Natural Language Processing

Exercise – Summer term 2026

klara.gutekunst@uni-kassel.de

Organization

Studienleistung

Programming Project: **Forensic Linguist Agent**

- Develop a modular agent system for forensic authorship analysis.
- Focus on *idiolectal style*: how a text is written rather than what it is about.
- A coordinator agent selects and executes reusable linguistic analysis skills.
- Ensure a transparent and reproducible workflow with inspectable intermediate results and uncertainty.
- Evaluate the system on authorship datasets and compare agent-based workflows to simpler baselines.

Agentic Workflow

What is an agent?

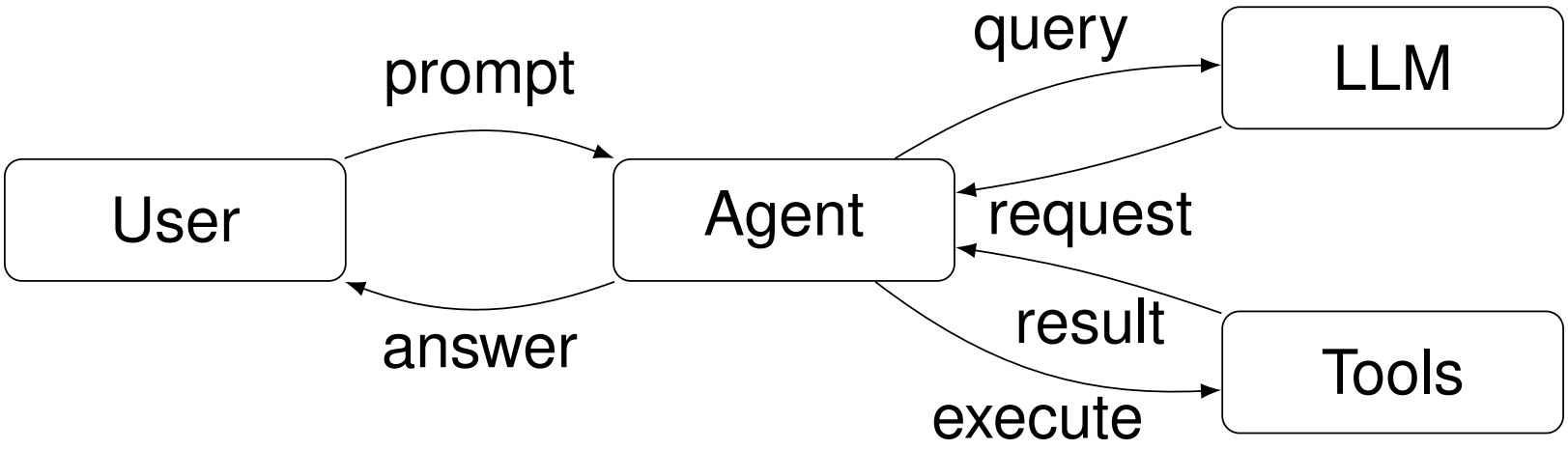


Figure: Simplified agentic workflow. A user prompt is processed by an agent that queries an LLM. The LLM may request tool calls, which are executed by the agent and returned as results. This loop continues until the agent produces a final answer.

Studienleistung

Details

Reminder of 28.04.2026

Your tasks include:

- Implement **tools** for linguistic analysis (e.g., linguistic feature extraction, stylometric analysis).
- **Persist** imediate **results** and **uncertainty** in a transparent way.
- Connect with a backend LLM.
- Identify risks and limitations of the agentic workflow and implement mitigation strategies.
- Evaluate the agent on benchmark datasets and compare to non-agentic baselines.
- Write an **4-page report** summarizing your approach, results, and insights in a two-column format including figures, tables, and references.

Studienleistung

Report

Checklist for the progress meeting:

- ❑ LaTeX ACM **template**
- ❑ 4-pages in two-column format including figures, tables, and references
- ❑ content [webis checklist]:
  - Title, author(s), abstract (5–10 sentences), keywords
  - Introduction
  - Related Work
  - Approach
  - Experiments
  - Conclusion
  - References
- ❑ Evaluate preliminary results and identify next steps.

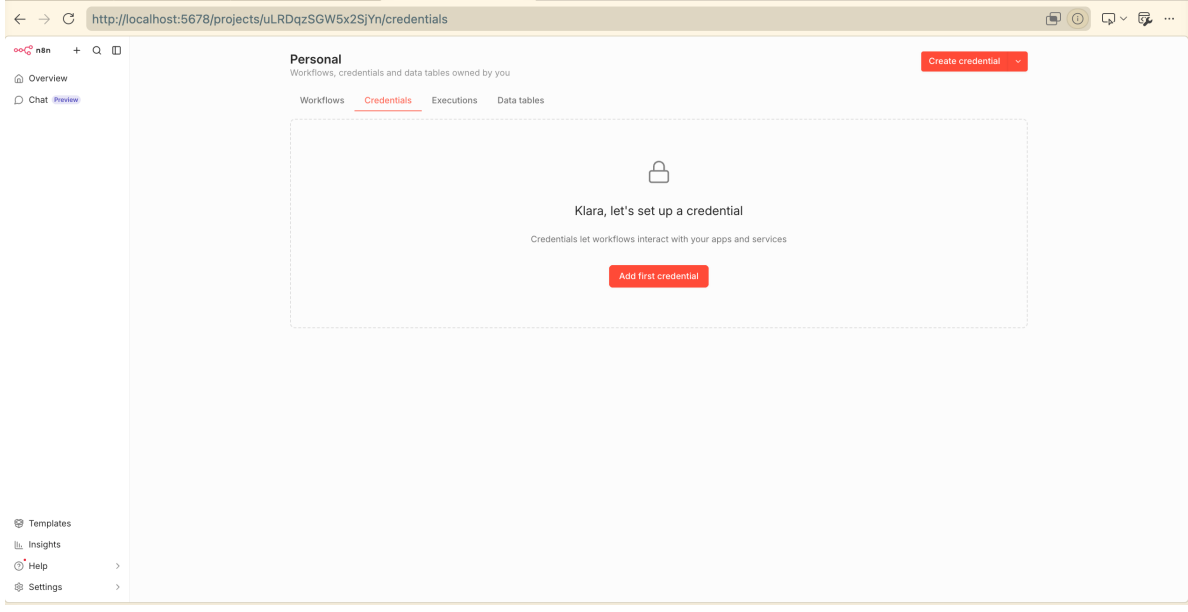
n8n

The following slides provide a brief overview of the n8n workflow automation tool, including setup instructions and the implementation of basic functionalities for the forensic linguist agent.

n8n Setup I

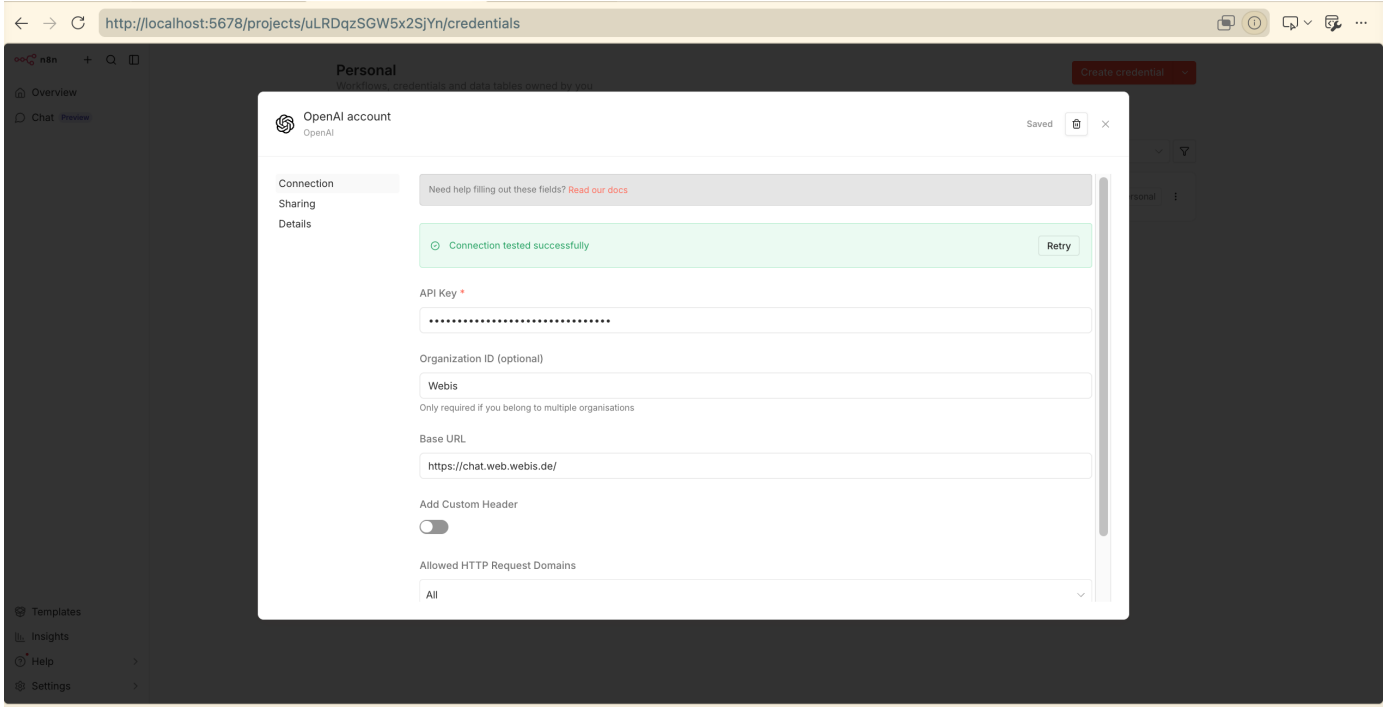
Start n8n

The *Docker Compose* configuration for n8n and Redis is available in the repository.
To start both services, run `docker compose up -d` from the repository root.
Once the containers are running, access the n8n editor at `http://localhost:5678`.



n8n Setup II

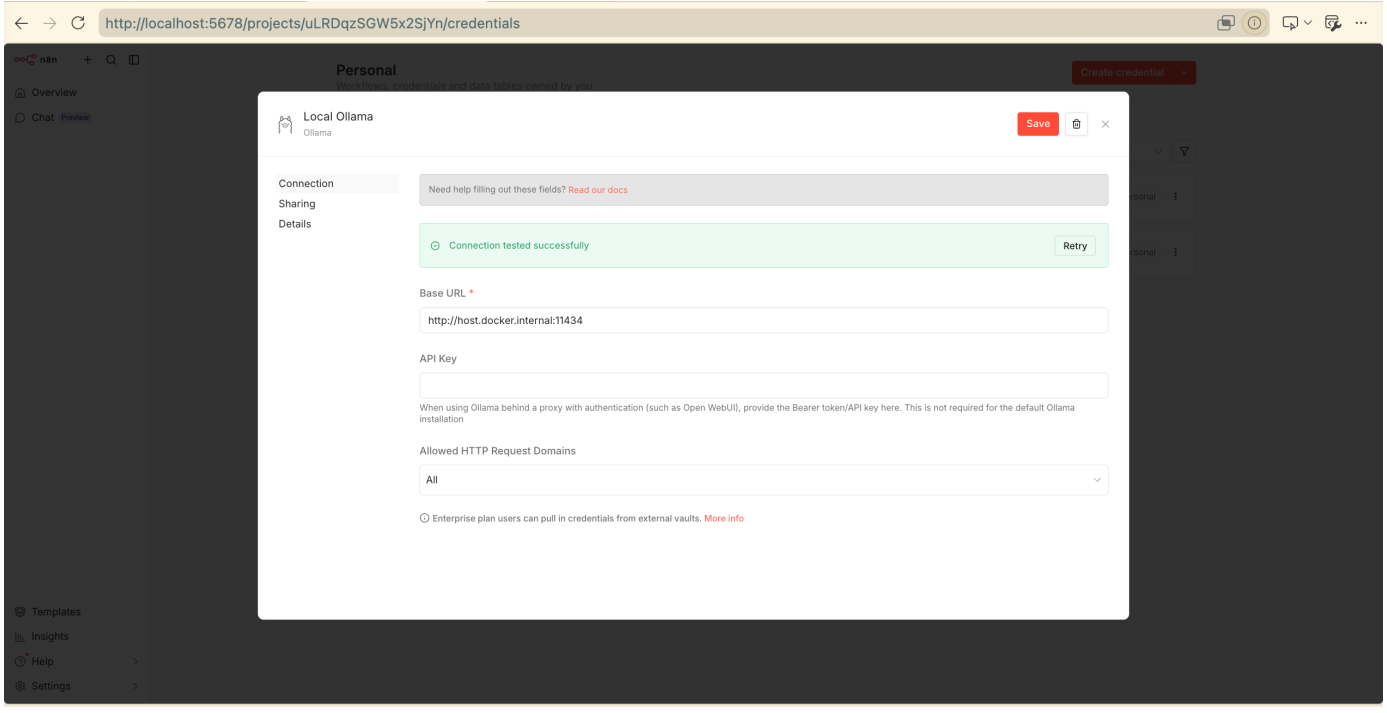
Configure the LLM Backend



Configure the Webis-hosted LLM using the OpenAI-compatible endpoint [n8n documentation].

n8n Setup III

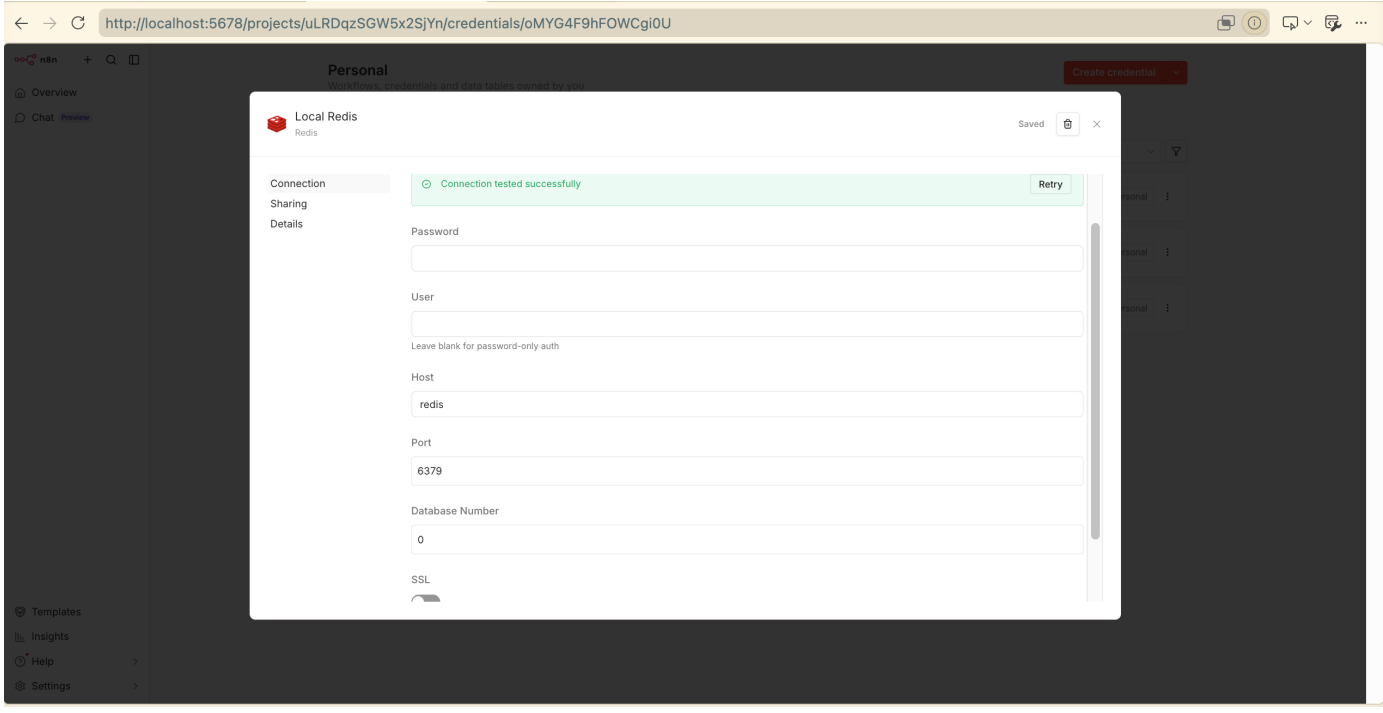
Configure the LLM Backend



Configure locally hosted LLM using the Ollama-compatible endpoint [n8n documentation].

n8n Setup IV

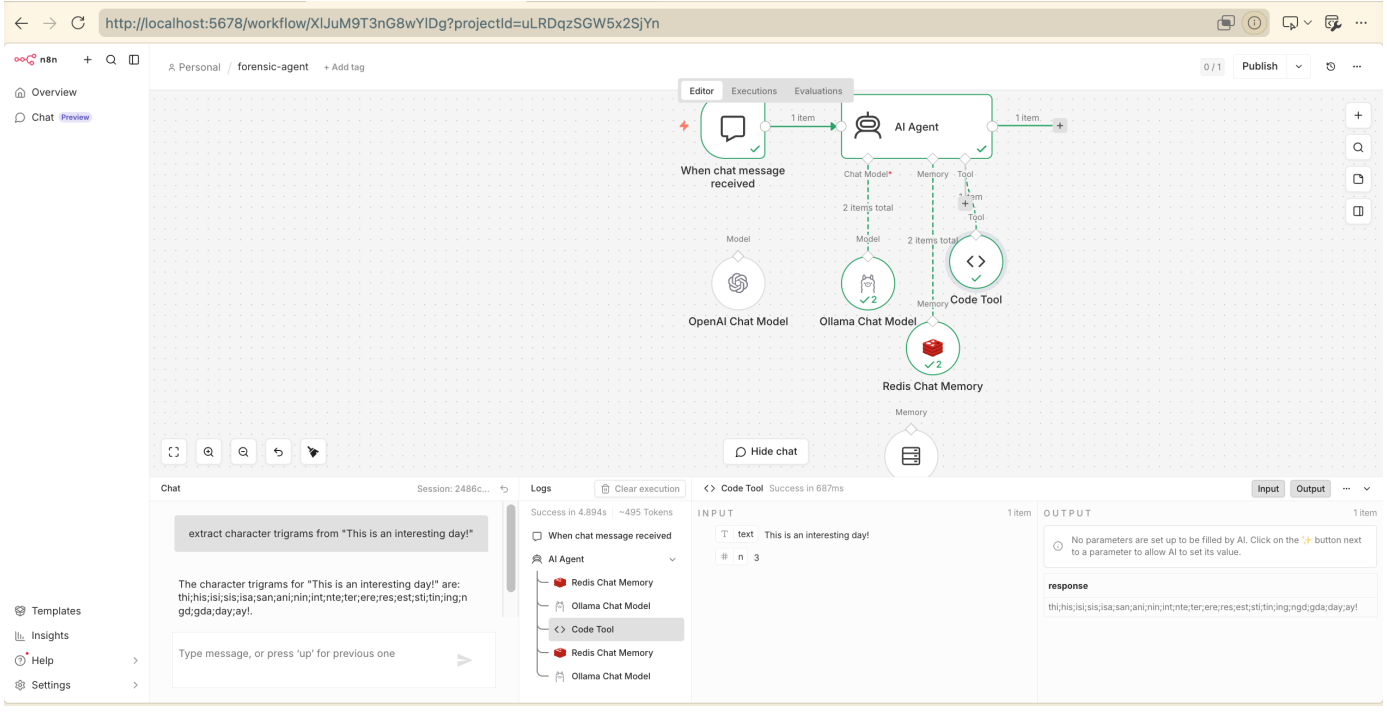
Configure the Redis Database



Configure the local Redis database. Unlike the setup described in the n8n documentation, a local Redis Docker container does not require authentication. Leave the username and password fields empty.

Persistent Storage

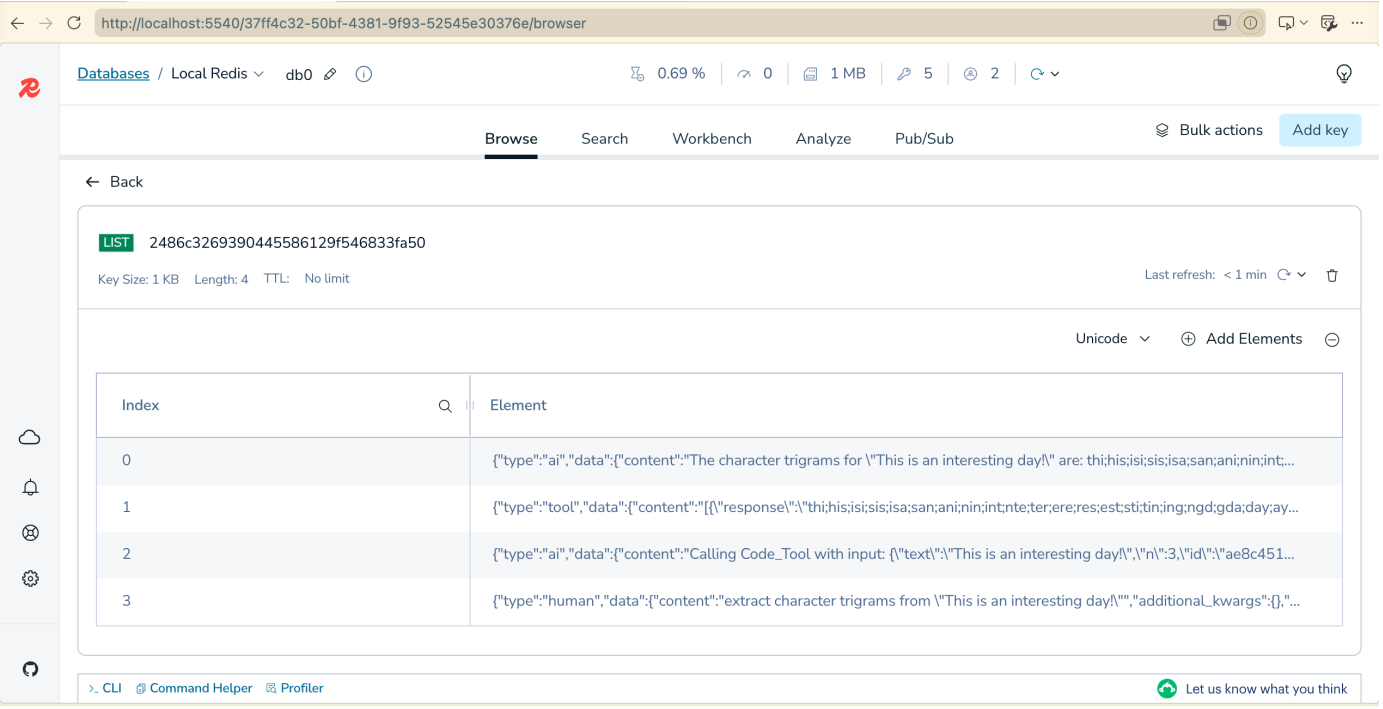
Redis Database



Working example of an n8n workflow using a local Ollama model (`granite4:3b`), a local Redis database, and a simple character n-gram extraction tool.

Persistent Storage

Redis Database



By mounting host directories (cf. repository), Redis data persists even after containers are stopped or recreated. The database can be explored using Redis Insight at `http://localhost:5540/`.

Tools

Overview

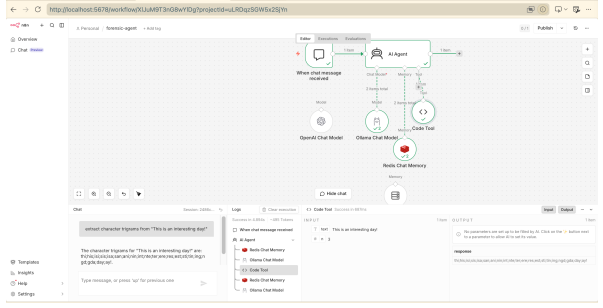
- custom functionality can be implemented
  - . . . directly in n8n using JavaScript Code nodes
  - . . . via external Python services (e.g., FastAPI) that are accessed via HTTP requests
  - . . .
- the following slides provide an example of a simple character n-gram extraction tool implemented in both ways and the corresponding system prompt configurations for the agent

AI Agent

Configurations (JavaScript Tool)

System Prompt when JavaScript tool is enabled:

```
You are Klara's local AI assistant.
For requests about n-grams:
1. Invoke the connected tool named ngram_tool and do not use memory.
2. Do not write "Calling ngram_tool".
3. Do not describe the tool call.
4. After the tool result is returned, copy the tool result exactly.
5. Never calculate n-grams yourself.

If the user does not provide n, use n = 3.
```



- Require Intermediate Steps: `true`
- n8n provides a built-in Code node that allows custom JavaScript logic to be defined and executed directly within workflows
- Python code node is more complicated

JavaScript Tool

Configurations (JavaScript Tool)

- Description: `Call this tool to obtain character n-grams separated by ";".`
- Language: `JavaScript`

Code:

```
const text = query.text ?? "";
const n = Number(query.n ?? 3);
const normalized = text.toLowerCase().replace(/\s+/g, "");
const grams = [];
for (let i = 0; i <= normalized.length - n; i++) {
grams.push(normalized.slice(i, i + n));
}
return grams.join(";");
```

Input Schema:

```
{
"type": "object",
"properties": {
   "text": {
     "type": "string",
     "description": "Text to split into n-grams"
   },
   "n": {
     "type": "integer",
     "description": "Size of n-gram character sequences",
     "minimum": 2,
     "default": 3
   }
},
"required": ["text"]
}
```

AI Agent

Configurations (External Tool)

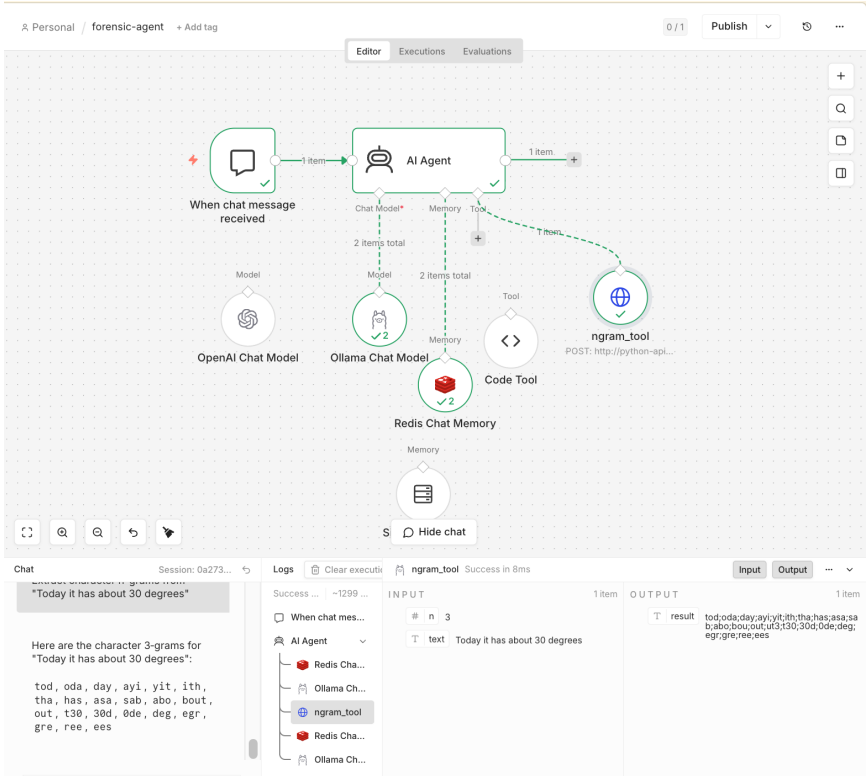
System Prompt when FastAPI tool is enabled:

```
You are Klara's local AI assistant.

Use the connected Redis memory for normal conversation and context retention.

For requests involving character n-grams:
1. Always invoke the connected HTTP request tool named 'ngram_tool'.
2. Never generate or calculate n-grams yourself.
3. Do not use Redis memory to answer n-gram requests.
4. Extract the input text from the user's request and pass it to the tool parameter 'text'.
5. Extract the n-gram length from the user's request and pass it to the tool parameter 'n'.
6. If the user does not specify 'n', use 'n = 3'.
7. Do not mention tool calls, HTTP requests, APIs, or internal reasoning.
8. Do not write phrases such as "Calling ngram_tool" or "Using a tool".
9. After the tool returns a result, return only the tool result and nothing else.
```

- Require Intermediate Steps: `true`

External Tool

Python Code Tool via FastAPI Tool



Uvicorn (i.e., web server) runs the FastAPI application.

```python
from fastapi import FastAPI
from pydantic import BaseModel

app = FastAPI()
class NgramReques
   text: str = ""
   n: int = 3

@app.post("/ngrams")
def ngrams(request: NgramRequest):
   normalized = "".join(request.text.lower().split())
   grams = [
     normalized[i:i + request.n]
     for i in range(len(normalized) - request.n + 1)
   ]
   return {"result": ";".join(grams)}
```

Code for the FastAPI application.
API is exposed using docker-compose.yml.

Ollama Chat Model

Configurations

- Sampling Temperature: `0,1`