Few-shot prompt for “Hierarchical Pattern”

```
prompt = (
        "You will be provided with three components: USER_JSON, TEXT_CONTENT, and OUTPUT_JSON_FORMAT. "
        "1. **USER_JSON**: This contains subclasses and their associated data properties without values. "
        "2. **TEXT_CONTENT**: This contains the text content from which you need to extract information. "
        "3. **OUTPUT_JSON_FORMAT**: This specifies the format in which you should return your response. "
        "Your task is to read the USER_JSON and identify the individuals from the TEXT_CONTENT for the given classes and subclasses. "
        "For each individuals, extract the corresponding values for the data properties. "
        "Return the results formatted as specified in OUTPUT_JSON_FORMAT. "
        "Do not include any additional messages or content in your response."
        f"\n\nUSER_JSON: {hierarchical_pattern}"
        f"\nOUTPUT_JSON_FORMAT: {output_format}"
        f"\nTEXT_CONTENT: {docdata}"
    )
  hierarchical_pattern = {
    "classes": {
      "Class_Name": {
        "subclasses": {
          "subclasses_name":{
            "data_properties": [
              "data_property_01",
              " data_property_02"
            ]
          }
        }
      }
    }
  }
```

```
output_format = '''{
  "ParentClass": {
    "SubClass": [
    {
      "individual_name": "value",
      "dataproperty_01": "value",
      "dataproperty_02": "value"
    },
    {
      "individual_name": "value",
      "dataproperty_01": "value",
      "dataproperty_02": "value"
      "dataproperty_03": "value"
    }
    ]
  }
  }'''
```

Few-shot prompt for “Binary Pattern”

```
 prompt = (
   "You will be provided with three components: USER_JSON, TEXT_CONTENT, and OUTPUT_JSON_FORMAT. "
   "1. **USER_JSON**: This contains OBJECT_PROPERTY relations of the ontology with their DOMAINS and RANGES."
   "2. **TEXT_CONTENT**: This contains the text content from which you need to extract individuals for ontology. "
   "3. **OUTPUT_JSON_FORMAT**: This specifies the format in which you should return your response. "

   "Your task is to read the USER_JSON and identify the object properties with their domains and ranges. Then, identify the individuals those follows the given object properties from the TEXT_CONTENT."
```

```
 "Return the results formatted as specified in OUTPUT_JSON_FORMAT. "

 "Do not include any additional messages or content in your response."

 f"\n\nUSER_JSON: {binary_relations}"
 f"\nOUTPUT_JSON_FORMAT: {jsonExample}"
 f"\nTEXT_CONTENT: {docdata}"
 )
binary_relations = '''
 {
 "OntologyAxiom": [
   {
   "Binary_Relation": "Domain_class RELATIONSHIP Range_class "
   "DOMAIN": " Domain_class ",
   "RANGE": " Range_class ",
   "OBJECT_PROPERTY": " RELATIONSHIP "
   }
 ]
 }
 '''

jsonExample= '''
 {
 "ObjectPropertyAxiom": [
   {
   "INDIVIDUAL": ["individual_1", "individual_2"],
   "OBJECT_PROPERTY": "relationship",
   "DOMAIN": "domain_of_the_relationship",
   "RANGE": "range_of_the_relationship",
   "AXIOM": "individual_1 relationship individual_2",
   },
```

{

"INDIVIDUAL": ["individual_1", "individual_3"],

"OBJECT_PROPERTY": "relationship",

"DOMAIN": "class_type_of_indididual_1",

"RANGE": "class_type_of_indididual_3",

"AXIOM": "individual_1 relationship individual_3",

}

]

}

'''

Few-shot prompt for "Nary Pattern"

prompt = (

"You will be provided with three components: USER_JSON, TEXT_CONTENT, and OUTPUT_JSON_FORMAT. "

"1. **USER_JSON**: This is the JSON defining my ontology's structure, representing classes with n-ary relationships and their connected range classes."

"2. **TEXT_CONTENT**: This contains the text content from which you need to extract individuals for ontology follwing the relationships. "

"3. **OUTPUT_JSON_FORMAT**: This specifies the format in which you should return your response. "

"Your task: Based on the n-ary class and their associated range classes in USER_JSON, generate individuals that adhere to the given relationships from the TEXT_CONTENT"

"For the n-ary class, assign a primary key by shortening the class name and appending a number"

"Then, connect each primary key instance with depending individuals of its range classes using their relationships. "

"Finally, provide the response folwoing the OUTPUT_JSON_FORMAT"

"Do not include any additional messages or content in your response."

f"\n\nUSER_JSON: {nary_relations}"

f"\nOUTPUT_JSON_FORMAT: {jsonExample}"

f"\nTEXT_CONTENT: {docdata}"

```
)
n_ary_entity = '''
{
 "NaryPropertyAxiom": {
    "ontology structure": [
       "Crop hasGrowingProblemEvent GrowingProblemEvent"
       "GrowingProblemEvent hasGrowingProblem GrowingProblem",
       "GrowingProblemEvent hasSymptom Symptom",
       "GrowingProblemEvent hasCausalAgent CausalAgent",
       "GrowingProblemEvent hasControlMethod ControlMethod",
       "GrowingProblemEvent hasPreventionMethod PreventionMethod"
    ]
}
}
'''

jsonExample= '''
{
 "NaryPropertyAxiom": {
    "pattern01": [
       "<crop_intance> hasGrowingProblemEvent <growing_problem_event_id>"
       "<growing_problem_event_id> hasGrowingProblem <GrowingProblem_name>",
       "<growing_problem_event_id> hasSymptom <Symptom_description>",
       "<growing_problem_event_id> hasCausalAgent <CausalAgent_name>",
       "<growing_problem_event_id>hasControlMethod <ControlMethod_description>",
       "<growing_problem_event_id> hasPreventionMethod <PreventionMethod_description>",
    ]
    "pattern02": [
       "<crop_intance> hasGrowingProblemEvent <growing_problem_event_id>"
       "<growing_problem_event_id> hasGrowingProblem <GrowingProblem_name>",
       "<growing_problem_event_id> hasSymptom <Symptom_description>",
```

```
      "<growing_problem_event_id> hasCausalAgent <CausalAgent_name>",
      "<growing_problem_event_id>hasControlMethod <ControlMethod_description>",
      "<growing_problem_event_id> hasPreventionMethod <PreventionMethod_description>",
    ]
}
}
```