Experimental Data Report Generator

Overview

The `src/utils/generate_experimental_data_report.py` script generates a comprehensive markdown report containing all experimental datasets used in the quantum pattern analysis research. This report provides complete datasets for reproducibility and further analysis.

Purpose

This script addresses the need to provide complete experimental data for research papers, including:

- **Framework Concept Extractions**: Complete datasets from Classiq, PennyLane, and Qiskit
- **Pattern Matching Results**: Full analysis results from pattern matching
- **Pattern Atlas Data**: Complete set of quantum patterns from PlanQK Pattern Atlas
- **Statistical Analysis**: All generated tables and metrics

Usage

Command Line

```
# Using just (recommended)
just experimental-data
```

```
# Direct execution
python -m src.utils.generate_experimental_data_report
```

Output

The script generates `docs/experimental_data.md` containing all experimental datasets.

Generated Content

1. Framework Concept Tables

- **Classiq Quantum Concepts**: Complete dataset from `data/classiq_quantum_concepts.csv` (comma-delimited)
- **PennyLane Quantum Concepts**: Complete dataset from `data/pennylane_quantum_concepts.csv` (comma-delimited)
- **Qiskit Quantum Concepts**: Complete dataset from `data/qiskit_quantum_concepts.csv` (semicolon-delimited)
- **Row Numbers**: All tables include sequential row numbers for easy reference

2. Pattern Analysis Results

- **Top Matched Concepts**: From `data/report/top_matched_concepts.csv`
- **Match Type Analysis**: From `data/report/match_type_counts.csv`
- **Framework Analysis**: From `data/report/matches_by_framework.csv`
- **Pattern Frequency**: From `data/report/patterns_by_match_count.csv`

3. Pattern Atlas Data

- **Complete Pattern List**: From `data/quantum_patterns.json`
- **Pattern Metadata**: Names, aliases, intents, and descriptions
- **Source Reference**: PlanQK Pattern Atlas citation

File Structure

```
docs/
└── experimental_data.md          # Generated experimental data report

data/
├── classiq_quantum_concepts.csv     # Classiq concepts (input)
├── pennylane_quantum_concepts.csv   # PennyLane concepts (input)
├── qiskit_quantum_concepts.csv      # Qiskit concepts (input)
├── quantum_patterns.json            # Pattern Atlas data (input)
└── report/                          # Analysis results (input)
    ├── top_matched_concepts.csv
    ├── match_type_counts.csv
    ├── matches_by_framework.csv
    └── patterns_by_match_count.csv
```

Features

Error Handling

- Graceful handling of missing files
- Clear error messages for debugging
- Continues processing even if some files are missing

Data Validation

- Checks file existence before processing
- Validates CSV structure
- Handles empty datasets appropriately
- **Delimiter Detection**: Automatically uses semicolon (;) for Qiskit files and comma (,) for others
- **Row Numbering**: Adds sequential row numbers to all tables for easy reference

Formatting

- Clean markdown tables
- Proper section organization
- Academic citation format
- Professional presentation

Integration

With Research Workflow

1. Run the main analysis: `just identify-concepts`
2. Generate pattern matching: `just run_main`
3. Create final report: `just report`
4. Generate experimental data: `just experimental-data`

With Paper Writing

The generated markdown file can be: - Converted to LaTeX for academic papers - Used directly in documentation - Referenced for data citations - Shared for reproducibility

Customization

Adding New Data Sources

To include additional datasets, modify the `ExperimentalDataReportGenerator` class:

```
# Add new file paths
self.new_data_file = self.data_dir / "new_data.csv"

# Add new table generation method
def _generate_new_data_table(self):
    # Implementation here
    pass
```

Modifying Output Format

The script uses pandas `to_markdown()` for table formatting. To change the format:

```
# In table generation methods
table_md = df.to_markdown(index=False, tablefmt="grid")
```

Dependencies

- `pandas`: Data manipulation and table formatting
- `pathlib`: File path handling
- `json`: JSON data processing
- `src.conf.config`: Project configuration

Testing

The script includes comprehensive tests in `tests/test_generate_experimental_data_report.py`:

- **16 test cases** covering all functionality
- **Error handling** for missing files and data issues
- **Mock testing** for isolated unit testing
- **Integration testing** for end-to-end functionality

Run tests with:

```
just test-file tests/test_generate_experimental_data_report.py
```

Benefits

1. **Reproducibility**: Complete datasets for research replication
2. **Transparency**: Full experimental data available
3. **Academic Standards**: Proper citations and references
4. **Automation**: No manual data compilation required
5. **Consistency**: Standardized format across all datasets
6. **Accessibility**: Easy to share and reference

Example Output

The generated markdown file includes:

```
# Experimental Data

## Overview
This document contains the complete experimental datasets...

## Classiq Quantum Concepts
The complete dataset of quantum concepts extracted from the Classiq framework.

**File**: `classiq_quantum_concepts.csv`
**Total Concepts**: 150

| name | summary |
|------|---------|
| classiq.circuit | Quantum circuit implementation |
| ... | ... |

## References
@online{PlanQK_QuantumPatterns_2024,
  author       = {{PlanQK}},
  title        = {Quantum Computing Patterns},
  year         = {2025},
  url          = {https://patternatlas.planqk.de/...},
  urldate      = {2025-09-28}
}
```

This comprehensive experimental data report ensures that all research data is properly documented and accessible for reproducibility and further analysis.