好问题👍，**`settings.yaml` 是你系统的“大脑”**，外部调用者不用管实现，系统内部一切行为（模块选择、数据源、LLM 路由策略等）都靠它来配置。

我帮你梳理一下 **合理的配置内容分类**：

---

1. 1. Provider 配置（大语言模型 / Embedding 引擎）

```
providers:
  openai:
    type: app.core.providers.openai_provider:OpenAIProvider
    api_key_env: OPENAI_API_KEY
    model_default: gpt-4o-mini
  ollama:
    type: app.core.providers.ollama_provider:OllamaProvider
    base_url: http://localhost:11434
    model_default: llama3.1
```

👉 定义可用的 LLM 提供商，支持多家（OpenAI, Azure, Ollama, vLLM…），并设置默认模型。

---

2. 2. LLM 路由策略

```
llm_policy:
  default: { model: gpt-4o-mini, provider: openai }
  embedding_provider: openai
  routes:
    GraphConstruction:
      alias_resolve:
        - { model: gpt-4o, provider: openai, ctx: 16k, price: 0.5 }
    RetrievalAgent:
      query_expand:
        - { model: gpt-4o-mini, provider: openai, ctx: 32k, price: 0.15 }
    ReasoningAgent:
      plan:
        - { model: gpt-4o, provider: openai, ctx: 128k, price: 3.0 }
      synthesize:
        - { model: llama3.1-70b, provider: ollama, ctx: 32k, price: 0.3 }
    VerifierAgent:
      factcheck:
        - { model: gpt-4o-mini, provider: openai, ctx: 32k, price: 0.15 }
```

👉 定义不同模块/用途调用 LLM 的路由规则（谁用哪个模型，优先级、价格上限、上下文窗口等）。

---

3. 3. 模块实现选择

```
modules:
  graph_construction:
app.modules.graph_construction.flow:GraphConstructionFlow
  retrieval: app.modules.retrieval.flow:RetrievalAgentFlow
  reasoning: app.modules.reasoning.flow:ReasoningAgentFlow
  verification: app.modules.verification.flow:VerifierAgentFlow
```

👉 指定每个模块用哪个实现类，保证可插拔。
比如 RetrievalAgent 可以切换 BM25、FAISS、HybridFlow 等不同实现。

---

4. 4. Graph Construction 模块参数

```
graph_construction:
  dataset: hotpotqa                # 或 wiki / custom
  data_path: data/hotpotqa/hotpot_dev_distractor_v1.json
  impl:
app.modules.graph_construction.impl_networkx:GraphConstructionNetworkX
  impl_kwargs:
    root_dir: data/graph
```

👉 配置数据源（HotpotQA / Wiki），文件路径，以及底层图存储实现。

---

5. 5. Retrieval 模块参数

```
retrieval:
  bm25_index: data/hotpotqa/bm25_index.pkl
  faiss_index: data/hotpotqa/faiss.index
  docs_path: data/hotpotqa/docs.jsonl
  top_k: 20
```

👉 配置文档索引路径、top_k 值，是否启用 BM25/FAISS。

---

6. 6. Reasoning 模块参数

```
reasoning:
  max_hops: 3
  strategy: planner_synth          # 或 tree_of_thoughts / reAct
```

👉 定义推理策略（Planner-Synthesizer, ToT, ReAct 等），以及最大多跳步数。

---

7. 7. Verification 模块参数

```
verification:
  rules:
    require_evidence: true
    min_coverage: 0.8
  use_llm: true
```

👉 设置验证规则：是否必须覆盖 supporting facts、最低覆盖率、是否调用 LLM 辅助检查。

---

8. 8. 全局参数（可选）

```
logging:
  level: INFO
  trace: true
  save_dir: logs/

metrics:
  enable: true
  output: data/metrics.json
```

👉 控制日志等级、是否启用 trace、metrics 输出路径。

---

9. ✅ 总结

在 `settings.yaml` 里你能配置：

1. **LLM provider**（OpenAI/Ollama…）
2. **LLM 路由策略**（哪个模块用哪个模型）
3. **模块实现类**（可插拔）
4. **GraphConstruction 数据源与存储**
5. **Retrieval 索引配置**
6. **Reasoning 策略**
7. **Verification 规则**
8. **全局 logging/metrics**

这样 `run_system.py` 调用时，外部用户完全不需要关心内部细节，只传入一个问题，系统就会根据 settings.yaml 自动 orchestrate。

要不要我帮你直接给一份 **完整的 settings.yaml 示例**，涵盖 HotpotQA 的场景（GraphConstruction 从 HotpotQA 加载，Retrieval 用 BM25+FAISS，Reasoning 用 PlannerSynth，Verification 用 Rules+LLM）？