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

# Préambule

Qu’est-ce qu’un grand modèle de langage (LLM) ?

Les grands modèles de langage (LLM) sont des modèles d’apprentissage automatique capables de comprendre et de générer des textes en langage humain. Ils fonctionnent en analysant des ensembles de données linguistiques massives. (1)

# Table des matières

[Préambule 1](#_Toc158208621)

[Table des matières 1](#_Toc158208622)

[1. Présentation des documents 2](#_Toc158208623)

[2. Analyse des documents 2](#_Toc158208624)

[2.1 Xebia – How to extract structured data from unstructured text using LLMs 2](#_Toc158208625)

[2.2 Langchain – Use cases – Extraction 4](#_Toc158208626)

[Bibliographie 7](#_Toc158208627)

# 1. Présentation des documents

Les documents ayant servi au recueil d’informations proviennent du site Xebia (article de septembre 2023) et du site Langchain (date inconnue mais copyrigt 2024 en bas du site).

Xebia est une entreprise de consultation en transformation digitale, il s’agit de professionnels du domaine, leurs informations peuvent s’avérer utiles et donner une vue du sujet traité.

Langchain est une librairie connue, open source, de développement d’applications autour des LLM. La documentation à travers cette librairie est indispensable pour savoir comment traiter le sujet en pratique.

# 2. Analyse des documents

Dans cette partie nous verrons les informations pertinentes contenues dans les deux pages.

## 2.1 Xebia – How to extract structured data from unstructured text using LLMs

L’intérêt des LLM est qu’ils permettent de compléter une tâche sans entraîner de modèles.

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 ».

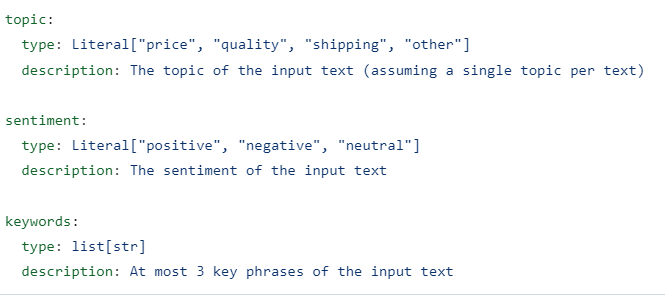


Figure 1: Exemple de structure de fichier yaml

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

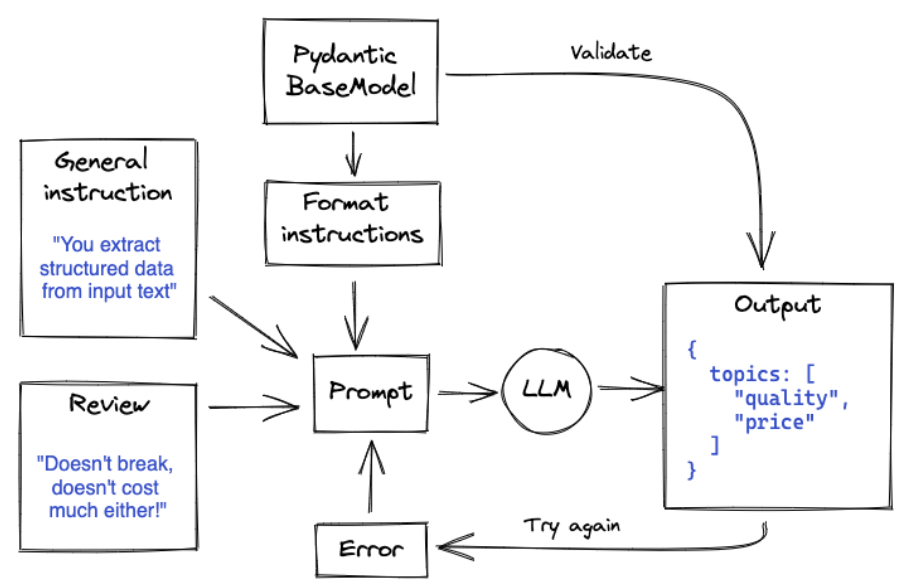


Figure 3: Schéma général de la demande au modèle

On voit sur la figure 3, que l’on forme un prompt, à l’aide de :

* Instructions de formatage
* Instructions générales
* Information d’entrée

On fournit ce prompt au modèle. Cela nous donne une sortie, qui est validée par le modèle pydantic.

Si on a une erreur, on peut tenter de s’adapter à cette erreur en itérant (automatiquement en donnant l’erreur ?) ou en adaptant le prompt.

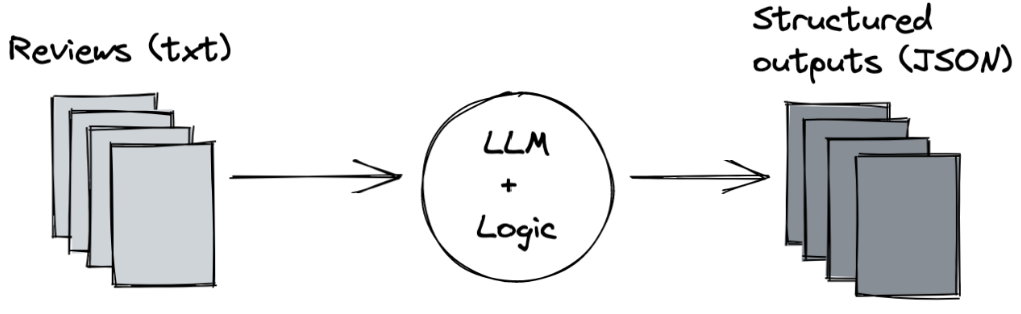


Figure 2: Séparation de la logique modèle d’avec les données

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

## 2.2 Langchain – Use cases – Extraction

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

* **Fonctions** : Certains LLM ont des fonctions 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).

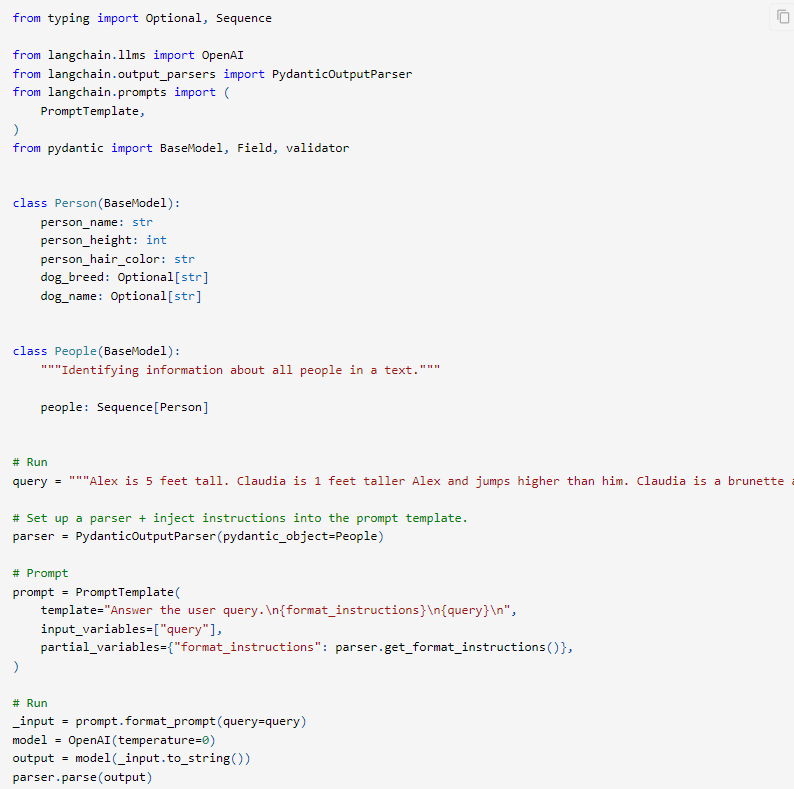
C’est le parsing qui nous intéresse, voici un exemple de parsing :

Figure 4: Demande de parsing sur un prompt

Sur la figure 4, on remarque :

* Utilisation la classe BaseModel (pydantic) dans une imbrication de classes pour définir les attentes de sorties. Création d’un objet « parser » contenant ces informations.
* Dans le modèle de prompt (langchain), définition d’un prompt général contenant toutes les informations, les informations sont également décompées par types dans les entrées (input\_data, format\_instructions, template).
* Run d’un modèle avec le prompt, segmentation de la sortie avec le parser.

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



Figure 5: Suivi sur LangSmith Trace

Il y a beaucoup d’exemples d'instruction, c’est le principe de l’ingénierie de prompt : on adapte le prompt pour s’adapter aux erreurs du modèle. Ce qui est dans le dictionnaire est assez dur à interpréter, d’autant plus que ce n’est pas affiché de manière lisible.

On remarque que l’exemple est plus concret que l'article de Xebia, et permet une mise en application pratique, il est complémentaire avec l'article de Xebia qui donne d'autres informations.

# Bibliographie

1. **Cloudflare.** [En ligne] https://www.cloudflare.com/fr-fr/learning/ai/what-is-large-language-model/.

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

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