Generative structured data extraction using LLMs

Mara Wilhelmi, Sherjeel Shabih, Martino Rıos Garcıa, Santiago Miret,Christoph Koc

4/11/24

Table of contents

1 Generative structured data extraction using LLMs

Structured data is at the heart of machine learning. LLMs offer a convenient way to generate structured data based on unstructured inputs. This book gives hands-on examples of the different steps in the extraction workflow using LLMs.

2 Constrained generation to guarantee syntactic correctness

> **Note**
>
> If we want to generate output that is structured in a specific way, we can use various techniques to
>
> - make the extraction more efficient (but automatically adding the "obvious" tokens)
> - make the generation guaranteed to be syntactically correct
> - make the generation sometimes more semantically correct, too

To enable constrained decoding, we will use one of the most popular packages for this task `instructor`. It is built on `pydantic` and can leverage function calling and JSON-mode of the OpenAI API as well as other constrained sampling approaches.

```
from pydantic import BaseModel, Field
from typing import List, Optional, Literal
import erdantic as erd
import instructor
from IPython.display import SVG
from openai import OpenAI
from dotenv import load_dotenv
load_dotenv('../.env', override=True)
```

```
True
```

2.1 Defining a data schema

For most constrained generation tasks, we need to define a data schema in a programmatic way. The most common way to do so is to use `pydantic` data classes. Here is an example of a simple data schema for a recipe:

```python
from pydantic import BaseModel

class Recipe(BaseModel):
    title: str
    ingredients: List[str]
    instructions: List[str]
```

This schema can also be extended to include descriptions of different fields or to only allow certain values for specific fields. For example, we could add a field for the number of servings and only allow positive integers.

```python
from pydantic import BaseModel, Field
from typing import Literal, List

class Recipe(BaseModel):
    title: str
    ingredients: List[str]
    instructions: List[str]
    servings: int = Field(..., gt=0, description="The number of servings for this recipe")
    rating: Literal["easy", "medium", "hard"] = Field("easy", description="The difficulty
```

If we want to extract copolymerization reactions a data schema could look like the following.

We can now use `instructor` to "patch" the OpenAI API client to ensure that our output fulfils the schema.

```python
client = instructor.patch(OpenAI(), mode=instructor.Mode.MD_JSON)


class Monomer(BaseModel):
    name: str = Field(..., title="Name", description="Name of the monomer.")
    reactivity_constant: Optional[float] = Field(
        None,
        title="Reactivity constant",
        description="Reactivity constant of the monomer. r1 for monomer 1 and r2 for monom
        ge=0,
    )
    reactivity_constant_error: Optional[float] = Field(
        None,
        title="Reactivity constant error",
        description="Error in the reactivity constant. Often indicated with +/-. Must be g
        ge=0,
```

```
    )
    q_parameter: Optional[float] = Field(
        None,
        title="Q parameter",
        description="Q parameter of the monomer. Q1 for monomer 1 and Q2 for monomer 2. Mu
        ge=0,
    )
    e_parameter: Optional[float] = Field(
        None,
        title="e parameter",
        description="e parameter of the monomer. e1 for monomer 1 and e2 for monomer 2.",
    )


class CopolymerizationReaction(BaseModel):
    temperature: Optional[float] = Field(
        ...,
        title="Temperature",
        description="Temperature at which the reaction is carried out",
    )
    temperature_unit: Optional[Literal["C", "K"]] = Field(
        ..., title="Temperature unit", description="Unit of temperature"
    )
    solvent: Optional[str] = Field(
        None,
        title="Solvent",
        description="Solvent used in the reaction. If bulk polymerization was performed, t
    )
    initiator: Optional[str] = Field(
        None, title="Initiator", description="Initiator used in the reaction"
    )
    monomers: Optional[List[Monomer]] = Field(
        ...,
        title="Monomers",
        description="Monomers used in the reaction. Ensure that the reactivity ratios are
        min_items=2,
        max_items=2,
    )
    polymerization_type: Optional[str] = Field(
        ...,
        title="Polymerization type",
```