## What We Have Explored and the Findings

1. **Serverless API and Inference Endpoints:**

**Method**: Direct interaction with models via text prompts provided through Hugging Face's hosted APIs.

**Results**: The outputs were unstructured and incomplete, frequently truncated due to reaching the max\_new\_tokens limit. These outputs lacked the necessary organization and completeness to meet JSON schema requirements.

1. **Local Model Implementation with Package Jsonformer**

**Models Tested**: Meta-llama-3 families, GPT-2, Gemma-2b, and Dolly-V2-12b.

**Results**: There was an improvement in output completeness but not in content relevance. The downside is that significant computational resources are required to download the model locally, and if a restart is necessary, it must be re-downloaded, which wastes time and computational resources.

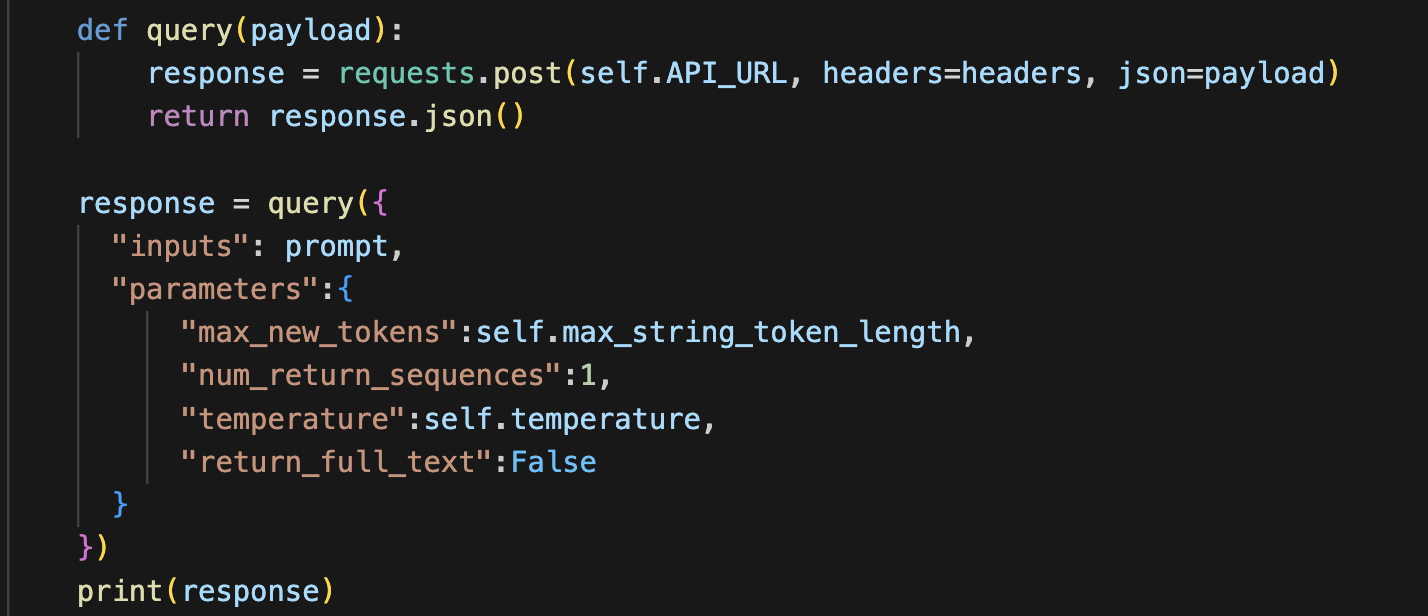
1. **Benchmarking with GPT-3.5:**

**Purpose**: for comparison (using authorized data)

**Results**: Similarly, GPT-3.5 struggled with key recognition and adherence to the JSON schema.

## Current Usage and Plan:

Currently, we are using **meta-llama/Meta-Llama-3-8B-Instruct**. Since we have identified shortcomings in the local implementation as mentioned above, we decided to integrate both approaches. To be more specific, we continued using the inference endpoint and serverless API (the model's URL can be used for future deployment) and combined it with a customized **jsonformer** class. This will allow us to call remote server interfaces while also obtaining structured data in the required JSON format.

**Core Code:**

**Mechanism**

The Jsonformer is designed to interact with hosted machine learning models to generate structured JSON outputs. It works by **extending prompts** to guide the model, then **processing the model's output** into a structured JSON format. The mechanism includes the following key operations:

* Prompt Extension: The base prompt is adjusted to guide the model toward generating output that can be structured into JSON format. This involves **specifying the desired JSON schema** in the prompt to align the model's output with our data structuring needs.
* Output Processing: After receiving the raw string output from the API, the Jsonformer parses this string and iteratively fills a schema-based structure to **convert it into structured JSON**.

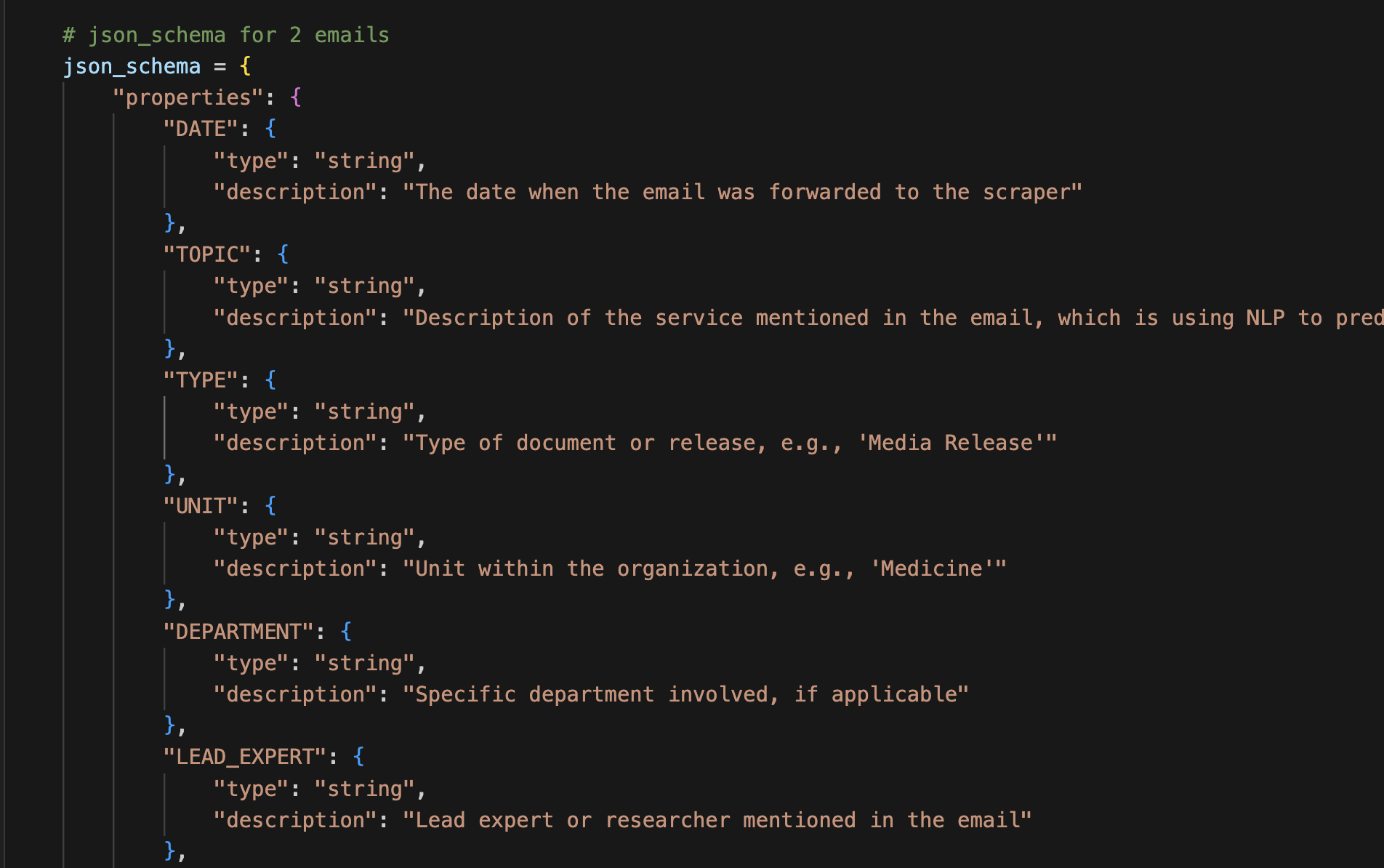
**Class Implementation**

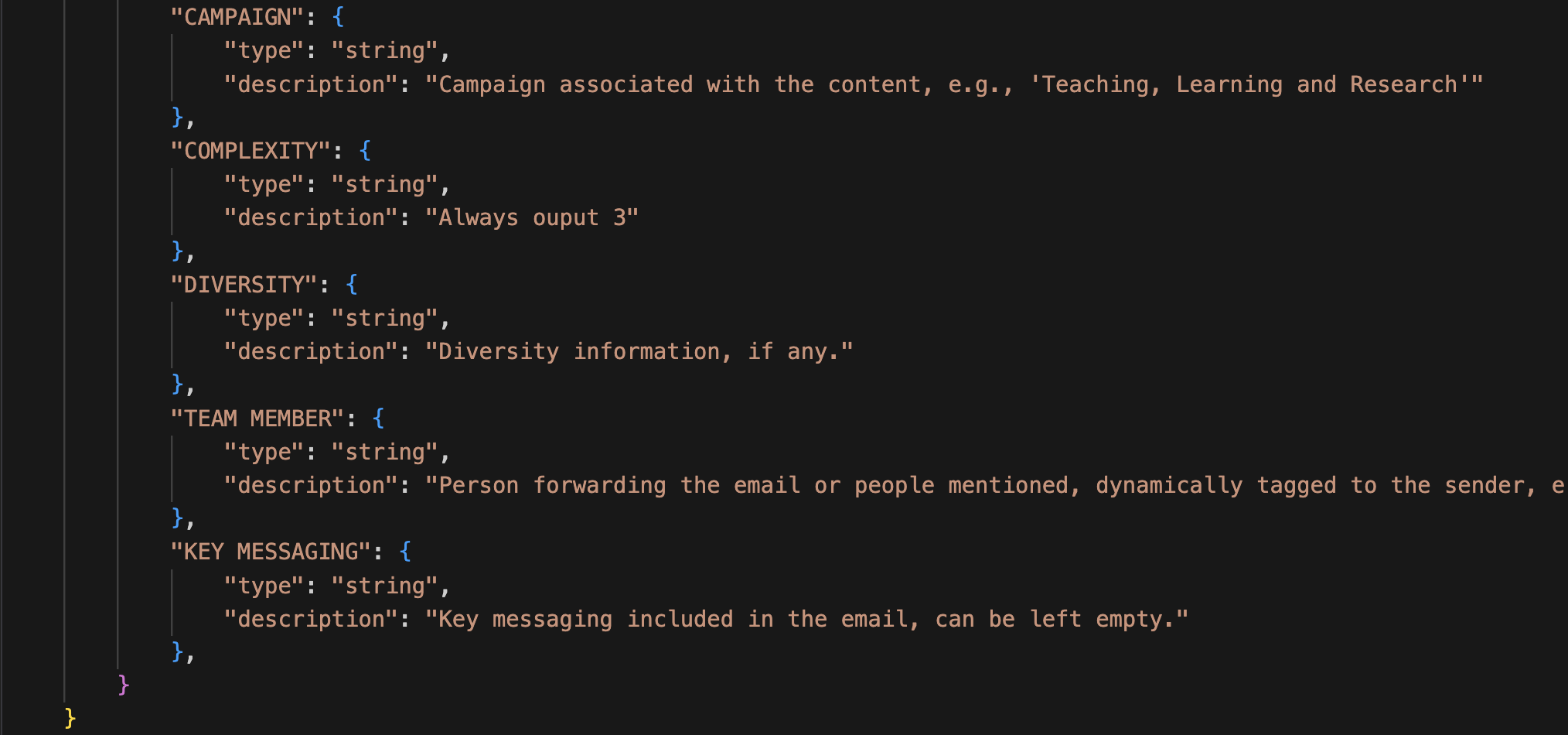
An overview of the Jsonformer class:

* **Initialization**: Parameters such as the API URL, tokenizer, JSON schema, and various settings related to prompt management and generation are set up. Debugging options are available to trace the generation steps.
* **Generate String**: constructs a prompt and makes an HTTP request to the specified API endpoint to retrieve a string based on the JSON schema requirement. It handles tokenization and adjusts parameters such as temperature to control the randomness of output generation.
* **Generate Object**: Iteratively populates an object according to the provided schema, calling the appropriate method to generate strings or nested objects as required.
* **Generate Value**: Determines the type of value to generate based on the schema, whether it's a string, object, or other types, and calls the relevant method to handle the generation.
* **Operational Flow**: The prompt is dynamically built using the current state of the JSON being generated, ensuring that each part of the schema is adhered to progressively. The generation marker is used to identify where in the JSON output the next generated value should be inserted.

**Operational Workflow:**

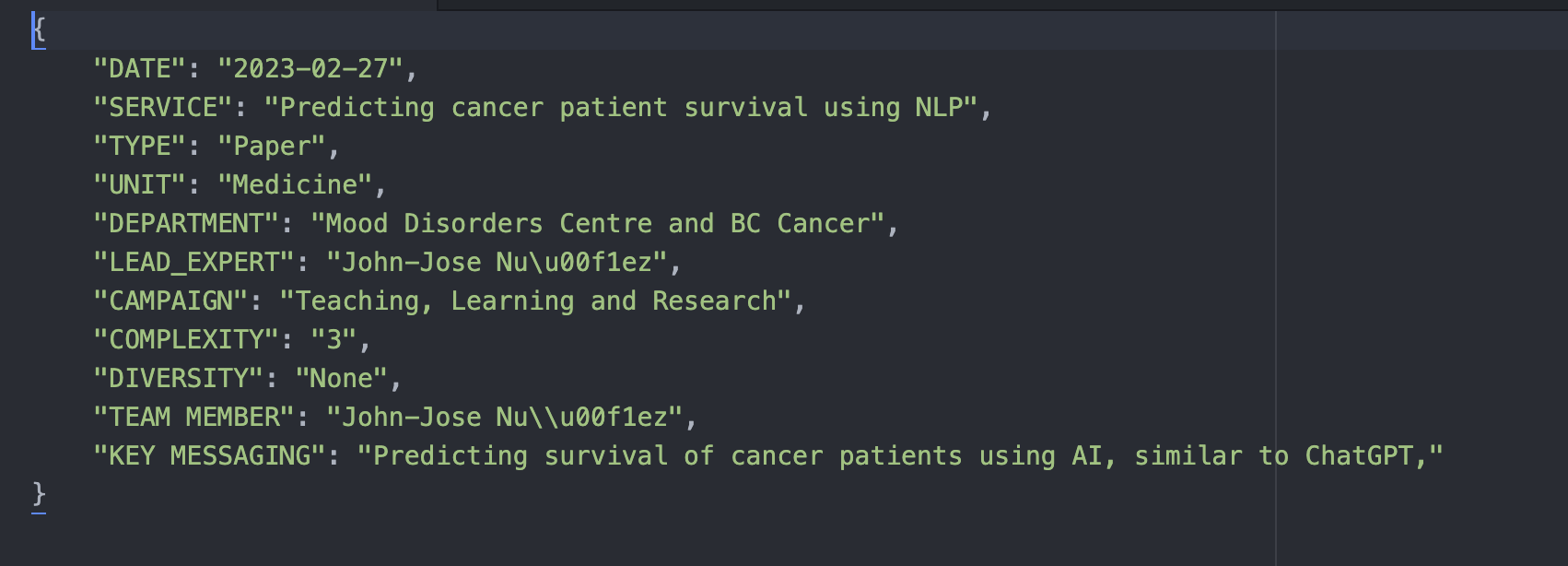
1. Provide the unstructured email file path to obtain the respective email content.
2. Specify the desired JSON schema (below is an example).



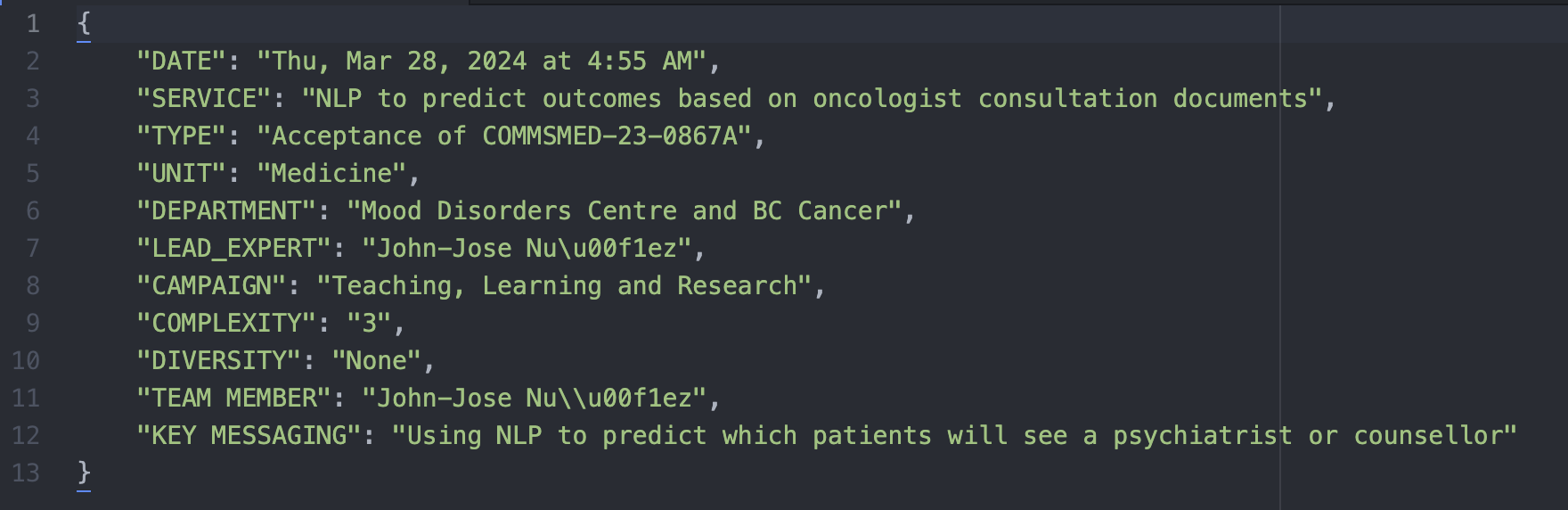


1. Provide the model ID and deployment URL when calling the jsonformer class, which will initiate requests to the remote model multiple times based on the number of fields desired. Each prompt will be automatically adjusted accordingly.
2. The final output will be data in JSON format. Below are 4 outputs of the example files.

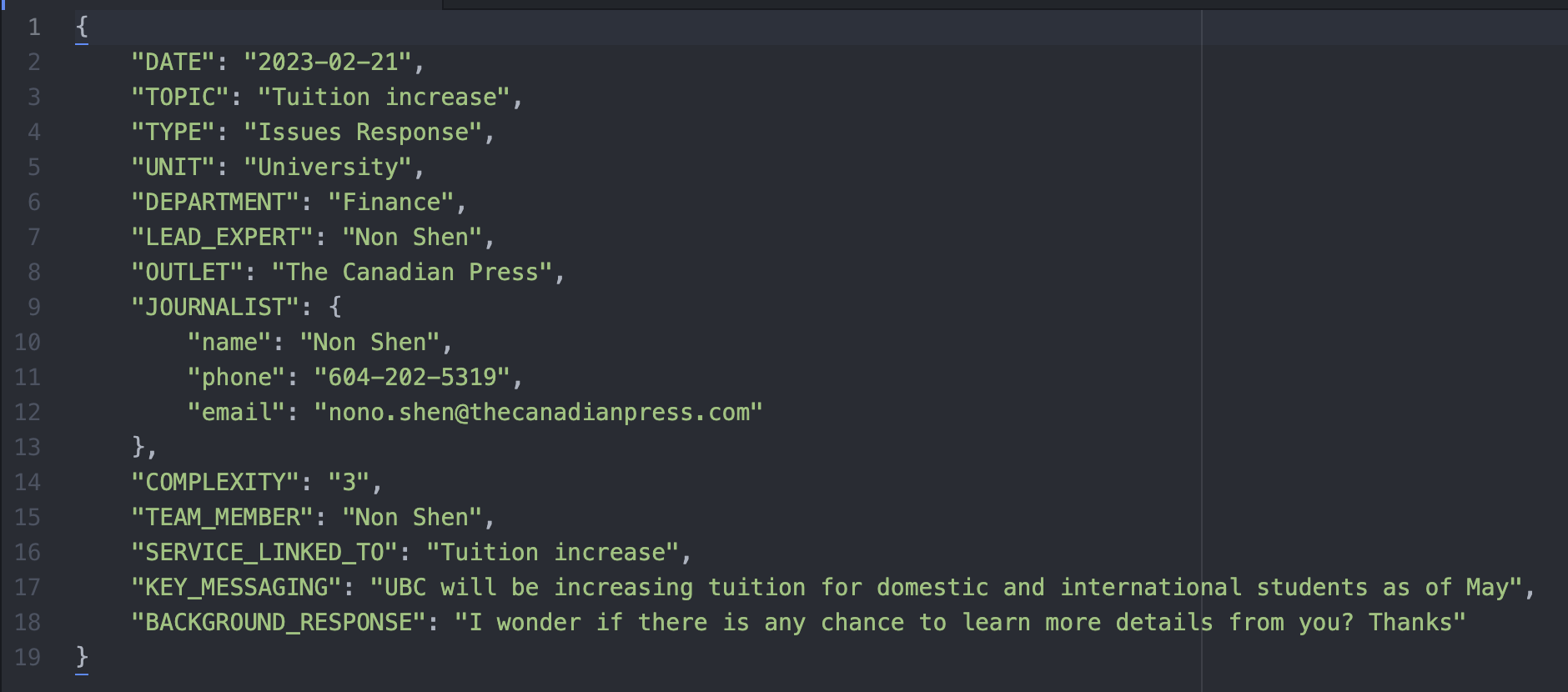
4.1 jj\_email\_1.eml:



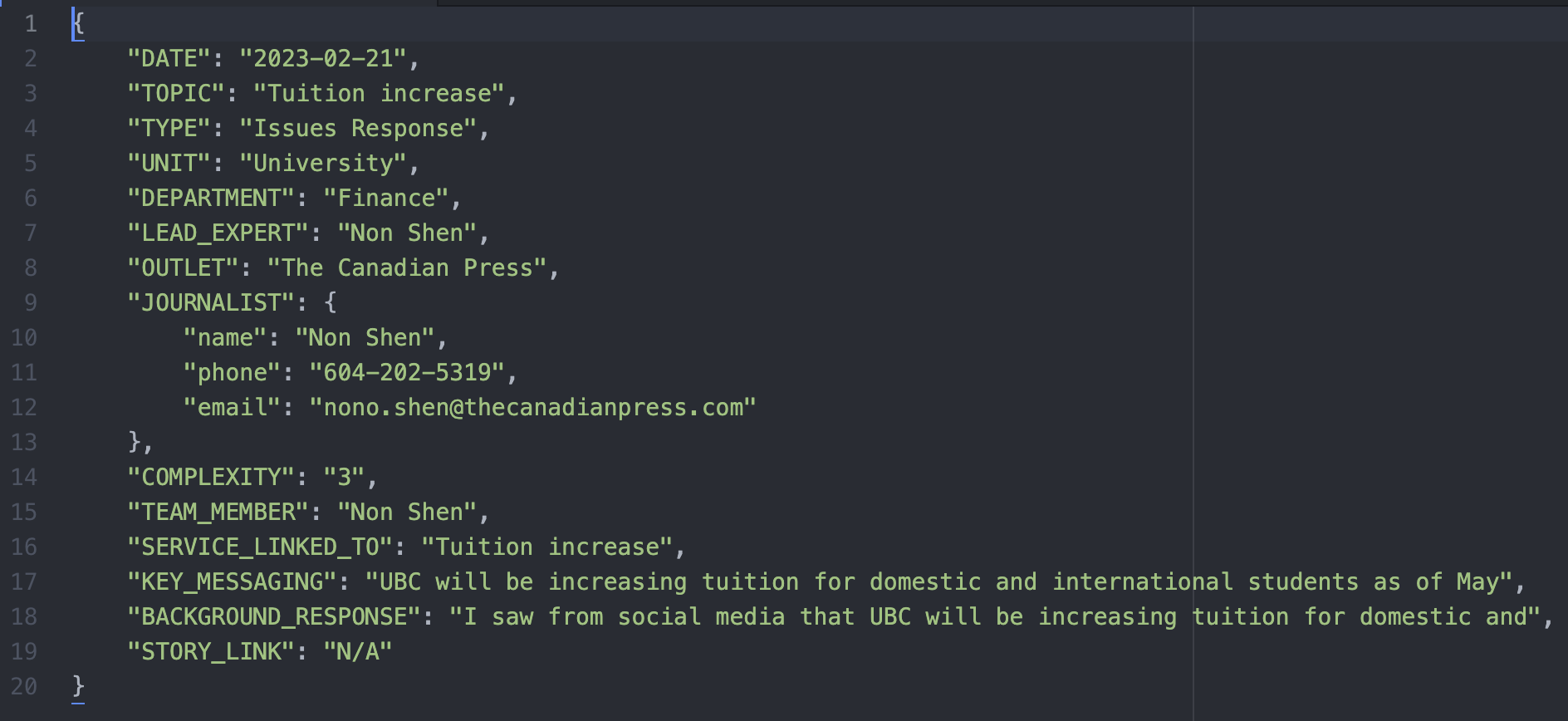
4.2 jj\_email\_2.eml:



4.3 form\_fill\_1.txt



4.4 other\_email\_1



**Model Parameters:**

* Size and Architecture: Meta-Llama-3-8B-Instruct is a large language model with **3.8 billion parameters**. The model architecture is based on the **transformer** framework.
* Training Data: The model was trained on a diverse dataset comprising several hundred thousand data points, including text from books, articles, websites, and other digital media.

**Future Improvements:**

Our goals for improvement include:

1. Finding a more effective model (as we only need to deploy it to get the URL and tokenizer, allowing us to seamlessly replace the model currently used in our code).
2. Adjusting prompts to better align with our requirements.
3. Currently, the JSON schema accepts only string and object fields. We plan to expand this to include boolean, int, float, and more.