ML Engineer Assignment

Task description

In file '3_furniture stores pages.csv' we have a listof URLs from furniture store web sites.
Our goal is to extract product names from the given web pages.

Challenges and solutions

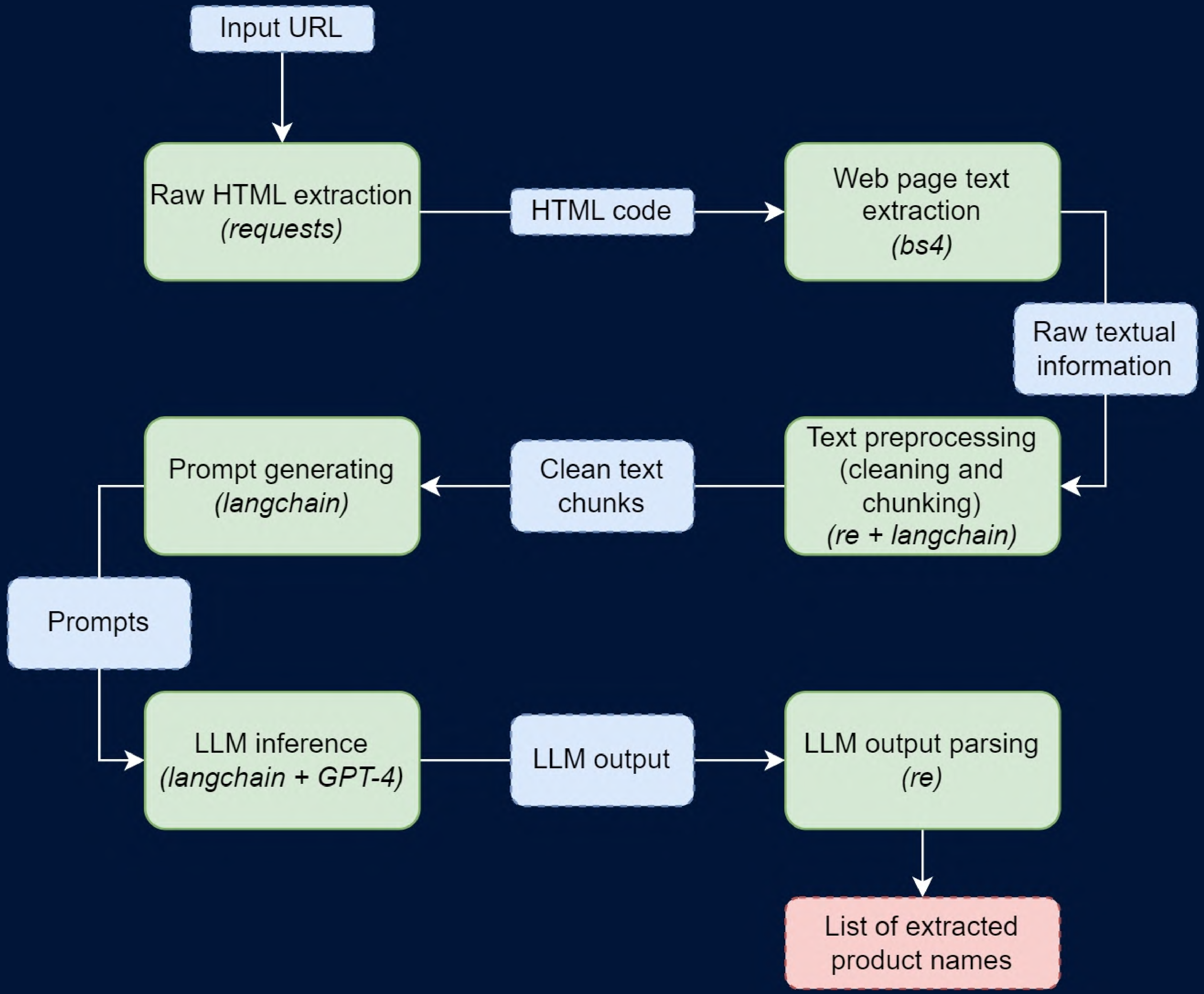
- Challenge: All web pages have different structure, so we cannot jparse them to extract entities of interest.
  Solution: Instead we can use NLP techniques to get that information.

- Challenge: Some approaches (like custom NER) requires manual data labeling, which can take a lot of time, as well as existing models fine-tuning.
  Solution: Find a pretrained model that doesn't require fine-tuning for this task.

- Challenge: There are almost no suitable pretrained models.
  Solution: No problem! Just use LLM for entity extraction.

Our approach

So, we can use LLM (GPT-4 in our case) and prompt engineering to recognise and extract product names from web pages. However, it requires some extra steps to achieve the actual result.


Input URL
Raw HTML extraction (requests)
HTML code
Web page text extraction (bs4)
Raw textual information
Text preprocessing (cleaning and chunking) (re + langchain)
Clean text chunks
Prompt generating (langchain)
Prompts
LLM inference (langchain + GPT-4)
LLM output
LLM output parsing (re)
List of extracted product names

1 Raw HTML extraction

To get raw HTML content of the web page we simply make a GET request to the specified URL using Python requests module.

2 Web page text extraction

We use BeautifulSoup4 library to extract all available textual data from the web page.

3 Text preprocessing

Using the magic of regular expression, we remove all unnecessary characters (e.g. whitespaces and linebreaks) from the raw text. In order not to exceed the input token limit, we split the texts into several overlapping chunks using CharacterTextSplitter from Langchain framework.

4 Prompt generating

Using Langchain prompt templates we generate a prompt for each chunk of text. According to the prompt, model should return a list of product names or “None” if no product is mentioned in text. Interface for GPT-4 is also provided by Langchain framework.

5 LLM inference and output parsing

Again, with the magic of regular expression we parse the output of GPT-4 model and extract entities of interest if any were found. The list of extracted product names is printed to console.

Results and evaluation

To evaluate the performance of our pipeline we need labeled data, so from the first 48 URLs we've obtained 19 working web pages, that contains products names, and then we've manually extracted entities of interest.

The main goal in this step is to compare lists of products extracted by LLM with our results, determine the number of true positive, false positive and false negative items and compute precision, recall and F1 scores.

Results and evaluation

Automatically computed metrics:

Precision	0.676
Recall	0.742
F1-score	0.708

We can see that results are not quite perfect and the reason for this is that our data labeling is subjective. So instead we have to compute these scores comparing extracted and actual results manually.

Precision	0.911
Recall	0.995
F1-score	0.951

Ok! This looks much better!

Known issues

- As we've already mentioned, it's hard to estimate the actual quality of the pipeline, since it requires data labeling that can be subjective.
- Sometimes LLM extracts entities that are not visible on the web page, so with actual product names we can get some junk (this problem explains not so high precision score).
- This app cannot be used "out-of-the-box" because it requires access to GPT-4 and Azure OpenAI Service. However, experiments with more accessible LLMs can fix this issue.

Conclusions

- More approaches to the given problem could be applied with labeled data, however, in our case entity extraction with LLM was the only choice.

- The current LLM approach is still not perfect, so experiments with other LLMs and prompt engineering could be useful.