Méthodologies d’extraction de données sur du texte utilisant les « modèles de langues » (LLM)

L’intérêt des LLM pour mon projet d’étude est qu’ils permettent de compléter une tâche sans entraîner de modèles. (1)

L’extraction de données utilisant les LLM se déroule en 4 étapes :

* Créer une demande
* Donner le texte non structuré au LLM
* Récupérer la sortie structurée du LLM
* La sauvegarder dans une base de données

On peut demander une sortie structurée au modèle (comme couleur = noir ou blanc) avec un fichier .yaml. La structure demandée sera chargée par des classes « Pydantic BaseModel ».

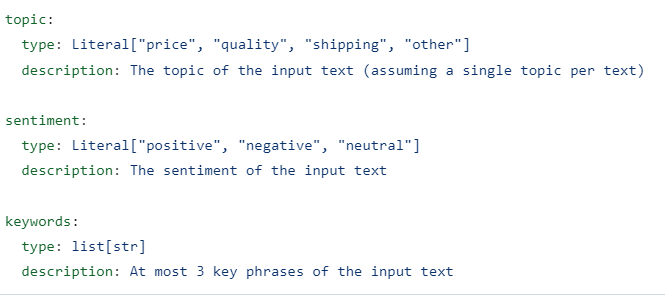


Figure 1: Exemple de structure de fichier yaml

On indique au modèle les indications de recherche dans description, et les sorties possibles (si on en a) dans keywords.

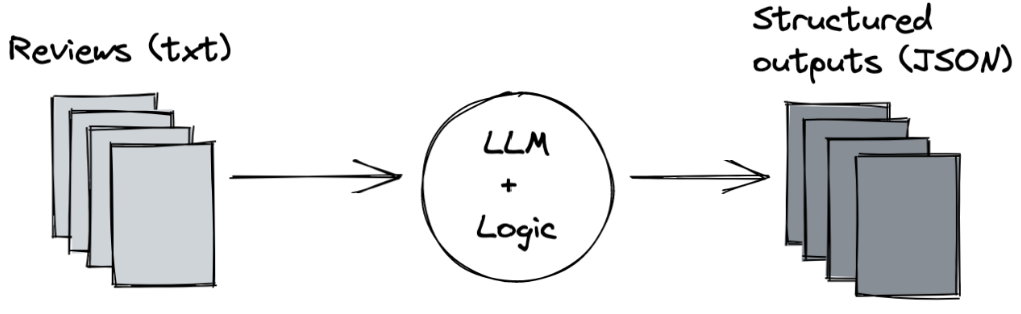
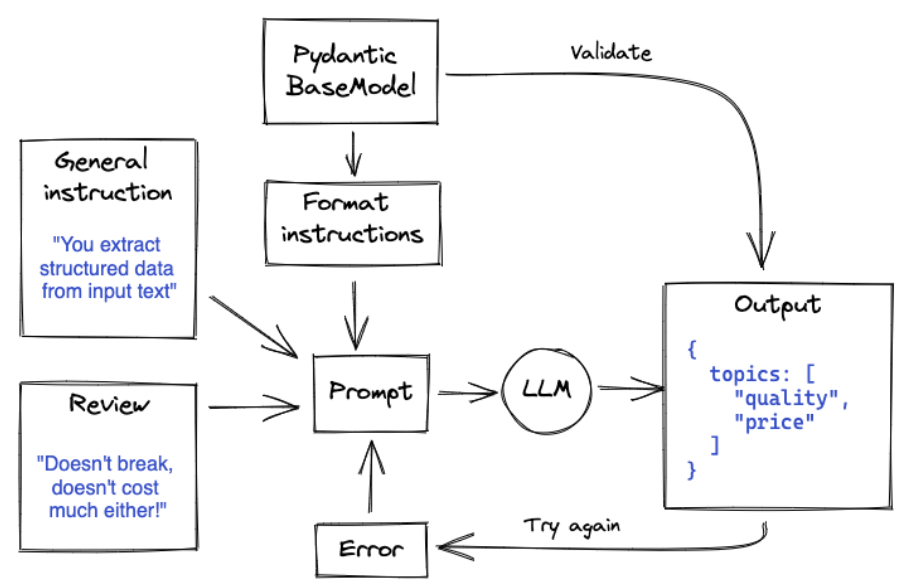
 

Figure 2: Séparation de la logique modèle d’avec les données

Figure 3: Schéma général de la demande au modèle

La dernière étape consiste à segmenter la logique modèle pour le rendre facilement utilisable.

Tout ceci vient d’un article réalisé par Xebia, une entreprise de consultation en transformation digitale.

Nous allons voir la documentation de « Langchain » sur ce sujet, c’est une librairie open source de développement d’applications autour des LLM.

Il y a deux types d’approches pour ce problème (2) :

* **Fonctions**: Certains LLM ont des fonction intégrées d’extraction arbitraires d’entités (peuvent donner des attributs que l’on n’a pas demandé).
* **Parsing**: Classes de structuration de sorties des LLM (ex Pydantic).

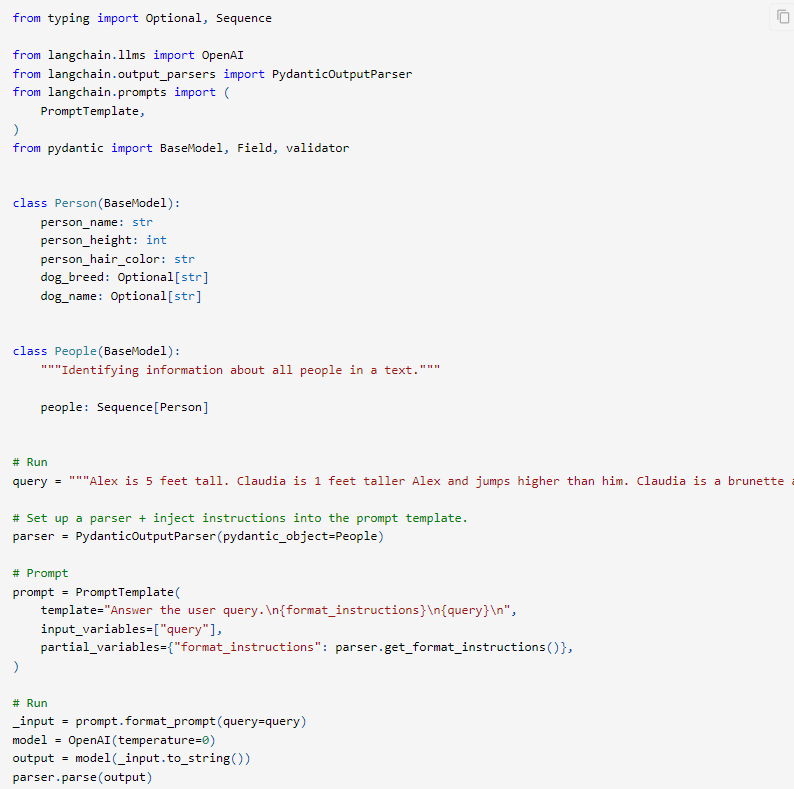
Pour le parsing:

Figure 4: Demande de parsing sur un prompt

Sur la figure 4, on remarque :

* Utilisation la classe BaseModel pour définir les attentes. Création d’un objet parser.
* Dans le modèle de prompt, définition d’un prompt général contenant toutes les informations, décomposables dans les différentes entrées (input\_data, format\_instructions, template).
* Run d’un modèle avec le prompt (devra être local et gratuit dans mon cas), segmentation de la sortie avec le parser.

Sur « LangSmith Trace », on voit le résultat, ainsi que ce qui a été donné au modèle :



Figure : Suivi sur LangSmith Trace

Bibliographie

1. **https://xebia.com/blog/archetype-llm-batch-use-case/.** [En ligne]

2. **https://python.langchain.com/docs/use\_cases/extraction.** [En ligne]