MEASUREMENTS PARSER

BASF Measurements Parser Challange

Problem Formulation

Input Data

- Given an XML file contains many patents, we follow these steps:

1. Split the patents.
2. We filter patents using <section> element to only include Chemistry patents.
3. We extract the test from the filtered patents, we use the text in <**description**>, <**abstract**>, <**claims**>, then put all text a txt file for later usage.
4. We further clean the text by removing unwanted characters.

Models

- GPT3.5-Turpo : Main model to parse a given text using a pre-engineered prompt and output format instruction.
- Embedding Model (OpenAI): Used it to embed documents and select relevant documents via Similarity measures and LangChain.

Prompts

- Few-Shots prompting: In few-shots experiments, we provide some high-quality examples for GPT, each contains contains both inputs and desired outputs for the desired task. These high-quality demonstration will show GPT the true intentions about the inut/output. Remember, we are only extracting one piece of information to solve one problem.
- Prompts Directory contains a sub directory for each prompt with the prompt id as the subdirectory name. each subdirectory contains mainly 2 files:

prompt.json: This file contains prompt instructions and output format, ex:

```
"prompt": "You are given a text from chemistry research paper, extract all the measurements mentioned in the text. Only Extract Measurement,
Unit and Value ranges.\nOnly extract measurements that have value and units.\nIf you find no measurements, return [].",
"output_format": "Your output should follow be a list of json objects, good output format : [ { 'measurement ' : 'string ', 'unit ' : 'string
', 'value ' : 'string ' }, { 'measurement ' : 'string ', 'unit ' : 'string ', 'value ' : 'string ' }", "id": "002"
}
```

examples.json: *Contains examples used with the prompts (in case few-shots prompting). The examples should be in a list of json objects, ex:*

```
[{ "text": "Text", "measurements": [ {"measurement":"ceria crystallite size", "unit": "nm", "value": "10 to 20"}, {"measurement":"alkaline earth oxides
crystallite size", "unit": "nm", "value": "20 to 40"}, {"measurement":"composite particles BET surface area", "unit": "m/g", "value": "30 to 80"}] }, {"text":
"Text", "measurements": [ {"measurement":"refractory metal oxide support surface area", "unit": "m^2/g", "value": "100 to 200"}, {"measurement":"coated
Substrate support", "unit": "g/in", "value": "1.5 to 7.0"}, {"measurement":"Alumina BET surface area", "unit": "m/g", "value": "200"} ] }, {"text": "The method
of claim 1, wherein the nitrogen oxide storage material has a surface area between about 30 and 80 m^2/g.", "measurements": [ {"measurement": "nitrogen oxide
surface area", "unit":"m^2/g", "value": "30 and 80"} ]}]
```

Prompt Evaluation

- For the Prompt Evaluation, we follow two paradigms:
- *Context Evaluation:* we use the examples in the prompts as inputs, and check whether GPT outputs the expected output or no. TO a prompt to be considered, the prompt need to score 100% accuracy for this evaluation.
- *Test set Evaluation:* After Context Evaluation, how can we make sure that the prompt is not overfitting over the examples? We carefully choose a test set (with inputs and ground truth) from the provided patents manually, then use it as evaluation set for all prompts, same test set is used for all prompts so we can compare.
- For this Version, I use a Custom prompt builder, I could use Langchain, but usually I use custom classes with first versions to fully understand the problem then later version I use Framework like Langchain.

	total_overall_accuracy	total_true_predictions	total_miss_predictions	total_false_predictions	number_of_tokens	prompt_id
0	0.96875	18	0	1	101	000
1	0.96875	18	0	1	103	001
2	0.96875	18	0	1	111	002
3	0.90625	17	0	2	112	003

Prompt Versioning

- It's important to version each prompt, for each prompt sub-directory, there will be logs, evaluation results, examples used, instructions used and output format used.

Parser

- I use Langchain to read text, split to Chunks then send it chunk by chunk to LLM.
- For the output format instruction, I use Custom outputformat attached with the prompt Instructions in order to version it.
- Output for each Chunk is then attached to the output list to be processed by the Output formatter.

Output Format

- Output format is challenging, We can craft our prompts to be explicit about the output format, but there's no guarantee that the outputs will always follow this format. That's why I preferred using custom formatter.

- For each prediction, the prediction is first parsed using json loader, if failed due to bad output format, it's added to bad json list for further processing.

```
"output_format": "Your output should follow be a list of json objects, good output format : [ { 'measurement ' : 'string ', 'unit ' : 'string ', 'value ' : 'string ' }, { 'measurement ' : 'string ', 'unit ' : 'string ', 'value ' : 'string '
```

Parsing Results

Experiment: 1

- We use 5 Patents in this experiment.
- Patents text is extracted, splitted to chunks with chunk size 6000 then sent chunk by chunk to LLM and predictions are collected.

Results

- Parse could predict 118 json Measurements from 5 patents with 54 Chunks.
- 40 Predictions had false JSON format, some of them were irrelevant (results can be viewd in the Output folder).
- After reviewing the predictions manually, I found that most of the prediction were irrelevant to the problem.

	measurement	unit	value
0	amplicon pacbio sequencing coverage	%	90
1	molecular lop replicate strain typings concord...	%	10
2	paralel testing betwen molecular lop and tradi...	%	98.2
3	analytical sensitivity/specificity heat-inacti...	%	96.23
4	analytical sensitivity/specificity heat-inacti...	%	9.97
...	...	...	...
113	pipe strength	mpa	267 to 282
114	pipe defect	None	None
115	yield strength after 96-h imersion	mpa	367 to 418
116	tensile strength in 50 c., 70%	mpa	8 to 34
117	sulfuric acid grouping	None	None

118 rows × 3 columns

Parsing Results

Experiment: 2

- We use same settings as Experiment 1 and same data, but this time chunks are first retrieved with relevance.
- We use OpenAI Embedding to Embed the documents then Langchain FAISS as a Retriever to retrieve the documents relevant to a certain prompt.
- We use `"Experiments with Measurements, Values and units of measurements“ as Relevance measure.`
- Then the retrieve documents are Passed to the LLM

Results:

- Out of 54 Chunks, the relevant chunks are 4.
- Number of Predicted Measurements are far less.
- No Wrong Format.
- All the predicted Measurements are relevant to the problem.

	measurement	unit	value
0	sample surface area	m^2/g	5 and about 350
1	coated Substrate support	g/in	1.5 to 7.0
2	Alumina BET surface area	m/g	200
3	nitrogen oxide surface area	m^2/g	30 and 80
4	polypeptide concentration	g/ml	1 to 9
5	surface area	m^2/g	13.5 or more, preferably 14.0 or more, more pr...
6	yield strength	MPa	230 or more, preferably 250 or more
7	tensile strength	MPa	380 or more, preferably 40 or more

Future Improvements

Data:

- Domain experts to select Test examples and to select the demonstrations.
- Use domain knowledge to further clean the data and filter the text.
- Use different technique (list similarity measures) to select various demonstration that reflects the problem at hand.

Prompts:

- Do more experimentations with different prompts (one-shot, zero-shot).
- Start using chain of thought for the evaluation and faulty prediction correction.

LLM Communication:

- I would add the support for paraller processing and multi threading for faster parsing and to overcome the rate limit problem.

Output format

- We can craft our prompts to be explicit about the output format, but there's no guarantee that the outputs will always follow this format. This is challenging,, my current approach is ignoring prediction that canot be loaded as json.
- I would use Chaining and Langchain agents to fix faulty prediction, asking GPT to correct wrong JSON string.