Exercise Sheet 5

Exercise 1

Exercise 1a) - Extract all element names from a BPMN file.

```
from pathlib import Path
import re
from typing import Set
from haystack import component

@component

class bpmn_element_extractor():
    @component.output_types(extracted_names=Set)
    def run(self, path):
        bpmn = Path(path).read_text(encoding='utf-8')
        pattern = re.compile(r'<bpmn:(startEvent|task|exclusiveGateway
                                |parallelGateway|endEvent)\b[^>]*\bname="([^"]+)"')
        matches = pattern.findall(bpmn)
        bpmn_names = set()
        for bpmn_element in matches:
            bpmn_names.add(bpmn_element[1])


        return {"extracted_named": bpmn_names}
```

Exercise 1b)

Example Datastructure:

```
class Tasks(BaseModel):
    cities: List[str]
```

output_validator = OutputValidator(pydantic_model=Tasks)

Exercise 1b) - Prompt

```
"""

Create a JSON object containing all BPMN "tasks" in this description of a process model: {{passage}}.

Only use information that is present in the passage. Follow this JSON schema, but only return the actual instances without any additional schema definition:

{{schema}}

Make sure your response is a dict and not a list.

{% if invalid_replies and error_message %}

  You already created the following output in a previous attempt: {{invalid_replies}}

  However, this doesn't comply with the format requirements from above and triggered this Python exception: {{error_message}}

  Correct the output and try again. Just return the corrected output without any extra explanations.

{% endif %}

"""
```

Exercise 1b) - Generator

```
from haystack_integrations.components.generators.ollama import OllamaGenerator

generator = OllamaGenerator(model="llama3.2:3b",
                            url = "http://localhost:11434",
                            timeout = 30*60,
                            generation_kwargs={
                              "num_ctx": 4096,
                              "temperature": 0,
                              })
```

Exercise 1c) – Comparison Component

```
from typing import Set
from haystack import component

@component
class SetComparison:
    @component.output_types(precision=float, recall=float)
    def run(self, set_ground_truth: Set, set_text_extract: Set):
        true_pos = set_ground_truth & set_text_extract
        false_pos = set_text_extract - true_pos
        false_neg = set_ground_truth - true_pos
        precision = len(true_pos)/(len(true_pos)+len(false_pos))
        recall = len(true_pos)/(len(true_pos)+len(false_neg))
        return {"precision": precision, "recall": recall}
```

Pipeline

```
from haystack import Pipeline
pipeline = Pipeline(max_runs_per_component=10)
#Add components to your pipeline
pipeline.add_component(instance=prompt_builder, name="prompt_builder")
pipeline.add_component(instance=chat_generator, name="llm")
pipeline.add_component(instance=output_validator, name="output_validator")
pipeline.add_component(instance=set_comparison, name="set_comparison")
pipeline.add_component(instance=bpmn_extract, name="bpmn_extract")
# Now, connect the components to each other
pipeline.connect("prompt_builder.prompt", "llm.messages")
pipeline.connect("llm.replies", "output_validator")
pipeline.connect("output_validator.valid_replies", "set_comparison.text_extract")
pipeline.connect("bpmn_extract", "set_comparison.set_ground_truth")
pipeline.connect("output_validator.invalid_replies", "prompt_builder.invalid_replies")
pipeline.connect("output_validator.error_message", "prompt_builder.error_message")
```

Exercise 2

2a-1. Example Document Store

```
from haystack.document_stores.in_memory import InMemoryDocumentStore

document_store = InMemoryDocumentStore()

from haystack import Document


docs = [Document(id="TaskMissing", content="Task {x} is missing in the
description of the process model. Please add the task at the correct
position.", meta={"Type": "Task", "Error": "Missing"}),

        Document(id="TaskHallucinated", content="Task {x} is hallucinated.
Please remove it from the description of the process model.", meta={"Type":
"Task", "Error": "Hallucinated"})]


document_store.write_documents(docs)
```

2a-2. Custom Retriever

```
from haystack import component
@component
class CustomRetrieve:
    def __init__(self, doc_store) -> None:
        self.doc_store = doc_store
    @component.output_types(prompt_type=str)
    def run(self, mistake_type: Tuple, name: str):
        documents = self.doc_store.filter_documents({"operator": "AND",
                                "conditions": [
                                    {"field": "Type", "operator": "==", "value": mistake_type[0]},
                                    {"field": "Error", "operator": "==", "value": mistake_type[1]}
                                     ]})
        prompt_out = documents[0].content.format(x=name)
        return {"prompt_type": prompt_out}
retriever = CustomRetrieve(doc_store=document_store)
```

2a-3. Example Prompt

```
from haystack.components.builders import PromptBuilder


template ="""
You are given a BPMN model and a corresponding descriptive text.
Given BPMN model with textual process description: {{ BPMN }}
Correct only the following mistake in the description: {{ mistake }}
"""


prompt_builder = PromptBuilder(template=template,
required_variables=["BPMN", "mistake"])
```

2a-4. Generator

```
from haystack_integrations.components.generators.ollama import
OllamaGenerator


generator = OllamaGenerator(model="llama3.1",
                            url = "http://localhost:11434",
                            timeout = 30*60,
                            generation_kwargs={
                              "num_ctx": 4096,
                              "temperature": 0,
                              })
```

2a-Pipeline

```
from haystack import Pipeline

rag_pipeline = Pipeline()
# Add components to your pipeline
rag_pipeline.add_component("retriever", retriever)
rag_pipeline.add_component("prompt_builder", prompt_builder)
rag_pipeline.add_component("llm", generator)

# Now, connect the components to each other
rag_pipeline.connect("retriever", "prompt_builder.mistake")
rag_pipeline.connect("prompt_builder.prompt", "llm.prompt")
```

Exercise 3

3a)

```
from haystack.dataclasses import ChatMessage


system_message = ChatMessage.from_system("You are an expert judge
on descriptions of BPMN models. You are given a descriptive text of
a BPMN model and have to judge it based on its readability by
giving it a score between 1 and 5.")
user_message = ChatMessage.from_user("""
Please judge following BPMN description based on its readability.
BPMN model description: {{ query }}
""")
messages = [system_message, user_message]
```

3a)

```
from haystack import Pipeline

pipeline = Pipeline()
# Add components to your pipeline
pipeline.add_component("prompt_builder", prompt_builder)
pipeline.add_component("llm", generator)

# Now, connect the components to each other
pipeline.connect("prompt_builder.prompt", "llm.messages")
```