock Cédric (44020455)
Anas Boutah (43014239)
Abdelkader Cherfaoui (43014163)

Lien du projet GitHub

https://github.com/offlinereal/prediction-bien-immobilier

Table des matières

1 Introduction

L'estimation du prix d'un bien immobilier repose sur un grand nombre de paramètres : la surface, la localisation, le nombre de pièces, l'année de construction, l'état général du logement, etc.

Si certains de ces critères peuvent être facilement quantifiés, d'autres dépendent d'éléments moins objectifs ou simplement absents de l'information disponible.

Ce projet a pour but de concevoir un outil, écrit en langage C, qui assiste l'utilisateur dans cette tâche en combinant des saisies manuelles avec la puissance d'un modèle d'intelligence artificielle. L'enjeu est double : d'une part, permettre à un programme C de communiquer efficacement avec une API d'IA en générant dynamiquement un prompt pertinent, et d'autre part, explorer concrètement le rôle que peut jouer l'intelligence artificielle dans un processus de prise de décision technique.

Il s'agit ainsi d'un projet à la frontière entre développement algorithmique, conception de programme et utilisation raisonnée d'un outil d'IA générative. Cette interaction, bien qu'assistée, suppose une compréhension fine de la manière dont le programme structure les données avant de les transmettre à l'intelligence artificielle.

L'approche retenue ne prétend pas produire des estimations parfaites, mais vise à proposer un outil souple, capable de s'adapter à des données partielles tout en restant lisible et compréhensible, tant dans sa logique que dans son exécution.

2 Environnement de travail

Le projet a été développé sur un ordinateur personnel fonctionnant sous **Windows 10**. Le langage utilisé est le **C**, avec une compilation manuelle effectuée via le terminal à l'aide du compilateur `gcc`, en liant explicitement la bibliothèque `libcurl` avec l'option `-lcurl`.

Le code a été rédigé à l'aide de l'éditeur **Visual Studio Code**, sans recours à un environnement de développement intégré (IDE) spécifique.

La bibliothèque **libcurl** a été utilisée pour envoyer des requêtes HTTP POST vers l'API d'OpenAI. Une clé d'API personnelle a permis d'interroger le modèle d'intelligence artificielle `gpt-3.5-turbo` hébergé sur `https://api.openai.com`.

Le projet a été versionné et publié sur la plateforme **GitHub**, conformément aux exigences du sujet.

Le rapport a été rédigé en LaTeX en respectant le modèle fourni par le professeur.

3 Description du projet et objectifs

Ce projet a pour but de concevoir un programme capable d'estimer le prix d'un bien immobilier en s'appuyant sur l'intelligence artificielle. Il s'inscrit dans le cadre du *Projet Informatique* de première année. L'enjeu principal n'est pas de créer un modèle de prédiction performant, mais plutôt de suivre un protocole de développement rigoureux, en combinant des compétences en programmation, en utilisation d'API et en interaction avec un modèle de langage.

Le fonctionnement est simple : l'utilisateur saisit les informations qu'il connaît sur le bien (localisation, surface, nombre de pièces, etc.) ou entre un tiret (`-`) lorsqu'il ne dispose pas de la donnée. Le programme construit alors dynamiquement une requête

en langage naturel, qui est envoyée à l'API d'OpenAI. En retour, une estimation de prix est affichée.

L'objectif du projet est donc double : d'un côté, proposer un outil fonctionnel qui exploite les capacités d'un modèle d'IA, et de l'autre, apprendre à structurer un petit projet informatique avec des technologies concrètes.

3.1 Objectifs pédagogiques

Ce projet nous a permis de :
— Développer une application en C autour d'un cas d'usage concret : l'estimation immobilière.
— Utiliser une API externe (OpenAI) avec la bibliothèque `libcurl`, en respectant les formats d'échange attendus (requêtes HTTP, JSON).
— Automatiser la génération d'un prompt en fonction des données fournies par l'utilisateur.
— Structurer un programme robuste, lisible, et accompagné d'une documentation claire et complète.

3.2 Particularités du projet

Plusieurs éléments rendent ce projet intéressant :
— Il fait appel à un modèle d'intelligence artificielle (GPT-3.5), ce qui évite de coder soi-même un algorithme de prédiction.
— Le prompt est généré automatiquement en fonction des données disponibles : l'utilisateur n'a pas besoin de tout connaître.
— Le programme est tolérant aux informations manquantes, ce qui le rend plus accessible à différents profils d'utilisateurs.
— Enfin, le cœur du travail repose sur la mise au point d'un protocole : comment formuler la question à l'IA, comment envoyer la requête, comment récupérer et interpréter la réponse.

4 Bibliothèques, Outils et technologies

Le développement de ce projet a reposé sur un ensemble d'outils et de bibliothèques adaptés aux exigences du langage C et à l'intégration d'une API web.

4.1 Environnement de développement

L'ensemble du projet a été réalisé sous Windows 10, à l'aide de l'éditeur Visual Studio Code (VS Code). Ce choix s'explique par la légèreté et la flexibilité de cet environnement, qui offre des fonctionnalités utiles comme la coloration syntaxique, la complétion automatique et une bonne intégration avec les terminaux de compilation.

4.2 Langage utilisé

Le cœur du programme est écrit en langage C. Ce choix pédagogique permet de travailler avec un langage qui nous est familier, proche du système, tout en intégrant une logique de projet structuré. La manipulation de chaînes de caractères, la gestion

mémoire et l'interaction avec des bibliothèques externes ont constitué des points d'apprentissage importants.

4.3 Communication avec l'API

Pour interagir avec l'API d'OpenAI, on a utilisé la bibliothèque `libcurl`, qui permet d'envoyer des requêtes HTTP depuis un programme en C. Elle a été essentielle pour établir la communication avec l'API en méthode POST, transmettre les en-têtes requis (clé d'authentification, type de contenu) et envoyer le corps de la requête au format JSON.

4.4 Modèle d'intelligence artificielle

Le programme repose sur le modèle GPT-3.5, accessible via l'API d'OpenAI. J'ai utilisé une clé API personnelle, générée à partir d'un de nos comptes OpenAI, pour effectuer les requêtes. GPT-3.5 a été choisi pour sa capacité à comprendre et interpréter des requêtes en langage naturel, ainsi que pour sa rapidité de réponse et sa fiabilité dans les estimations formulées.

4.5 Format JSON

Le corps des requêtes et des réponses s'appuie sur le format JSON. Car le format est léger et structuré, adapté aux échanges entre le programme et l'API. La gestion manuelle du JSON dans un langage comme le C (qui ne dispose pas de support natif pour ce format) a représenté un défi intéressant, nécessitant une attention particulière à la structure des données.

5 Travail réalisé

5.1 Fonctionnalités prévues

Le projet visait à proposer un outil en ligne de commande permettant à l'utilisateur d'estimer le prix d'un bien immobilier à l'aide d'une intelligence artificielle, en l'occurrence le modèle GPT-3.5. Les principales fonctionnalités envisagées étaient les suivantes :

- Lecture des caractéristiques d'un bien immobilier depuis l'entrée standard (système de saisie interactive),
- Gestion des champs optionnels (possibilité de ne pas répondre à toutes les questions),
- Construction dynamique d'un prompt en langage naturel adapté aux informations fournies,
- Envoi de la requête à l'API d'OpenAI via la bibliothèque `libcurl`,
- Affichage de l'estimation générée par l'IA,
- Possibilité de sauvegarder la réponse dans un fichier texte (fonctionnalité optionnelle).

5.2 Fonctionnalités réalisées

L'ensemble des fonctionnalités principales a été implémenté avec succès :

— Le programme récupère correctement les informations saisies par l'utilisateur, y compris la gestion de champs laissés vides (par l'entrée d'un tiret ■ - ■),
— Un prompt cohérent est construit dynamiquement à partir des données entrées,
— La requête est correctement formatée en JSON et envoyée à l'API GPT-3.5 via `libcurl`,
— La réponse de l'IA est affichée en console de façon claire et lisible.

5.3 Fonctionnalités non réalisées

La fonctionnalité de sauvegarde automatique des résultats dans un fichier n'a pas été implémentée. Cela est principalement dû à un recentrage des priorités sur la stabilisation des interactions avec l'API (gestion des erreurs, codage JSON correct, traitement de la réponse) qui s'est avéré plus complexe que prévu en langage C. L'accent a donc été mis sur un fonctionnement robuste du cœur du programme.

L'une des fonctionnalités prévues était l'extraction et l'affichage propre de la réponse de l'IA, sans exposer toute la structure JSON renvoyée par l'API d'OpenAI. Toutefois, cette fonctionnalité n'a pas pu être finalisée dans les délais.

Plus précisément, bien que la requête soit correctement construite et que la réponse de l'API soit bien reçue, le programme affiche encore la totalité du JSON brut, y compris les métadonnées. L'extraction ciblée du champ `content` (contenant la réponse générée par le modèle) s'est avérée plus complexe que prévu, notamment en raison des contraintes liées à la gestion de chaînes de caractères en langage C, ainsi que du manque de bibliothèques natives pour le traitement du JSON.

Des solutions ont été envisagées (utilisation de parseurs JSON tiers, extraction manuelle avec pointeurs), mais n'ont pas pu être pleinement mises en œuvre dans le temps imparti, le développement s'étant déjà révélé exigeant sur d'autres aspects techniques.

Cette fonctionnalité reste donc en suspens, mais pourra être ajoutée dans une version ultérieure du projet avec un peu plus de temps et de recul.

6 Difficultés rencontrées

Le développement de ce projet a soulevé plusieurs défis, principalement d'ordre technique.

L'implémentation des requêtes à l'API d'OpenAI en langage C s'est avérée particulièrement complexe. Le langage C ne disposant pas de bibliothèques natives pour manipuler des formats modernes comme le JSON ou pour interagir simplement avec des services HTTP, il a été nécessaire d'utiliser `libcurl` pour envoyer les requêtes, et de gérer manuellement l'analyse de la réponse. Ce travail a demandé une compréhension fine du fonctionnement des buffers, des pointeurs, ainsi que du formatage des chaînes de caractères.

Un autre point de friction a concerné l'extraction de la réponse utile depuis le JSON renvoyé par l'API. En l'absence d'un parseur JSON dédié en C dans notre projet, il n'a pas été possible de nettoyer proprement la réponse pour n'afficher que le texte pertinent. Cette limitation reste une piste d'amélioration pour les versions futures.

Par ailleurs, une partie du projet a nécessité une exploration préalable de diverses documentations techniques (libcurl, API OpenAI, format JSON, compilation manuelle sous Windows), ce qui a rallongé les temps de développement. L'intégration des diffé-

rents éléments (code, API, environnement) dans un flux cohérent a donc demandé plus de rigueur que prévu initialement.

Malgré ces difficultés, le cœur du projet a pu être mené à bien, avec un programme fonctionnel qui illustre l'objectif fixé.

7 Bilan

7.1 Conclusion

Ce projet a été l'occasion d'explorer un usage concret et innovant de l'intelligence artificielle, en la mettant au service d'un programme développé en C. Il a permis d'appréhender les mécanismes fondamentaux de la communication client-serveur (requêtes HTTP, gestion d'API, formats de données), tout en confrontant la rigueur du C à la souplesse d'une IA générative.

Travailler sur ce projet a également renforcé l'importance de la structuration progressive d'un problème : formuler un besoin, le traduire en prompt, le formater selon les contraintes de l'API, puis restituer la réponse de manière cohérente. Cet aller-retour constant entre logique humaine et logique machine a constitué un véritable apprentissage.

7.2 Perspectives

Des pistes claires d'amélioration se dégagent : la plus immédiate serait l'ajout d'un traitement post-réponse, afin d'extraire automatiquement le contenu pertinent du format JSON renvoyé par l'API. Une autre perspective serait d'introduire des options interactives pour permettre à l'utilisateur de personnaliser davantage sa requête.

Enfin, ce prototype pourrait servir de socle à des applications plus spécialisées : estimation locative, diagnostics automatiques, simulateurs d'achat, etc. Ce type d'architecture démontre comment un langage système comme le C peut dialoguer avec des modèles d'IA de haut niveau, dans des contextes concrets et extensibles.

8 Webographie

Références

[CAT] `savoircoder.fr/cat`

[LC] *libcurl - The multiprotocol file transfer library*, `https://curl.se/libcurl/c/`, documentation officielle de la bibliothèque `libcurl`, utilisée pour envoyer des requêtes HTTP.

[JSMN] *JSMN : Minimalistic JSON parser in C*, `https://zserge.com/jsmn.html`, documentation d'un parseur JSON, utile pour comprendre les bases de la lecture de structures JSON.

[OAI] *OpenAI API Documentation*, `https://platform.openai.com/docs/api-reference`, documentation officielle de l'API OpenAI utilisée dans ce projet pour interroger un modèle GPT.

[GPT] *ChatGPT – OpenAI*, `https://chat.openai.com`, utilisé tout au long du développement du projet, notamment pour comprendre des notions techniques, améliorer et aider dans la conception de certaines parties du code, clarifier le fonctionnement des API.

[SE] *Stack Overflow*, `https://stackoverflow.com`, plateforme communautaire utilisée ponctuellement pour résoudre des problèmes de compilation, d'allocation mémoire ou d'implémentation de requêtes HTTP.

9 Annexes

Annexe A : Cahier des charges

Le cahier des charges de notre projet *Protocole de mise au point d'un modèle IA de prédiction des prix immobiliers* était le suivant :

1. **Objectif du projet :**

   Développer un programme en C capable d'interagir avec une intelligence artificielle via une API (ici, celle d'OpenAI), afin d'estimer le prix d'un bien immobilier à partir d'un certain nombre d'informations fournies par l'utilisateur.

2. **Fonctionnalités clés :**
   - Permettre à l'utilisateur de saisir les caractéristiques d'un bien immobilier (localisation, surface, nombre de pièces, état, etc.), avec la possibilité de laisser certains champs vides.
   - Générer dynamiquement un prompt en langage naturel adapté aux données saisies.
   - Interroger le modèle GPT-3.5 via l'API d'OpenAI en envoyant le prompt construit.
   - Afficher la réponse de l'IA de manière lisible à l'écran.
   - Gérer les éventuelles erreurs de connexion ou de réponse de l'API.

3. **Interface utilisateur :**

   Le programme repose sur une interface en ligne de commande simple, interactive et claire, permettant une saisie guidée des informations par l'utilisateur.

4. **Compatibilité :**

   L'application doit fonctionner sous les systèmes Unix/Linux et Windows 10, avec un environnement de développement courant (Visual Studio Code, terminal, bibliothèque libcurl).

5. **Performance :**

   Le programme doit être léger, réactif et ne pas dépasser un temps d'exécution de quelques secondes pour une requête API typique.

6. **Sécurité et confidentialité :**

   L'implémentation doit utiliser une clé API personnelle, saisie ou stockée de façon sécurisée, et ne pas exposer celle-ci dans le code source partagé (notamment sur GitHub).

7. **Organisation et rendu :**
   - Le projet doit être déposé sur GitHub.
   - Un rapport technique rédigé en LaTeX doit accompagner le projet.
   - Des captures d'écran de démonstration doivent être incluses pour valider le fonctionnement.

Annexe B : Exemple d'exécution du projet

Nous joignons ci-dessous deux captures d'écran issues de nos tests sur le terminal (cf. Figure 1 et Figure 2) :

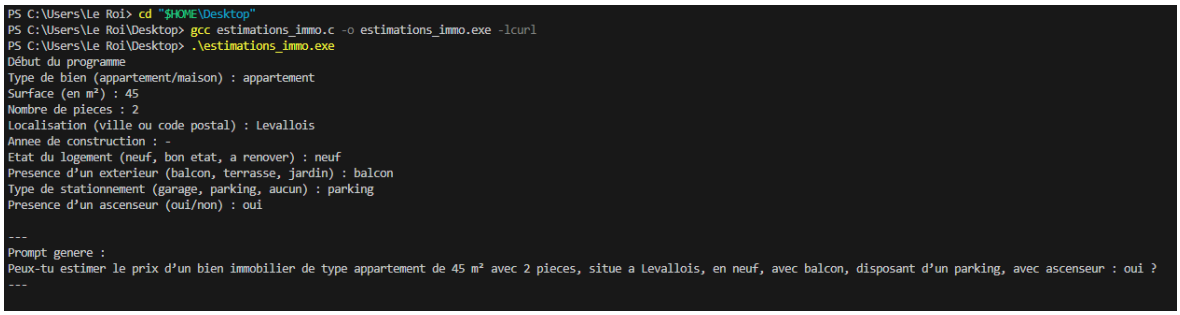

FIGURE 1 – Exécution d'un cas de test à Levallois

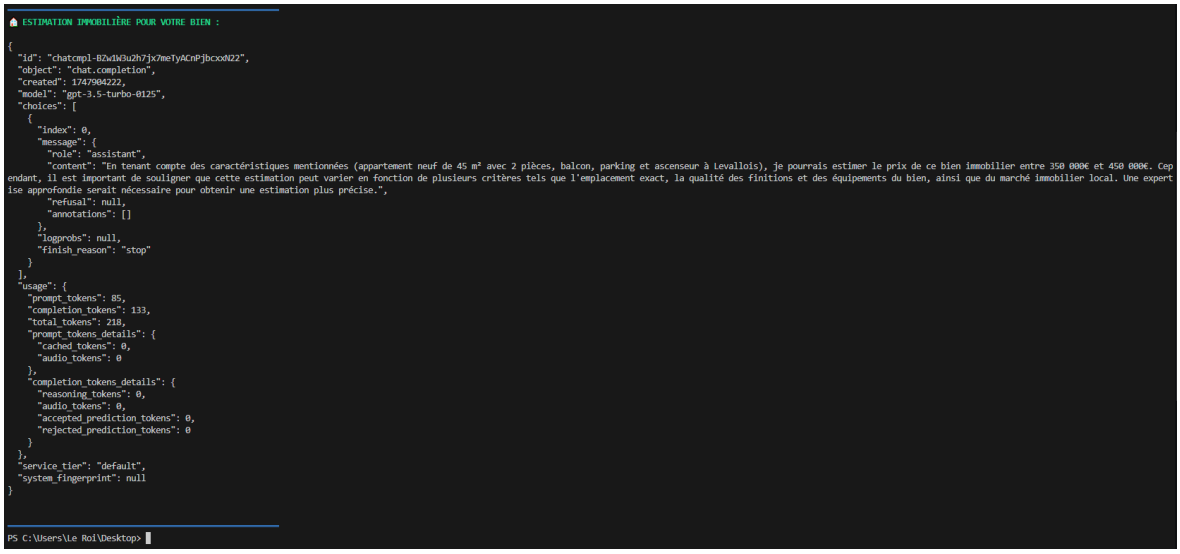

FIGURE 2 – Réponse de l'IA pour un bien situé à Levallois

Annexe C : Manuel utilisateur

Ce manuel utilisateur fournit des instructions détaillées sur la façon d'utiliser notre programme d'estimation de prix immobilier.

Démarrage du programme :

Pour commencer, compilez et lancez l'application depuis votre terminal. Le programme vous affichera alors une série de questions pour saisir les caractéristiques de votre bien immobilier.

Saisie des caractéristiques :

Pour chaque caractéristique demandée (type de bien, surface, nombre de pièces, localisation, année de construction, état, extérieur, stationnement, ascenseur), tapez la réponse correspondant à votre bien puis validez avec la touche `Entrée`.

Si vous ne connaissez pas une information ou préférez ne pas la renseigner, tapez simplement le caractère `"-"` (moins) puis validez. Cela n'empêchera pas le programme de générer une estimation.

Génération du prompt :

Après la saisie de toutes les données, le programme construit automatiquement une requête détaillée décrivant votre bien immobilier à destination de l'API d'intelligence artificielle.

Cette requête sera affichée à l'écran pour que vous puissiez la vérifier.

Obtention de l'estimation :

Le programme envoie la requête à l'API OpenAI et récupère une estimation de prix sous forme textuelle.

Cette estimation s'affiche ensuite dans le terminal, encadrée pour une meilleure lisibilité.

Fin du programme :

Une fois l'estimation affichée, le programme se termine automatiquement.

Pour refaire une estimation :

Il suffit de relancer le programme et de recommencer la saisie.

Remerciments :

Nous espérons que ce programme vous aidera à obtenir rapidement une estimation fiable de votre bien immobilier. Bonne utilisation !