Adaptive HMI Ontology Generator: Installation and Usage Guide

This guide will help you set up and use the Adaptive HMI Ontology Generator with the new file structure that separates user stories and competency questions.

Installation

Step 1: Set up the directory structure

Make sure you have the following directory structure:

```
adaptive_hmi_ontology/
├── main.py
├── config.py
├── data/
│   ├── patterns.csv
│   ├── competency_questions.csv
│   ├── user_stories.csv
│   └── output/
└── src/
    ├── __init__.py
    ├── api_client.py
    ├── ontogenia.py
    ├── improved_ontology_merger.py
```

Step 2: Install dependencies

```
bash

pip install rdflib pandas requests
```

Files to Update

You need to create or update the following files:

1. config.py

Replace your existing config.py with the version provided in the "Complete config.py" artifact.

2. main.py

Replace your existing main.py with the version provided in the "Complete main.py" artifact.

3. patterns.csv

Replace your existing patterns.csv with the version provided in the "Full patterns.csv" artifact.

4. competency_questions.csv

Create a new file at data/competency_questions.csv with the content provided in the "Full competency_questions.csv" artifact.

5. user_stories.csv

Create a new file at data/user_stories.csv with the content provided in the "Full user_stories.csv" artifact.

6. improved_ontology_merger.py

Create a new file at src/improved_ontology_merger.py with the content provided in the "src/improved_ontology_merger.py" artifact.

Usage

Once you have set up all the files, you can run the ontology generator:

```bash
python main.py
```

Command-line Options

The main.py script supports several command-line options:

- `--cq-file`: Path to a custom competency questions CSV file
- `--user-stories-file`: Path to a custom user stories CSV file
- `--patterns-file`: Path to a custom patterns CSV file
- `--output-dir`: Directory to save the output files
- `--num-questions`: Number of questions to process (default: all)
- `--use-situation-event`: Emphasize situation-event modeling in the prompts

Example:

```bash
python main.py --num-questions 10 --use-situation-event
```

How the System Works

The system now uses two separate CSV files:

1. **user_stories.csv**: Contains detailed user stories with unique IDs
2. **competency_questions.csv**: Contains competency questions linked to user stories via StoryID

When the system runs, it:

1. Loads all user stories into a dictionary
2. Loads all competency questions and maps them to their corresponding user stories
3. Processes each competency question with its story context
4. Generates an ontology module for each question
5. Merges the modules into a comprehensive ontology
6. Detects and resolves duplicate entities during merging

Adding New Content

To add a new user story:

1. Add a new row to user_stories.csv with a unique StoryID and the story text

To add new competency questions:

1. Add new rows to competency_questions.csv with unique CQIDs, the question text, and the StoryID of the related story

Troubleshooting

If you encounter issues:

1. Check the log file (ontology_generation_*.log) for error messages
2. Ensure all CSV files have the correct format and column names
3. Make sure the StoryIDs in competency_questions.csv match those in user_stories.csv
4. Verify that the Claude API URL in config.py is correct