***link to these slides and all libraries/references at the end

Specification	Implication
Accurate, stable grounding	*Return OMOP-standardised terms without hallucination or unpredictable model output*
Respect OMOP-style constraints	*Generalisable methods to enforce CDM conventions (stronger than just enum lists)* *Specify task-dependent preferences for term sets and concept hierarchies*
Zero-shot, configurable pipeline	*New targets and vocabularies specified declaratively without retraining*
Reusable & portable	*Support sharing of validated configurations - no re-development required*
Operate on standard professional grade machines	*Must produce workable results with models that can run without dedicated GPU i.e. typically no larger than ~3-7B parameters* High-level *abstraction to allow more powerful or bespoke models where resourcing and throughput/reasoning demands allow*
Run under heavily restricted environments	*Support linking to locally hosted models, vocabularies & configuration resources* *Integrate with securable model hosts*

***link to these slides and all libraries/references at the end

Specification	Implication
Accurate, stable grounding	*Return OMOP-standardised terms without hallucination or unpredictable model output*
Respect OMOP-style constraints	*Generalisable methods to enforce CDM conventions (stronger than just enum lists)* *Specify task-dependent preferences for term sets and concept hierarchies*
Zero-shot, configurable pipeline	*New targets and vocabularies specified declaratively without retraining*
Reusable & portable	*Support sharing of validated configurations - no re-development required*
Operate on standard professional grade machines	*Must produce workable results with models that can run without dedicated GPU i.e. typically no larger than ~3-7B parameters* High-level *abstraction to allow more powerful or bespoke models where resourcing and throughput/reasoning demands allow*
Run under heavily restricted environments	*Support linking to locally hosted models, vocabularies & configuration resources* *Integrate with securable model hosts*

Semantically Grounded

***link to these slides and all libraries/references at the end

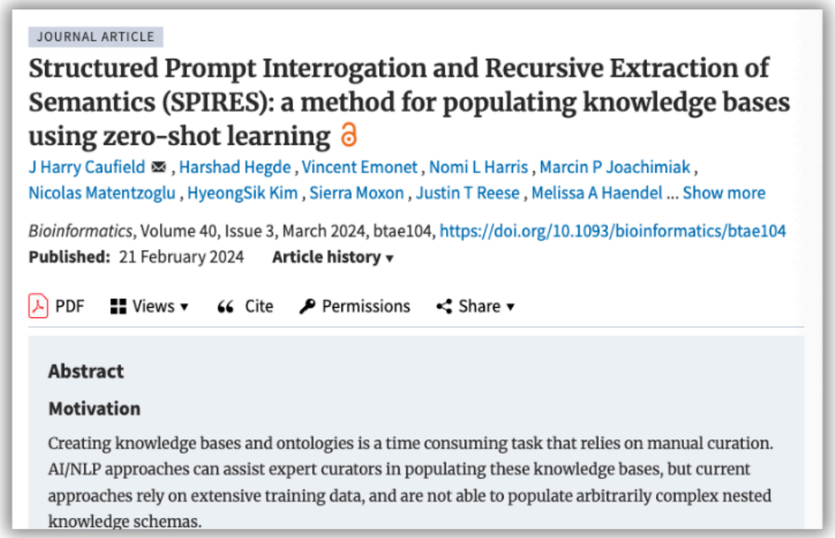
Specification	Implication
Accurate, stable grounding	*Return OMOP-standardised terms without hallucination or unpredictable model output*
Respect OMOP-style constraints	*Generalisable methods to enforce CDM conventions (stronger than just enum lists)* *Specify task-dependent preferences for term sets and concept hierarchies*
Zero-shot, configurable pipeline	*New targets and vocabularies specified declaratively without retraining*
Reusable & portable	*Support sharing of validated configurations - no re-development required*
Operate on standard professional grade machines	*Must produce workable results with models that can run without dedicated GPU i.e. typically no larger than ~3-7B parameters* High-level *abstraction to allow more powerful or bespoke models where resourcing and throughput/reasoning demands allow*
Run under heavily restricted environments	*Support linking to locally hosted models, vocabularies & configuration resources* *Integrate with securable model hosts*

Semantically Grounded ⟷ **Configurable & Sharable**

***link to these slides and all libraries/references at the end

Specification	Implication
Accurate, stable grounding	*Return OMOP-standardised terms without hallucination or unpredictable model output*
Respect OMOP-style constraints	*Generalisable methods to enforce CDM conventions (stronger than just enum lists)* *Specify task-dependent preferences for term sets and concept hierarchies*
Zero-shot, configurable pipeline	*New targets and vocabularies specified declaratively without retraining*
Reusable & portable	*Support sharing of validated configurations - no re-development required*
Operate on standard professional grade machines	*Must produce workable results with models that can run without dedicated GPU i.e. typically no larger than ~3-7B parameters* High-level *abstraction to allow more powerful or bespoke models where resourcing and throughput/reasoning demands allow*
Run under heavily restricted environments	*Support linking to locally hosted models, vocabularies & configuration resources* *Integrate with securable model hosts*

Semantically Grounded ⟷ **Configurable & Sharable** ⟷ **Hostable & Secure**



JOURNAL ARTICLE

Structured Prompt Interrogation and Recursive Extraction of Semantics (SPIRES): a method for populating knowledge bases using zero-shot learning

J Harry Caufield , Harshad Hegde , Vincent Emonet , Nomi L Harris , Marcin P Joachimiak , Nicolas Matentzoglu , HyeongSik Kim , Sierra Moxon , Justin T Reese , Melissa A Haendel ... Show more

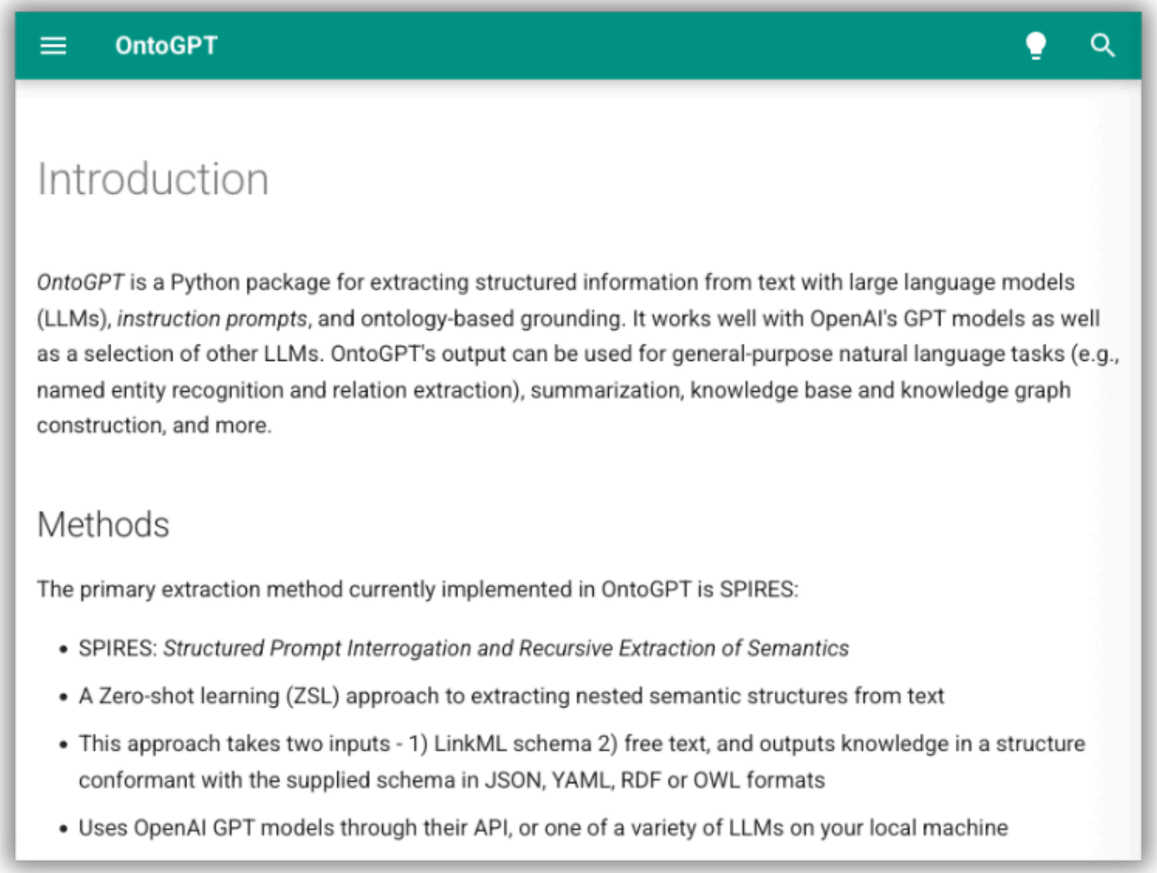
Bioinformatics, Volume 40, Issue 3, March 2024, btae104, https://doi.org/10.1093/bioinformatics/btae104
Published: 21 February 2024 **Article history**

PDF Views Cite Permissions Share

Abstract

Motivation

Creating knowledge bases and ontologies is a time consuming task that relies on manual curation. AI/NLP approaches can assist expert curators in populating these knowledge bases, but current approaches rely on extensive training data, and are not able to populate arbitrarily complex nested knowledge schemas.



OntoGPT

Introduction

OntoGPT is a Python package for extracting structured information from text with large language models (LLMs), *instruction prompts*, and ontology-based grounding. It works well with OpenAI's GPT models as well as a selection of other LLMs. OntoGPT's output can be used for general-purpose natural language tasks (e.g., named entity recognition and relation extraction), summarization, knowledge base and knowledge graph construction, and more.

Methods

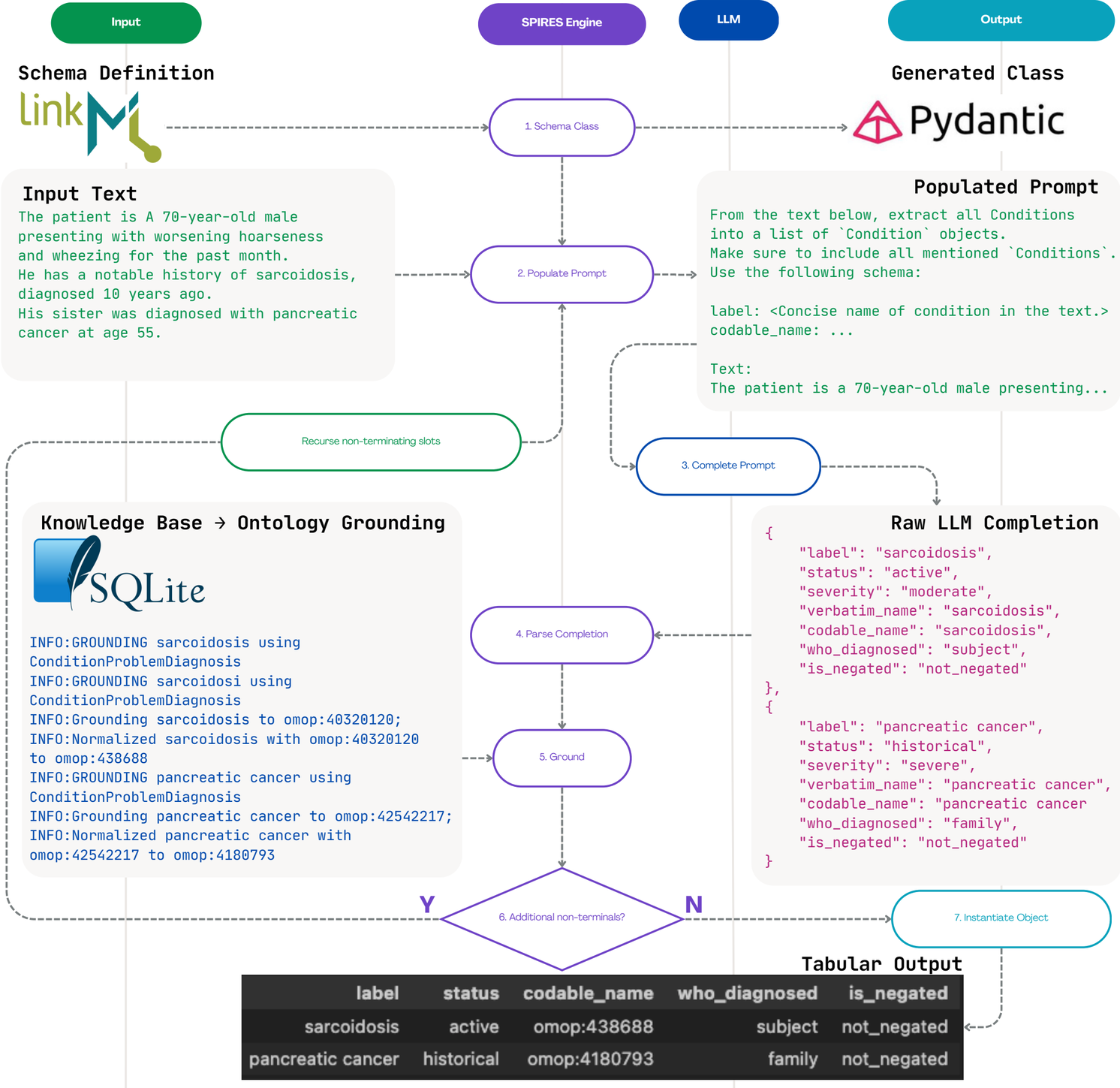
The primary extraction method currently implemented in OntoGPT is SPIRES:

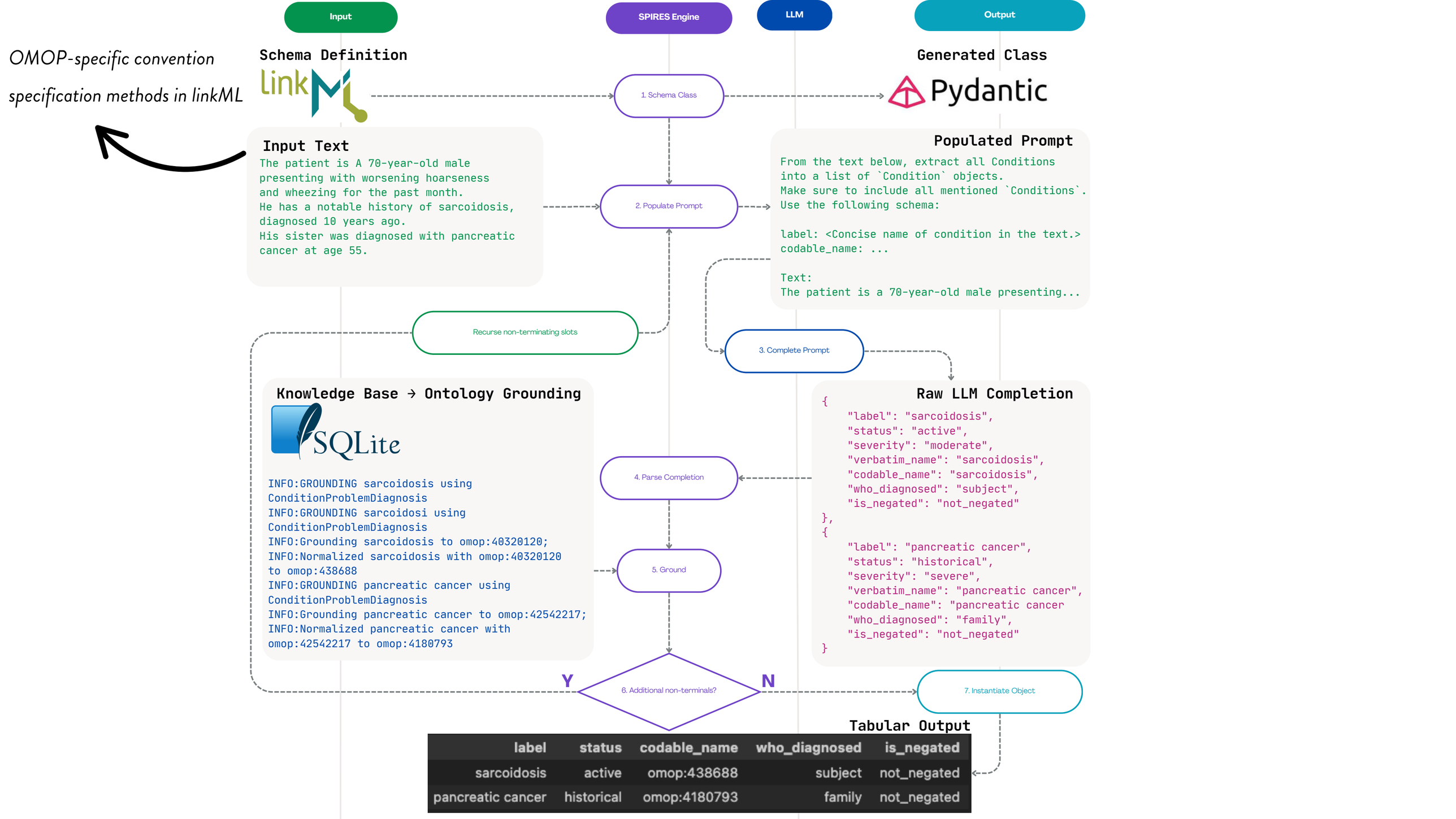
- SPIRES: *Structured Prompt Interrogation and Recursive Extraction of Semantics*
- A Zero-shot learning (ZSL) approach to extracting nested semantic structures from text
- This approach takes two inputs - 1) LinkML schema 2) free text, and outputs knowledge in a structure conformant with the supplied schema in JSON, YAML, RDF or OWL formats
- Uses OpenAI GPT models through their API, or one of a variety of LLMs on your local machine

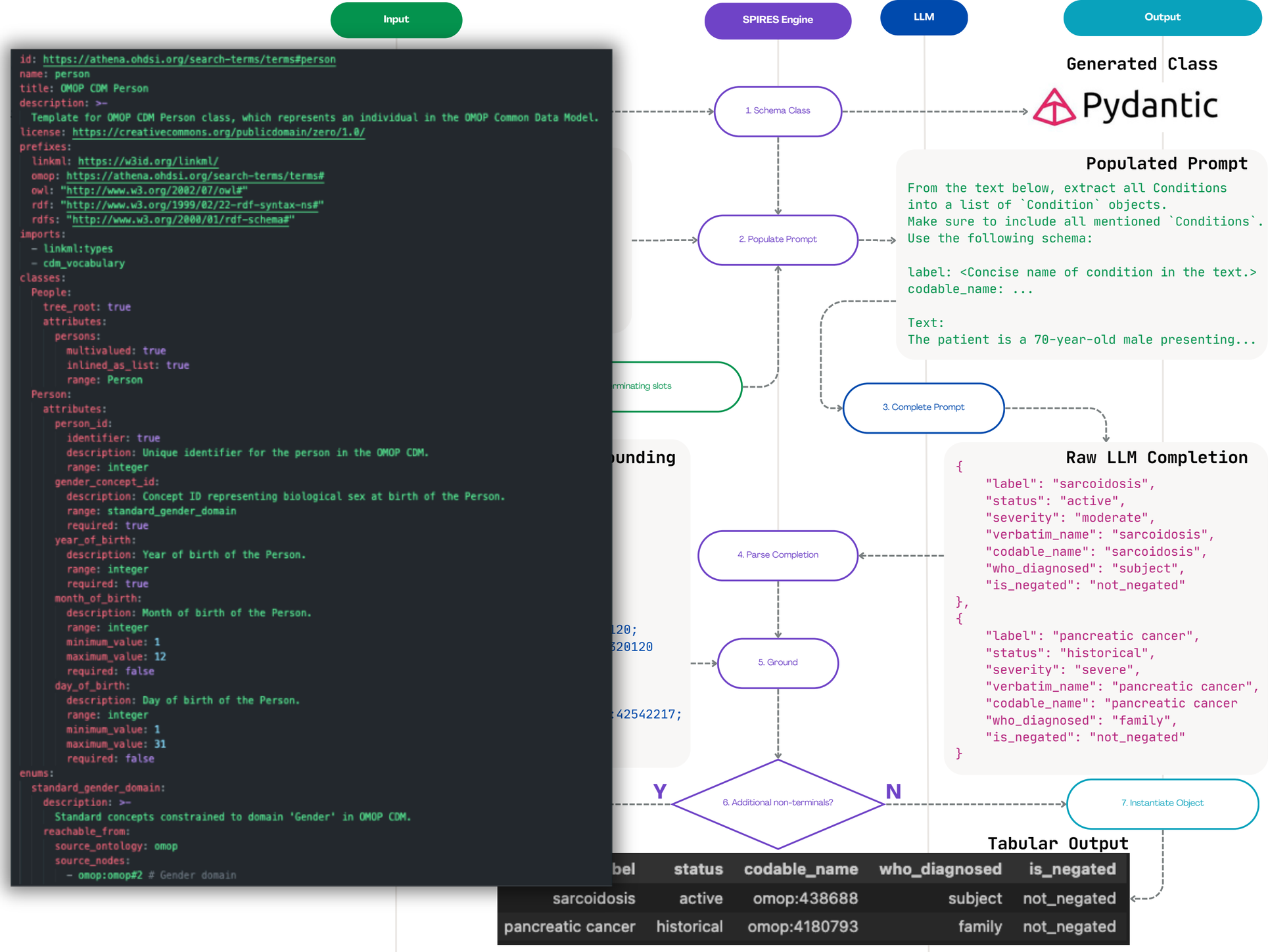
- *Works OK, with some limitations*
  - *Brittle parsing of outputs*
  - *No OMOP-specific grounding files*
  - *Issues working in restricted / airgapped environments*
  - *Strict cardinality requirements don't reflect reality of clinical notes*

(N.B. for the most part, these limitations are specific to the implementation, not conceptual. i.e. solvable)

SPIRES (from: Monarch Initiative)


Input
SPIRES Engine
LLM
Output
Schema Definition
LinkML
1. Schema Class
Generated Class
Pydantic
Input Text
The patient is A 70-year-old male
presenting with worsening hoarseness
and wheezing for the past month.
He has a notable history of sarcoidosis,
diagnosed 10 years ago.
His sister was diagnosed with pancreatic
cancer at age 55.
2. Populate Prompt
Populated Prompt
From the text below, extract all Conditions
into a list of `Condition` objects.
Make sure to include all mentioned `Conditions`.
Use the following schema:
label: <Concise name of condition in the text.>
codable_name: ...
Text:
The patient is a 70-year-old male presenting...
Recurse non-terminating slots
3. Complete Prompt
Knowledge Base → Ontology Grounding
SQLite
INFO:GROUNDING sarcoidosis using
ConditionProblemDiagnosis
INFO:GROUNDING sarcoidosi using
ConditionProblemDiagnosis
INFO:Grounding sarcoidosis to omop:40320120;
INFO:Normalized sarcoidosis with omop:40320120
to omop:438688
INFO:GROUNDING pancreatic cancer using
ConditionProblemDiagnosis
INFO:Grounding pancreatic cancer to omop:42542217;
INFO:Normalized pancreatic cancer with
omop:42542217 to omop:4180793
Raw LLM Completion
{
"label": "sarcoidosis",
"status": "active",
"severity": "moderate",
"verbatim_name": "sarcoidosis",
"codable_name": "sarcoidosis",
"who_diagnosed": "subject",
"is_negated": "not_negated"
},
{
"label": "pancreatic cancer",
"status": "historical",
"severity": "severe",
"verbatim_name": "pancreatic cancer",
"codable_name": "pancreatic cancer
"who_diagnosed": "family",
"is_negated": "not_negated"
}
4. Parse Completion
5. Ground
Y
6. Additional non-terminals?
N
7. Instantiate Object
Tabular Output
label
status
codable_name
who_diagnosed
is_negated
sarcoidosis
active
omop:438688
subject
not_negated
pancreatic cancer
historical
omop:4180793
family
not_negated


Input
SPIRES Engine
LLM
Output
OMOP-specific convention
specification methods in linkML
Schema Definition
linkML
1. Schema Class
Generated Class
Pydantic
Input Text
The patient is A 70-year-old male
presenting with worsening hoarseness
and wheezing for the past month.
He has a notable history of sarcoidosis,
diagnosed 10 years ago.
His sister was diagnosed with pancreatic
cancer at age 55.
2. Populate Prompt
Populated Prompt
From the text below, extract all Conditions
into a list of `Condition` objects.
Make sure to include all mentioned `Conditions`.
Use the following schema:
label: <Concise name of condition in the text.>
codable_name: ...
Text:
The patient is a 70-year-old male presenting...
Recurse non-terminating slots
3. Complete Prompt
Knowledge Base → Ontology Grounding
SQLite
INFO:GROUNDING sarcoidosis using
ConditionProblemDiagnosis
INFO:GROUNDING sarcoidosi using
ConditionProblemDiagnosis
INFO:Grounding sarcoidosis to omop:40320120;
INFO:Normalized sarcoidosis with omop:40320120
to omop:438688
INFO:GROUNDING pancreatic cancer using
ConditionProblemDiagnosis
INFO:Grounding pancreatic cancer to omop:42542217;
INFO:Normalized pancreatic cancer with
omop:42542217 to omop:4180793
Raw LLM Completion
{
"label": "sarcoidosis",
"status": "active",
"severity": "moderate",
"verbatim_name": "sarcoidosis",
"codable_name": "sarcoidosis",
"who_diagnosed": "subject",
"is_negated": "not_negated"
},
{
"label": "pancreatic cancer",
"status": "historical",
"severity": "severe",
"verbatim_name": "pancreatic cancer",
"codable_name": "pancreatic cancer
"who_diagnosed": "family",
"is_negated": "not_negated"
}
4. Parse Completion
5. Ground
Y
6. Additional non-terminals?
N
7. Instantiate Object
Tabular Output
label
status
codable_name
who_diagnosed
is_negated
sarcoidosis
active
omop:438688
subject
not_negated
pancreatic cancer
historical
omop:4180793
family
not_negated




```yaml
id: https://athena.ohdsi.org/search-terms/terms#person
name: person
title: OMOP CDM Person
description: >-
  Template for OMOP CDM Person class, which represents an individual in the OMOP Common Data Model.
license: https://creativecommons.org/publicdomain/zero/1.0/
prefixes:
  linkml: https://w3id.org/linkml/
  omop: https://athena.ohdsi.org/search-terms/terms#
  owl: "http://www.w3.org/2002/07/owl#"
  rdf: "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  rdfs: "http://www.w3.org/2000/01/rdf-schema#"
imports:
  - linkml:types
  - cdm_vocabulary
classes:
  People:
    tree_root: true
    attributes:
      persons:
        multivalued: true
        inlined_as_list: true
        range: Person
  Person:
    attributes:
      person_id:
        identifier: true
        description: Unique identifier for the person in the OMOP CDM.
        range: integer
      gender_concept_id:
        description: Concept ID representing biological sex at birth of the Person.
        range: standard_gender_domain
        required: true
      year_of_birth:
        description: Year of birth of the Person.
        range: integer
        required: true
      month_of_birth:
        description: Month of birth of the Person.
        range: integer
        minimum_value: 1
        maximum_value: 12
        required: false
      day_of_birth:
        description: Day of birth of the Person.
        range: integer
        minimum_value: 1
        maximum_value: 31
        required: false
enums:
  standard_gender_domain:
    description: >-
      Standard concepts constrained to domain 'Gender' in OMOP CDM.
    reachable_from:
      source_ontology: omop
      source_nodes:
        - omop:omop#2 # Gender domain
```

OMOP-specific convention specification methods in linkML

Input

SPIRES Engine

LLM

Output

Schema Definition

linkML

Generated Class

Pydantic

1. Schema Class

Input Text

```
The patient is A 70-year-old male
presenting with worsening hoarseness
and wheezing for the past month
```

2. Populate Prompt

Populated Prompt

```
From the text below, extract all Conditions
into a list of `Condition` objects.
Make sure to include all mentioned `Conditions`.
Use the following schema:

label: <Concise name of condition in the text.>
codable_name: ...

Text:
The patient is a 70-year-old male presenting...
```

3. Complete Prompt

Raw LLM Completion

```
{
    "label": "sarcoidosis",
    "status": "active",
    "severity": "moderate",
    "verbatim_name": "sarcoidosis",
    "codable_name": "sarcoidosis",
    "who_diagnosed": "subject",
    "is_negated": "not_negated"
},
{
    "label": "pancreatic cancer",
    "status": "historical",
    "severity": "severe",
    "verbatim_name": "pancreatic cancer",
    "codable_name": "pancreatic cancer
    "who_diagnosed": "family",
    "is_negated": "not_negated"
}
```

4. Parse Completion

5. Ground

6. Additional non-terminals?

N

7. Instantiate Object

Tabular Output

label	status	codable_name	who_diagnosed	is_negated
sarcoidosis	active	omop:438688	subject	not_negated
pancreatic cancer	historical	omop:4180793	family	not_negated

linkml documentation

Search

CONTENTS:

- LinkML at a glance
- Quick Install Guide
- Tutorial
- LinkML Schemas
- Working with Data
  - Converting between different representations
  - **Data Validation**
  - Working with RDF and LinkML
  - CSVs and Tabular Data
  - Python
  - Working with data in SQL Databases
- Generators
- How-to Guides
- Examples of use
- The LinkML Ecosystem
- LinkML specification
- Contribution Guidelines
- FAQ

DEVELOPERS GUIDE:

- CLI

The `linkml.validator` package and CLI

This package contains the main entry point for various flexible validation strategies.

Validation in Python code

If you are writing your own Python code to perform validation the simplest approach is to use the `linkml.validator.validate()` function. For example:

```python
from linkml.validator import validate

instance = {
    "id": "ORCID:1234",
    "full_name": "Clark Kent",
    "age": 32,
    "phone": "555-555-5555",
}

report = validate(instance, "personinfo.yaml", "Person")

if not report.results:
    print('The instance is valid!')
else:
    for result in report.results:
        print(result.message)
```

This function takes a single instance (typically represented as a Python dict) and validates it according to the given schema (specified here by a path to the source file, but dict or object representation of the schema is also accepted). This example also explicitly specifies which class within the schema (`Person`) the data instance should adhere to. If this is omitted, the function will attempt to infer it.

The other high-level function is `linkml.validator.validate_file()`. It loads data instances from a file and validates each of them according to a class in a schema. Assuming the contents of `people.csv` look like:

```
id,full_name,age,phone
ORCID:1234,Clark Kent,32,555-555-5555
ORCID:5678,Lois Lane,33,555-555-1234
```

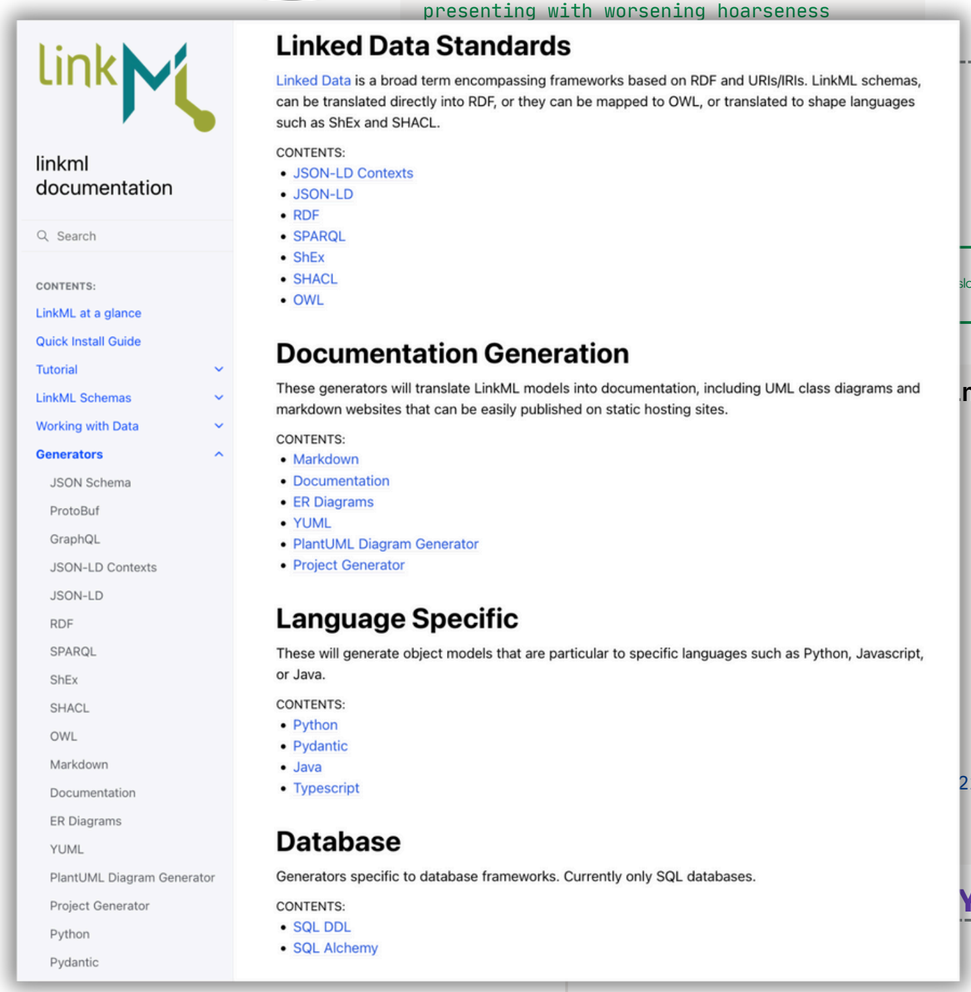
Each row can be validated with:

```python
from linkml.validator import validate_file

report = validate_file("people.csv", "personinfo.yaml", "Person")
```

Under the hood, both of these functions use a strategy of generating a JSON Schema artifact from the LinkML schema and validating instances using a JSON Schema validator.

OMOP-specific convention

specification methods in linkML



Input

SPIRES Engine

LLM

Output

Schema Definition

linkML

1. Schema Class

Generated Class

Pydantic

Input Text

```
The patient is A 70-year-old male
presenting with worsening hoarseness
```

2. Populate Prompt

Populated Prompt

```
From the text below, extract all Conditions
into a list of `Condition` objects.
Make sure to include all mentioned `Conditions`.
Use the following schema:

label: <Concise name of condition in the text.>
codable_name: ...

Text:
The patient is a 70-year-old male presenting...
```

3. Complete Prompt

Raw LLM Completion

```
{
    "label": "sarcoidosis",
    "status": "active",
    "severity": "moderate",
    "verbatim_name": "sarcoidosis",
    "codable_name": "sarcoidosis",
    "who_diagnosed": "subject",
    "is_negated": "not_negated"
},
{
    "label": "pancreatic cancer",
    "status": "historical",
    "severity": "severe",
    "verbatim_name": "pancreatic cancer",
    "codable_name": "pancreatic cancer
    "who_diagnosed": "family",
    "is_negated": "not_negated"
}
```

4. Parse Completion

5. Ground

6. Additional non-terminals?

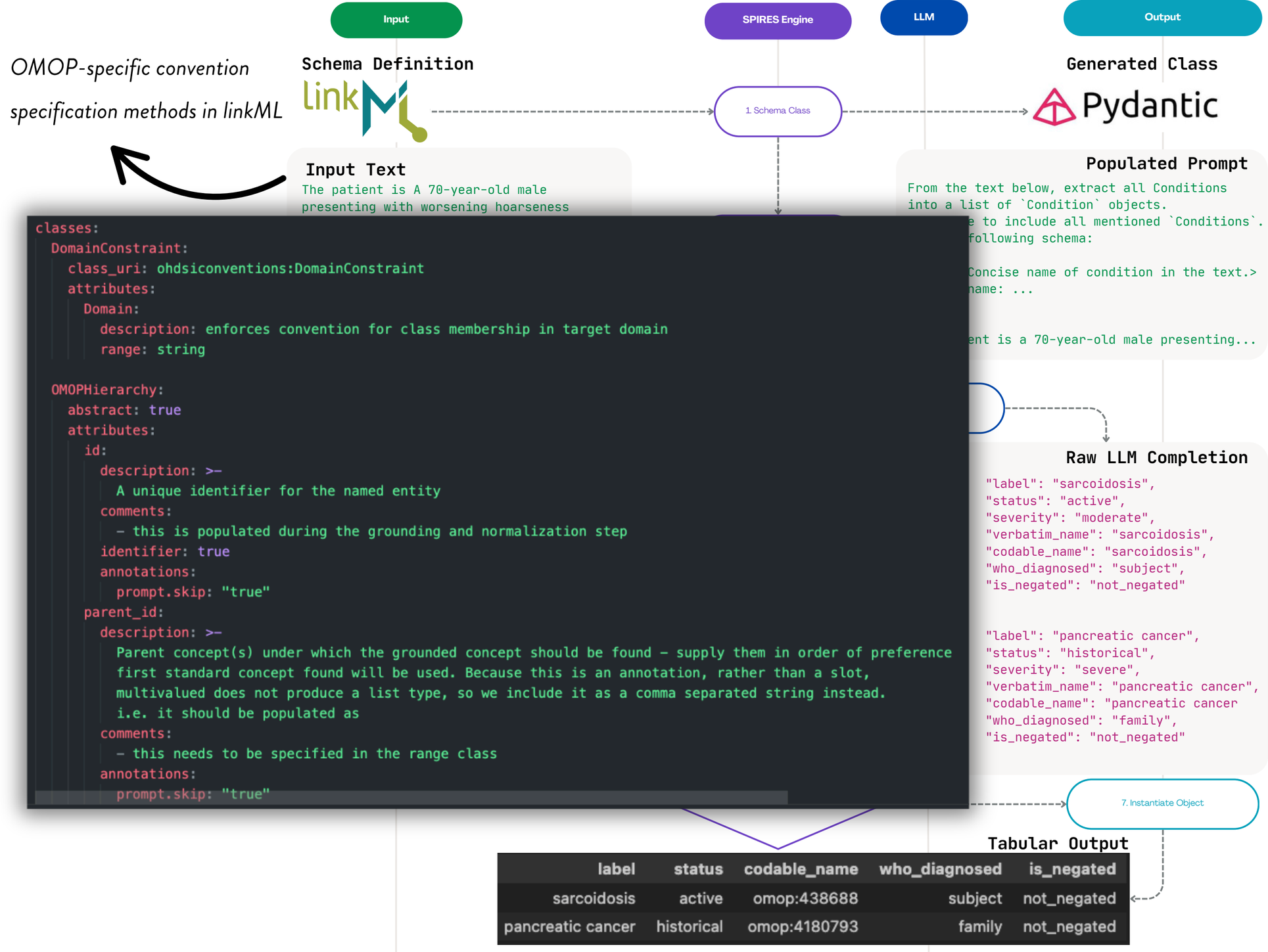
Y

N

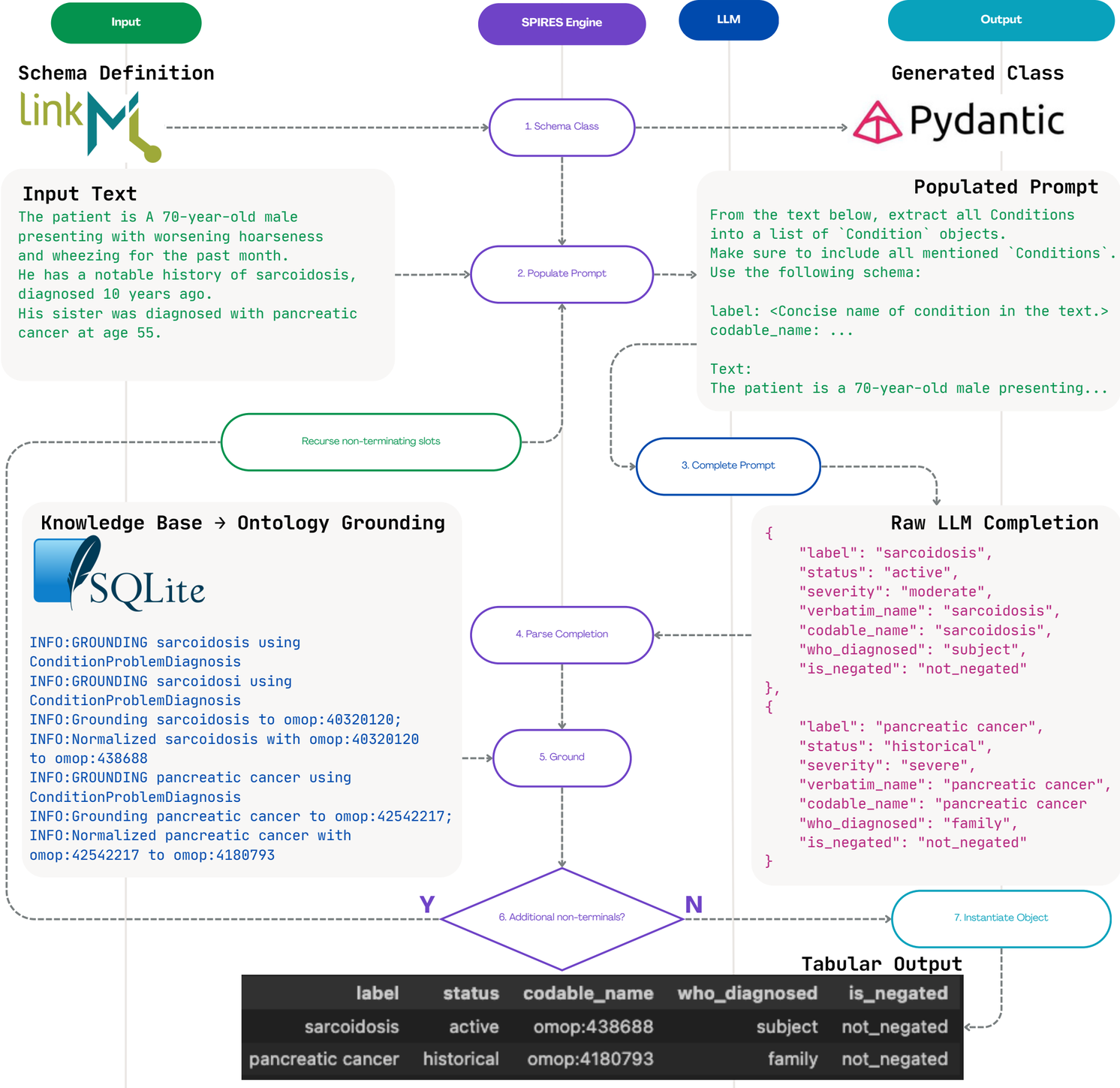
7. Instantiate Object

Tabular Output

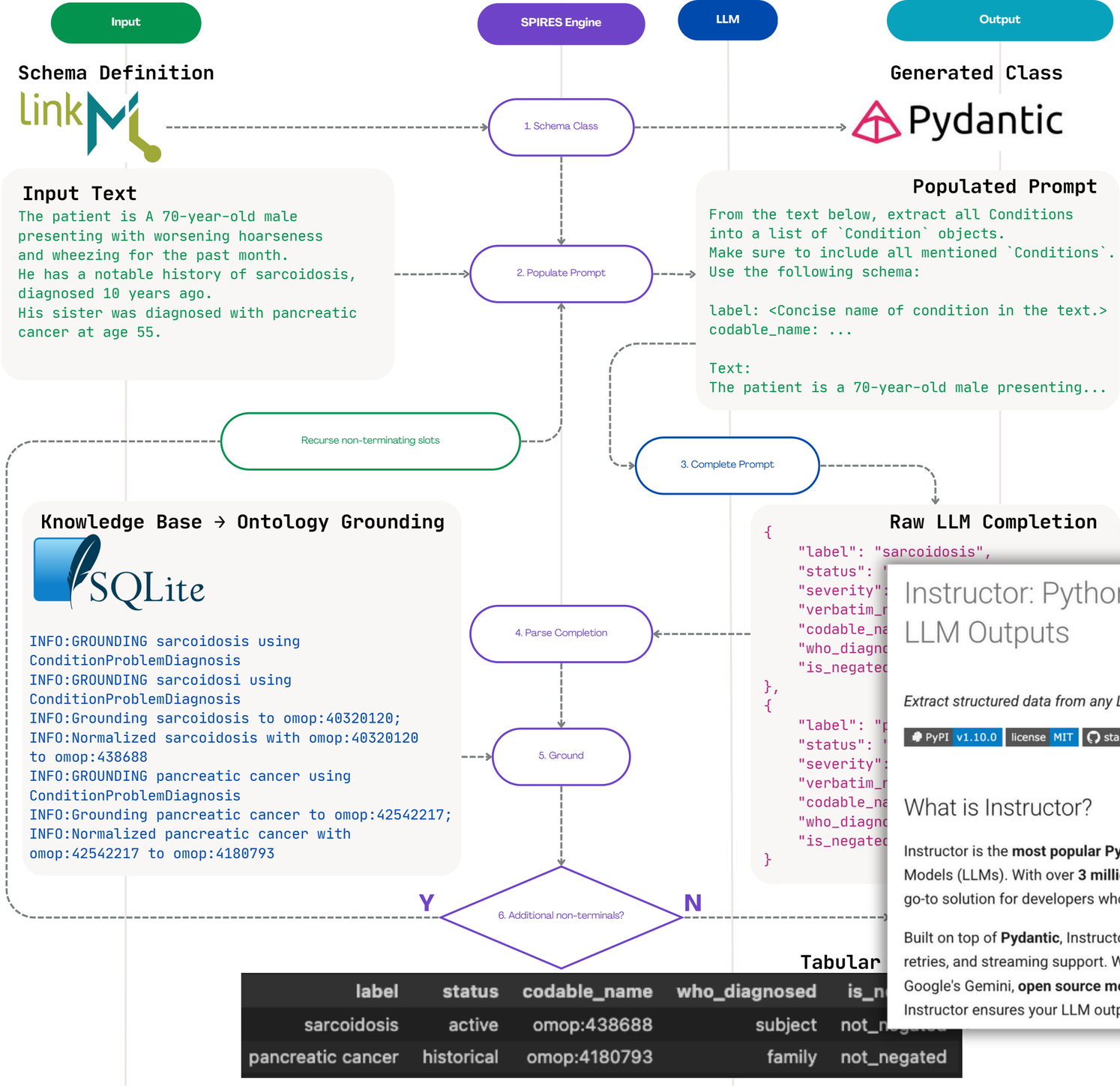
label	status	codable_name	who_diagnosed	is_negated
sarcoidosis	active	omop:438688	subject	not_negated
pancreatic cancer	historical	omop:4180793	family	not_negated



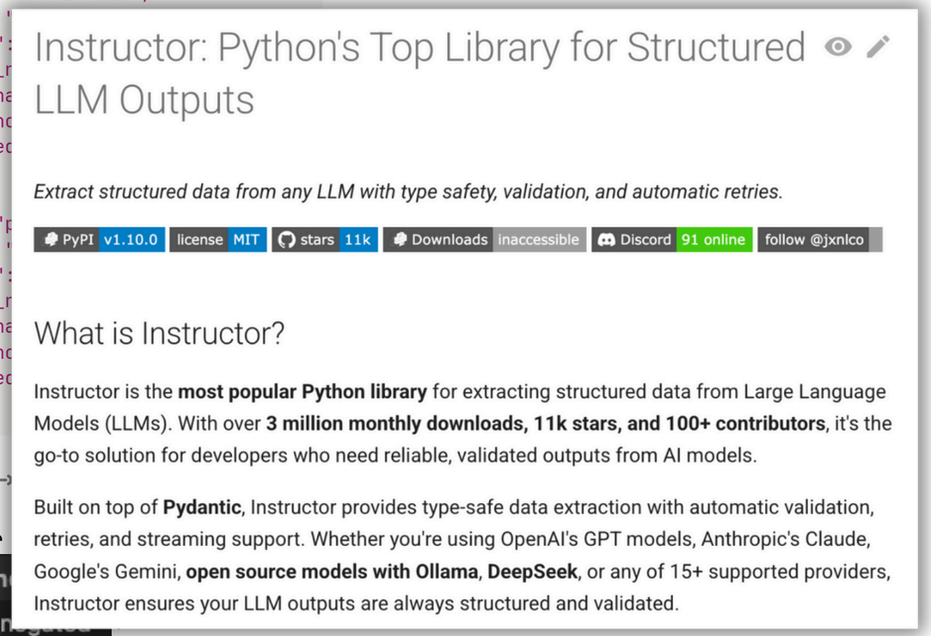

```yaml
classes:
  DomainConstraint:
    class_uri: ohdsiconventions:DomainConstraint
    attributes:
      Domain:
        description: enforces convention for class membership in target domain
        range: string

  OMOPHierarchy:
    abstract: true
    attributes:
      id:
        description: >-
          A unique identifier for the named entity
        comments:
          - this is populated during the grounding and normalization step
        identifier: true
        annotations:
          prompt.skip: "true"
      parent_id:
        description: >-
          Parent concept(s) under which the grounded concept should be found - supply them in order of preference
          first standard concept found will be used. Because this is an annotation, rather than a slot,
          multivalued does not produce a list type, so we include it as a comma separated string instead.
          i.e. it should be populated as
        comments:
          - this needs to be specified in the range class
        annotations:
          prompt.skip: "true"
```

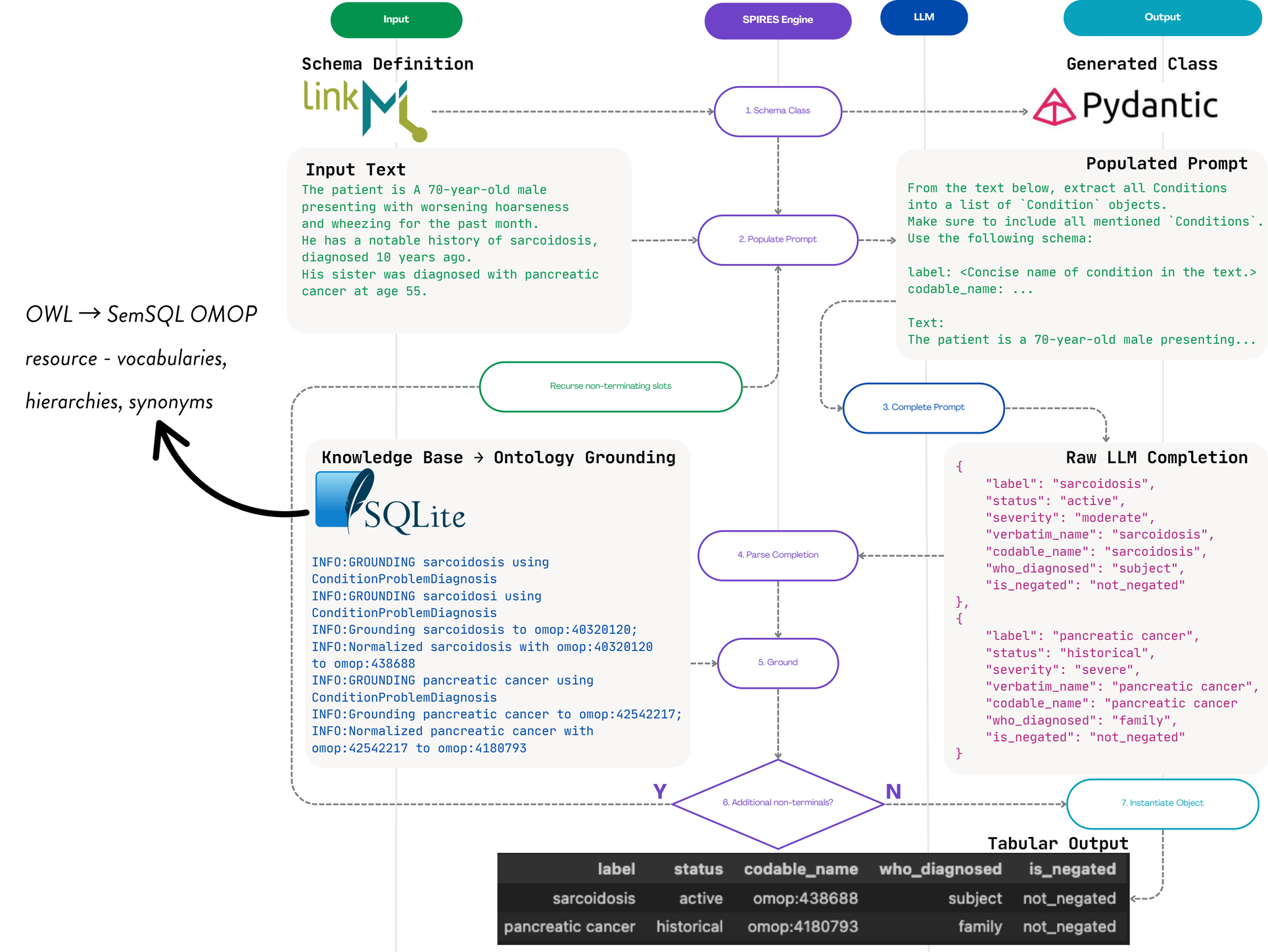


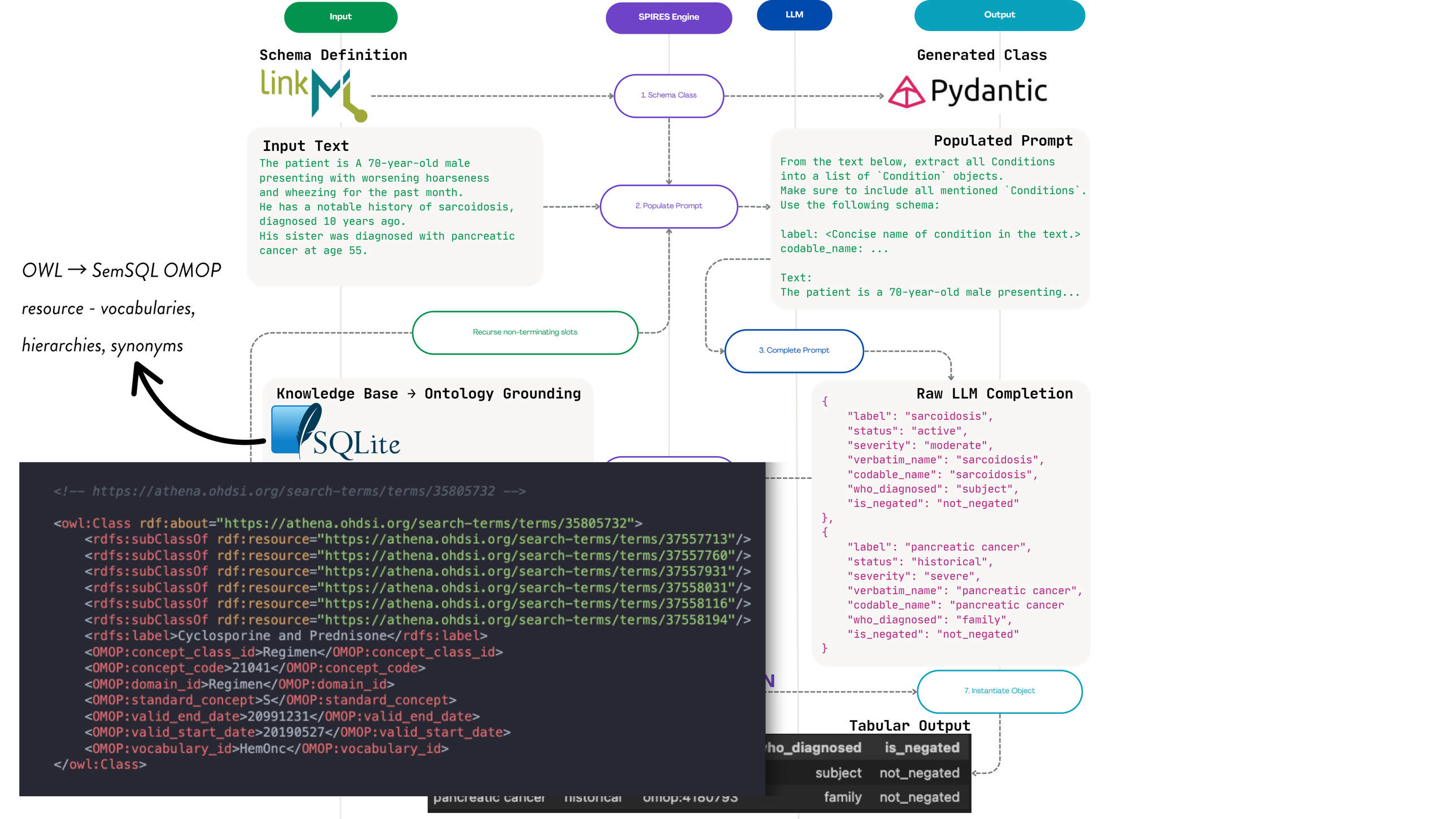
Improved robustness with updated LLM interrogation libraries

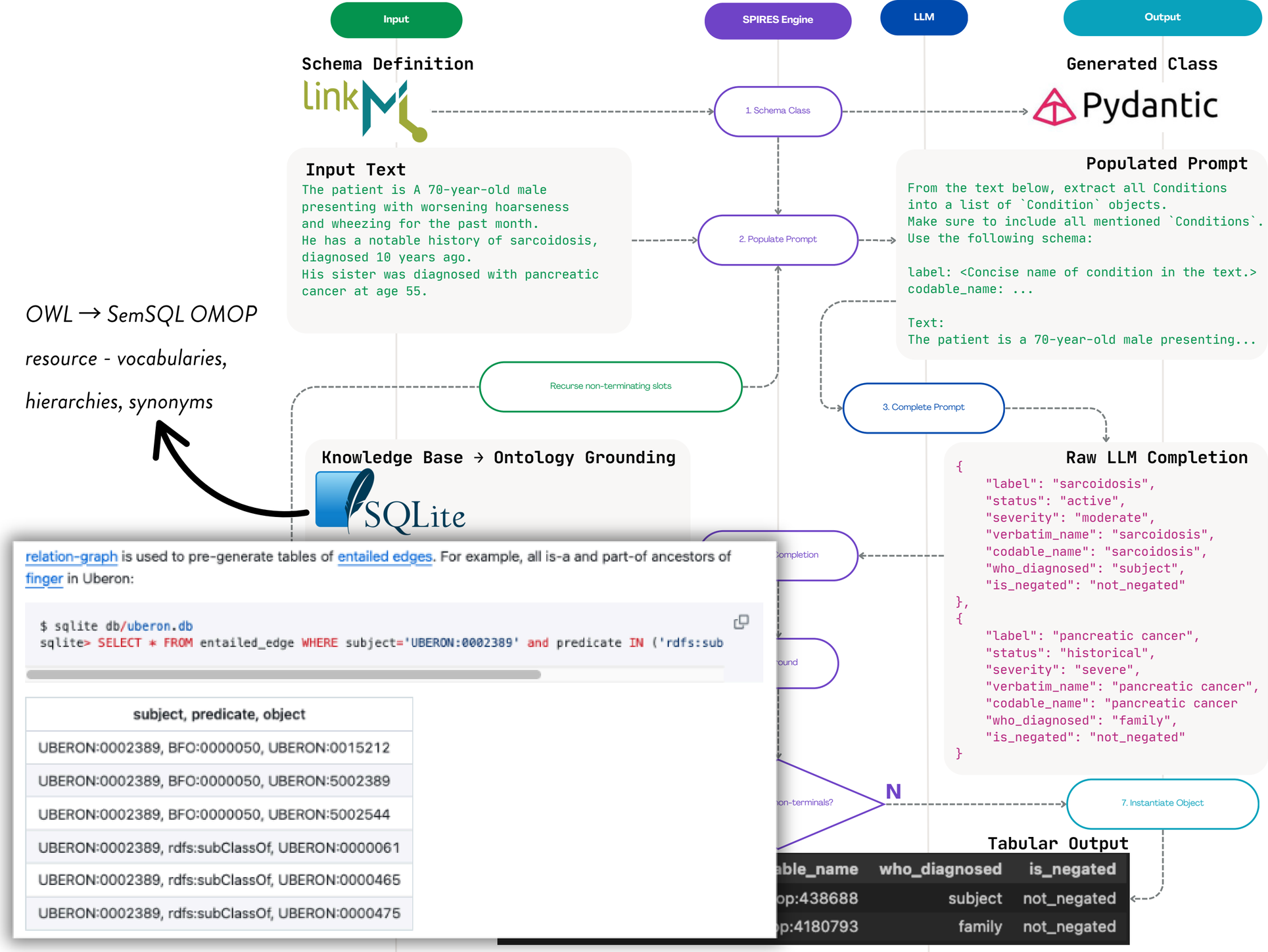


Improved robustness with updated LLM interrogation libraries




Input
SPIRES Engine
LLM
Output
Schema Definition
Generated Class
1. Schema Class
Pydantic
Input Text
The patient is A 70-year-old male presenting with worsening hoarseness and wheezing for the past month.
He has a notable history of sarcoidosis, diagnosed 10 years ago.
His sister was diagnosed with pancreatic cancer at age 55.
2. Populate Prompt
Populated Prompt
From the text below, extract all Conditions into a list of `Condition` objects.
Make sure to include all mentioned `Conditions`.
Use the following schema:
label: <Concise name of condition in the text.>
codable_name: ...
Text:
The patient is a 70-year-old male presenting...
Recurse non-terminating slots
3. Complete Prompt
OWL → SemSQL OMOP resource - vocabularies, hierarchies, synonyms
Knowledge Base → Ontology Grounding
SQLite
INFO:GROUNDING sarcoidosis using ConditionProblemDiagnosis
INFO:GROUNDING sarcoidosi using ConditionProblemDiagnosis
INFO:Grounding sarcoidosis to omop:40320120;
INFO:Normalized sarcoidosis with omop:40320120 to omop:438688
INFO:GROUNDING pancreatic cancer using ConditionProblemDiagnosis
INFO:Grounding pancreatic cancer to omop:42542217;
INFO:Normalized pancreatic cancer with omop:42542217 to omop:4180793
Raw LLM Completion
{
"label": "sarcoidosis",
"status": "active",
"severity": "moderate",
"verbatim_name": "sarcoidosis",
"codable_name": "sarcoidosis",
"who_diagnosed": "subject",
"is_negated": "not_negated"
},
{
"label": "pancreatic cancer",
"status": "historical",
"severity": "severe",
"verbatim_name": "pancreatic cancer",
"codable_name": "pancreatic cancer
"who_diagnosed": "family",
"is_negated": "not_negated"
}
4. Parse Completion
5. Ground
6. Additional non-terminals?
Y
N
7. Instantiate Object
Tabular Output
label | status | codable_name | who_diagnosed | is_negated
sarcoidosis | active | omop:438688 | subject | not_negated
pancreatic cancer | historical | omop:4180793 | family | not_negated

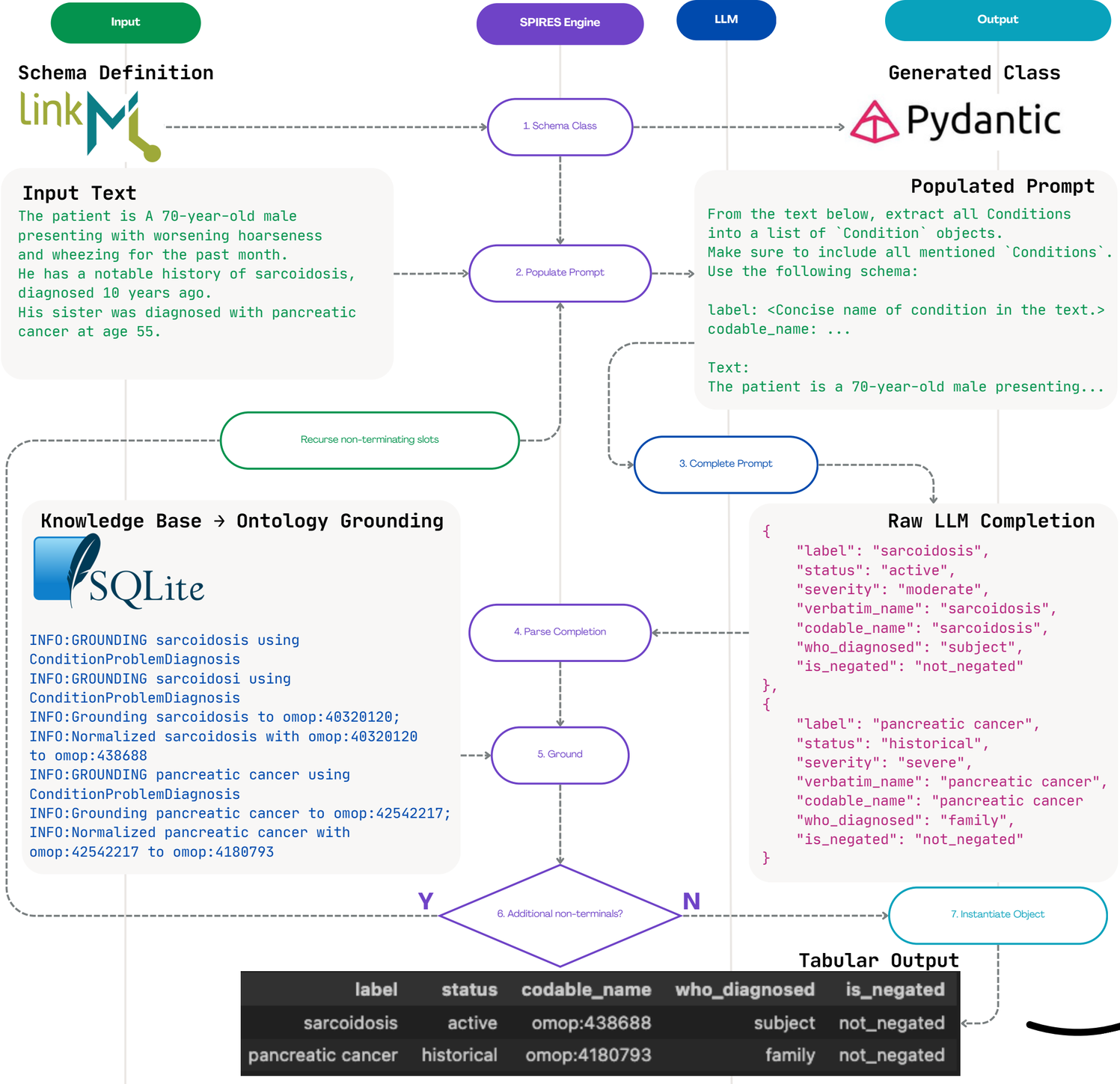

Input
SPIRES Engine
LLM
Output
Schema Definition
linkML
1. Schema Class
Generated Class
Pydantic
Input Text
The patient is A 70-year-old male
presenting with worsening hoarseness
and wheezing for the past month.
He has a notable history of sarcoidosis,
diagnosed 10 years ago.
His sister was diagnosed with pancreatic
cancer at age 55.
2. Populate Prompt
Populated Prompt
From the text below, extract all Conditions
into a list of `Condition` objects.
Make sure to include all mentioned `Conditions`.
Use the following schema:
label: <Concise name of condition in the text.>
codable_name: ...
Text:
The patient is a 70-year-old male presenting...
Recurse non-terminating slots
3. Complete Prompt
OWL → SemSQL OMOP
resource - vocabularies,
hierarchies, synonyms
Knowledge Base → Ontology Grounding
SQLite
Raw LLM Completion
{
"label": "sarcoidosis",
"status": "active",
"severity": "moderate",
"verbatim_name": "sarcoidosis",
"codable_name": "sarcoidosis",
"who_diagnosed": "subject",
"is_negated": "not_negated"
},
{
"label": "pancreatic cancer",
"status": "historical",
"severity": "severe",
"verbatim_name": "pancreatic cancer",
"codable_name": "pancreatic cancer
"who_diagnosed": "family",
"is_negated": "not_negated"
}
<!-- https://athena.ohdsi.org/search-terms/terms/35805732 -->
<owl:Class rdf:about="https://athena.ohdsi.org/search-terms/terms/35805732">
<rdfs:subClassOf rdf:resource="https://athena.ohdsi.org/search-terms/terms/37557713"/>
<rdfs:subClassOf rdf:resource="https://athena.ohdsi.org/search-terms/terms/37557760"/>
<rdfs:subClassOf rdf:resource="https://athena.ohdsi.org/search-terms/terms/37557931"/>
<rdfs:subClassOf rdf:resource="https://athena.ohdsi.org/search-terms/terms/37558031"/>
<rdfs:subClassOf rdf:resource="https://athena.ohdsi.org/search-terms/terms/37558116"/>
<rdfs:subClassOf rdf:resource="https://athena.ohdsi.org/search-terms/terms/37558194"/>
<rdfs:label>Cyclosporine and Prednisone</rdfs:label>
<OMOP:concept_class_id>Regimen</OMOP:concept_class_id>
<OMOP:concept_code>21041</OMOP:concept_code>
<OMOP:domain_id>Regimen</OMOP:domain_id>
<OMOP:standard_concept>S</OMOP:standard_concept>
<OMOP:valid_end_date>20991231</OMOP:valid_end_date>
<OMOP:valid_start_date>20190527</OMOP:valid_start_date>
<OMOP:vocabulary_id>HemOnc</OMOP:vocabulary_id>
</owl:Class>
7. Instantiate Object
Tabular Output
who_diagnosed
is_negated
subject
not_negated
family
not_negated



relation-graph is used to pre-generate tables of entailed edges. For example, all is-a and part-of ancestors of finger in Uberon:

```
$ sqlite db/uberon.db
sqlite> SELECT * FROM entailed_edge WHERE subject='UBERON:0002389' and predicate IN ('rdfs:sub
```

subject, predicate, object
UBERON:0002389, BFO:0000050, UBERON:0015212
UBERON:0002389, BFO:0000050, UBERON:5002389
UBERON:0002389, BFO:0000050, UBERON:5002544
UBERON:0002389, rdfs:subClassOf, UBERON:0000061
UBERON:0002389, rdfs:subClassOf, UBERON:0000465
UBERON:0002389, rdfs:subClassOf, UBERON:0000475


Input
SPIRES Engine
LLM
Output
Schema Definition
Generated Class
1. Schema Class
Input Text
The patient is A 70-year-old male
presenting with worsening hoarseness
and wheezing for the past month.
He has a notable history of sarcoidosis,
diagnosed 10 years ago.
His sister was diagnosed with pancreatic
cancer at age 55.
2. Populate Prompt
Populated Prompt
From the text below, extract all Conditions
into a list of `Condition` objects.
Make sure to include all mentioned `Conditions`.
Use the following schema:
label: <Concise name of condition in the text.>
codable_name: ...
Text:
The patient is a 70-year-old male presenting...
Recurse non-terminating slots
3. Complete Prompt
Knowledge Base → Ontology Grounding
SQLite
INFO:GROUNDING sarcoidosis using
ConditionProblemDiagnosis
INFO:GROUNDING sarcoidosi using
ConditionProblemDiagnosis
INFO:Grounding sarcoidosis to omop:40320120;
INFO:Normalized sarcoidosis with omop:40320120
to omop:438688
INFO:GROUNDING pancreatic cancer using
ConditionProblemDiagnosis
INFO:Grounding pancreatic cancer to omop:42542217;
INFO:Normalized pancreatic cancer with
omop:42542217 to omop:4180793
Raw LLM Completion
{
"label": "sarcoidosis",
"status": "active",
"severity": "moderate",
"verbatim_name": "sarcoidosis",
"codable_name": "sarcoidosis",
"who_diagnosed": "subject",
"is_negated": "not_negated"
},
{
"label": "pancreatic cancer",
"status": "historical",
"severity": "severe",
"verbatim_name": "pancreatic cancer",
"codable_name": "pancreatic cancer
"who_diagnosed": "family",
"is_negated": "not_negated"
}
4. Parse Completion
5. Ground
Y
6. Additional non-terminals?
N
7. Instantiate Object
Tabular Output
label
status
codable_name
who_diagnosed
is_negated
sarcoidosis
active
omop:438688
subject
not_negated
pancreatic cancer
historical
omop:4180793
family
not_negated
Modified grounding steps to handle
OMOP-specific constraints

Input

SPIRES Engine

LLM

Output

Schema Definition

LinkML

Generated Class

1. Schema Class

```python
STANDARD = 'omop:standard_concept'
IN_DOMAIN = 'omop:in_domain'
IN_CLASS = 'omop:in_class'
IN_VOCABULARY = 'omop:in_vocabulary'

def is_standard_concept(annotator: BasicOntologyInterface, concept_id: str) -> bool:
    # expects concept_id as a CURIE i.e. 'omop:123456'
    return annotator.entity_metadata_map(concept_id).get(STANDARD, None) is not None

def omop_metadata(annotator: BasicOntologyInterface, concept_id: str, target: str, predicate=IN_DOMAIN) -> bool:
    for _, _, obj in annotator.relationships([concept_id], predicates=[predicate]):
        if obj == target:
            return True
    return False

def has_ancestry(annotator: BasicOntologyInterface, concept_id: str, parent_id: str) -> bool:
    ancestors = list(annotator.ancestors([concept_id], predicates=[IS_A]))
    return parent_id in ancestors

def first_best_parent(annotators, parents, object_id):
    for annotator in annotators:
        for parent in parents:
            if has_ancestry(annotator, object_id, parent):
                return parent

def first_best_annotation(annotators, parents, annotations):
    """Return the first annotator that has the best annotation for the given object_id."""
    best_so_far = None
    for ann in annotations:
        for annotator in annotators:
            if is_standard_concept(annotator, ann.object_id):
                if best_so_far is None:
                    best_so_far = ann
                parent = first_best_parent(annotators, parents, ann.object_id)
                if parent:
                    return ann, parent
    return best_so_far, None
```

Input Text

The patient is A 70-year-old male
presenting with worsening hoarseness
and wheezing for the past month.
He has a notable history of sarcoidosis,
diagnosed 10 years ago.
His sister was diagnosed with pancreatic
cancer at age 55.

2. Populate Prompt

Recurse non-terminating slots

Knowledge Base → Ontology Grounding

SQLite

```
INFO:GROUNDING sarcoidosis using
ConditionProblemDiagnosis
INFO:GROUNDING sarcoidosi using
ConditionProblemDiagnosis
INFO:Grounding sarcoidosis to omop:40320120;
INFO:Normalized sarcoidosis with omop:40320120
to omop:438688
INFO:GROUNDING pancreatic cancer using
ConditionProblemDiagnosis
INFO:Grounding pancreatic cancer to omop:42542217;
INFO:Normalized pancreatic cancer with
omop:42542217 to omop:4180793
```

4. Parse Completion

5. Ground

```
    "who_diagnosed": "family",
    "is_negated": "not_negated"
}
```

6. Additional non-terminals?

Y

N

7. Instantiate Object

Tabular Output

label	status	codable_name	who_diagnosed	is_negated
sarcoidosis	active	omop:438688	subject	not_negated
pancreatic cancer	historical	omop:4180793	family	not_negated

Modified grounding steps to handle OMOP-specific constraints

INPUT	llama3 label	IDs	evaluation	usagi label	IDs	evaluation	medspacy label	IDs	evaluation
rectum + pelvis	Pelvic region;Rectum structure	4044034;4144541	correct	PEL	4123163	invalid			no match
(l) lung	Lung structure	4213162	correct	Lung structure	4213162	correct	Entire lung	4111459	correct
ph i prostate+sv	Prostatic structure	4165732	valid	Prostatic structure	4165732	correct			no match
glottis	Glottis structure	4047227	correct	Glottis structure	4047227	correct	Entire glottis	4131315	correct
l chest wall	Chest wall structure	4193513	correct	Chest wall structure	4193513	correct	Entire chest wall	4109932	correct
lt femur	Bone structure of femur	4323581	correct	Bone structure of femur	4323581	correct	Entire bone of femur	37115374	correct
ph1 prostate	Prostatic structure	4165732	correct	Prostatic structure	4165732	correct	Entire prostate	4110208	correct
l/s spine	lumbar	4045660	correct	Structure of vertebral column	4227378	valid	Entire vertebral column	4185891	valid
ph2 prostate bed	Prostatic structure	4165732	correct	Prostatic structure	4165732	correct	Entire prostate	4110208	correct
distal oesophagus	Esophageal structure	4140098	correct	Esophageal structure	4140098	correct	Esophageal structure	4140098	correct
rtbreast+scf+imc+sib	Breast structure	4298444	valid	Breast structure	4298444	correct			no match
prostate + pelvis	Prostatic structure;Pelvic region	4165732;4044034	correct	PEL	4123163	invalid	Entire prostate, Entire pelvis	4110208, 4041832	correct
rt nasal ala			no match	Lateral nasal artery	37157433	invalid			no match
t11-l3	thoracic;lumbar	4047490;4045660	correct	ST11	4159026	invalid	Level of the eleventh thoracic vertebra	4134469	near match
t9-l3	thoracic	4047490	correct	T9-T10 rotator thoracis	4077547	invalid			no match
upper pelvis	Pelvic region	4044034	correct	PEL	4123163	invalid	Entire pelvis	4041832	correct
rt parietal	Brain structure	4133034	correct	Structure of left parietal bone	37158682	near match			no match
right pelvis	Pelvic region	4044034	correct	Structure of right renal pelvis	4184440	invalid	Entire pelvis	4041832	correct
(r) breast/low axilla	Breast structure;Axillary region structure	4298444;4238919	correct	Axillary region structure	4238919	correct	Entire breast	4108283	correct
thyroid	Thyroid structure	4321375	correct	Thyroid structure	4321375	correct	Thyroid structure	4321375	correct

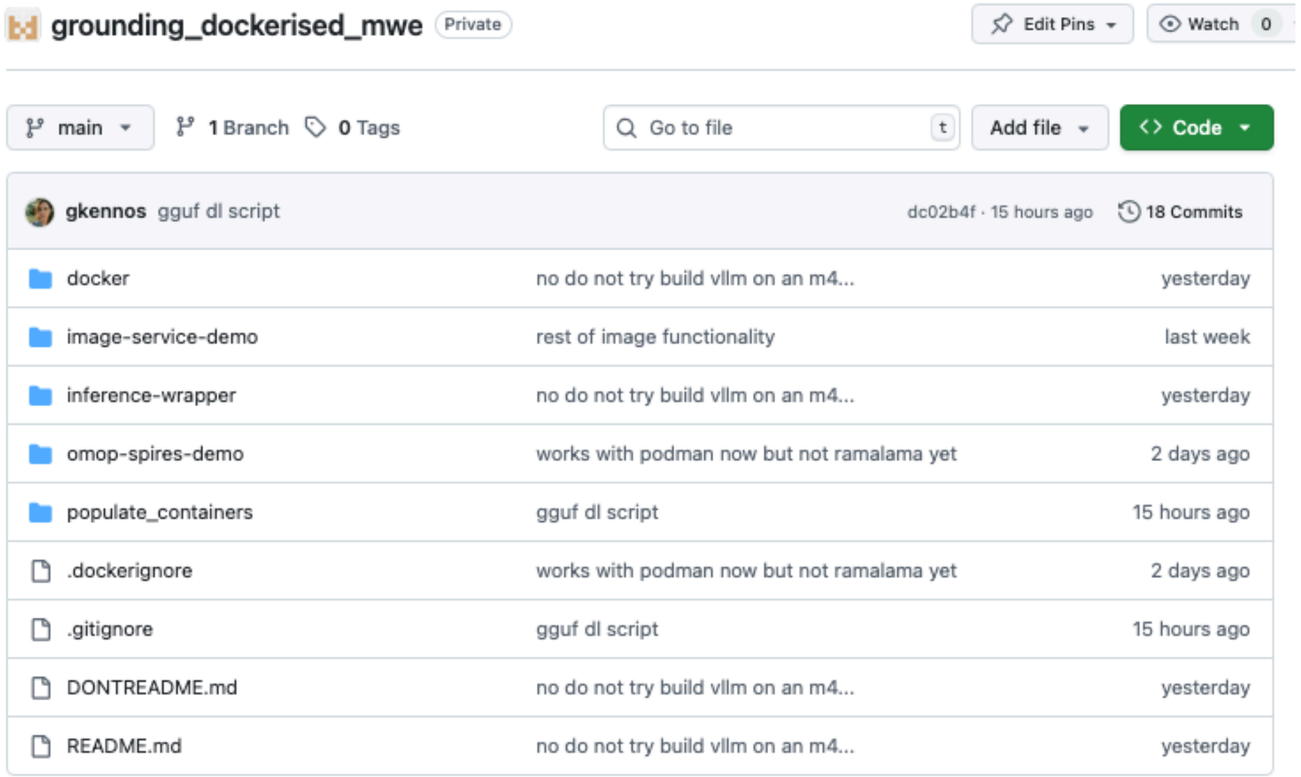
Proof of concept in Real Data - Radiotherapy Regions

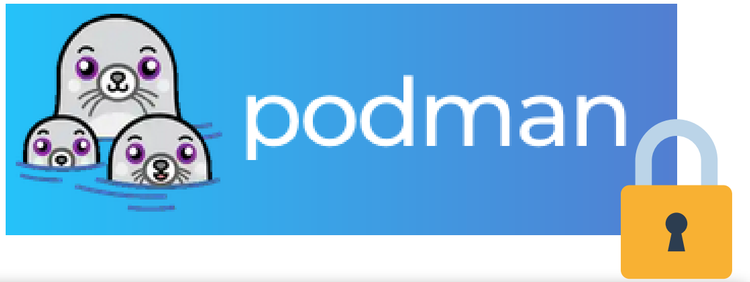
```yaml
classes:
  Region:
    tree_root: true
    attributes:
      label:
        description: >-
          The name of the radiation therapy region verbatim as it appears in the text.
        range: string
      # give the model two chances to ground the region
      # some models show preference for more or less specificity
      # need to codify disambiguation - closest match?
      location:
        description: >-
          Target location of the radiation therapy region without modifiers.
          Remove modifiers like radiation technique, relative location, laterality,
          leaving just the target location.
        range: BodySite
      body_site:
        description: >-
          Specific body site or organ mentioned in the radiation therapy region.
          This should be a list of each discrete anatomical site.
          Do not use abbreviations or acronyms.
        range: BodySite
        required: false
        multivalued: true
      laterality:
        description: >-
          The laterality of the radiation therapy region, if this is mentioned.
          This should be "left", "right", "bilateral", or "na" if not specified.
          It is often specified as an abbreviation like 'lt', 'rt', (l), (r), r, l etc.
        range: Laterality
        required: false
```

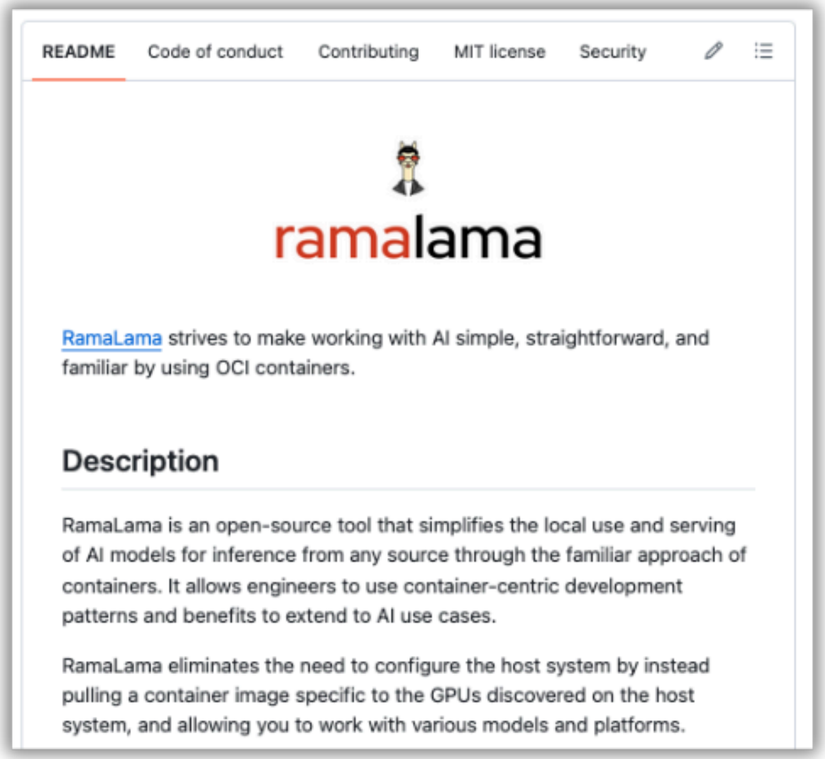
```yaml
  BodySite:
    is_a: OMOPHierarchy
    id_prefixes:
      - omop
    annotations:
      annotators: 'OMOP_OWL/ohdsi_test.db'
      # body site, body organ - in order of preference
      parent_id: omop:4190005, omop:4240671
  Laterality:
    is_a: OMOPEnum
    attributes:
      concept_name:
        range: LateralityEnum
    annotations:
      meaning: concept_id
enums:
  LateralityEnum:
    permissible_values:
      left:
        meaning: omop:45883143
      right:
        meaning: omop:45881626
      bilateral:
        meaning: omop:21498852
      na:
        description: The radiation therapy region does not specify laterality
```

Maridulu Budyari Gumal
Working together for good health and wellbeing

Proof of concept in Real Data - Radiotherapy Regions







Current Focus - Hosted, Integrated, Shared, Secured

SPIRES paper

OntoGPT Library

OMOP-links

Semantic SQL

Ramalama

linkML docs

Python Instructor

These slides

More detailed write-up

georgina.kennedy@unsw.edu.au

Maridulu Budyari Gumal
Working together for good health and wellbeing

More Reading & Contact