Module Generator and Sunflower

- ***Making Graph Databases Speak Python and Reveal Patterns***

Authors: Dan Hales & Yiyi Luo

Introduction

Graph databases have revolutionized how we store and analyze connected data, enabling powerful insights into relationships that traditional databases struggle to reveal. However, working with graph databases typically requires specialized knowledge of query languages like Cypher, creating a barrier for many developers and data analysts. This paper introduces two complementary tools designed to overcome these challenges: Module Generator and Sunflower.

Module Generator transforms the structure of Neo4j graph databases into intuitive Python code, eliminating the need to write complex queries. By automatically generating Python functions that map directly to database operations, it allows developers to interact with graph data using familiar programming patterns. Sunflower builds upon this foundation, offering a "pattern-first" approach to graph exploration. It automatically identifies meaningful relationship patterns in the database and presents them through interactive visualizations, making complex network insights accessible to non-technical users.

Together, these tools democratize access to graph data, enabling anyone with basic Python knowledge to extract valuable insights from interconnected information. Using a movie database featuring films like "The Matrix" as our example, we'll demonstrate how these tools simplify the process of querying and analyzing graph relationships. By the end of this paper, you'll understand how Module Generator and Sunflower can help your organization leverage the full potential of graph databases without requiring specialized expertise.

A noteworthy aspect of this project is the innovative development approach behind Sunflower. While Module Generator was traditionally developed by our team through conventional programming methods, Sunflower represents a pioneering experiment in AI-human collaboration. To create Sunflower, we uploaded the Module Generator codebase to Claude AI along with a conceptual prompt describing Sunflower's intended purpose and functionality. The AI then generated the entire application—all code files, structure, and implementation details—based on this foundation. This approach fundamentally transforms the development workflow, where the human role shifts from writing code to providing vision and validation. Sunflower demonstrates how complex software can emerge from the combination of existing code foundations, clear conceptual direction, and AI's ability to generate coherent, functional implementations. This project offers not only valuable tools for graph database interaction but also insights into the future of software development through human-AI partnership.

Contents

I. What is Graph Database?

Basic Concept

A graph database stores data in terms of nodes (entities like movies or people) and relationships (connections between them). Unlike traditional databases with separate tables, graph databases directly connect related information. This makes it ideal for finding patterns and exploring relationships, like seeing which actors worked with Lana across multiple films without complex queries.

Traditional Database

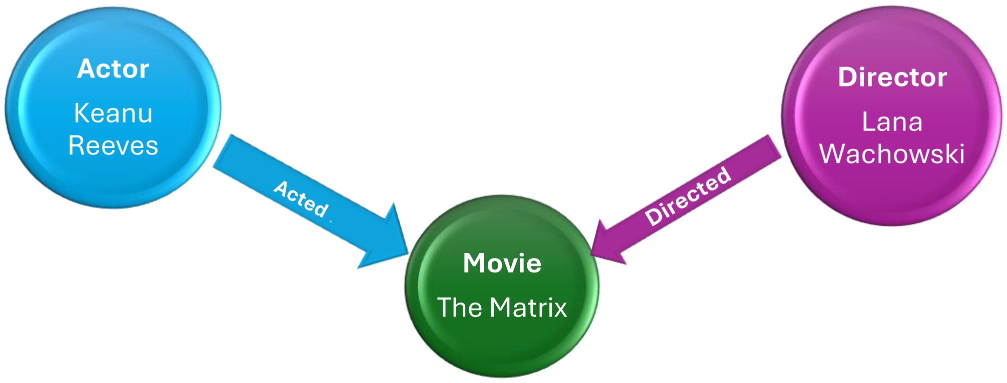
Movies Table

Movie_id	**title**	**released**	**runtime**	**budget**
1	The Matrix	1999	136	$63,000,000
2	The Matrix Reloaded	2003	138	$150,000,000
3	The Matrix Revolutions	2003	129	$150,000,000

Persons Table

Person_id	**name**	**born**	**gender**
101	Lana Wachowski	1965-06-21	Female
102	Keanu Reeves	1964-09-02	Male
103	Carrie-Anne Moss	1967-08-21	Female

Graph Database (Neo4j)



Example Query Comparison

Find all actors who worked with Lana Wachowski on The Matrix:

SQL (Traditional Database):

```
SELECT p.name, p.born
FROM People p
JOIN Actor_Movie am ON p.person_id = am.actor_id
JOIN Movies m ON am.movie_id = m.movie_id
JOIN Director_Movie dm ON m.movie_id = dm.movie_id
JOIN People dir ON dm.director_id = dir.person_id
WHERE m.title = 'The Matrix'
AND dir.name = ‘Lana Wachowski';
```

Cypher (Graph DB):

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(d:Person)
WHERE m.title = 'The Matrix' AND d.name = 'Lana Wachowski'
RETURN p.name, p.born;
```

II. What is Module Generator?

Basic Concept

Module Generator automatically transforms a graph database's structure into easy-to-use Python code. It discovers what's in your database (like movies, actors, directors) and creates functions that let developers work with this data using familiar Python syntax instead of specialized database queries. It essentially creates a Python translator for your graph database, making it accessible to anyone who knows basic Python.

Module Generator Workflow

1. Neo4j Movie Database (Starting Point)

- Contains data about The Matrix:
    - Movie: "The Matrix" (1999)
    - Director: Lana Wachowski
    - Actors: Keanu Reeves, Carrie-Anne Moss

  - Relationships: Wachowski DIRECTED The Matrix, Reeves/Moss ACTED_IN The Matrix

2. Connect to Database

- Connect to the Neo4j database containing the movie data
- Provide credentials (URI, username, password)

3. Discover What's in Database

- Module Generator examines the database structure
- Finds movie data with properties like "title" and "released"
- Finds person data with properties like "name" and "born"
- Discovers relationships: DIRECTED, ACTED_IN

4. Generate Python Code

- Creates functions to interact with the database:

```
def movie(title=None, released=None, **props):
    """Find movies matching the given properties"""
    # Auto-generated search logic
    # Returns matching movies

def find_actors_in_movie(movie_title):
    """Find actors who appeared in a specific movie"""
    # Auto-generated relationship query
    # Returns list of actors
```

5. Ready-to-Use Python Module

- Developers can now use the generated code:

```
import moviegraph

# Find The Matrix movie
The Matrix = moviegraph.movie(title="The Matrix")


# Find all actors in The Matrix
The Matrix_actors = moviegraph.find_actors_in_movie("The
Matrix")
# Returns: [{"name": "Keanu Reeves"}, {"name": "Carrie-Anne
Moss"}, ...]
```

III. What is Sunflower?

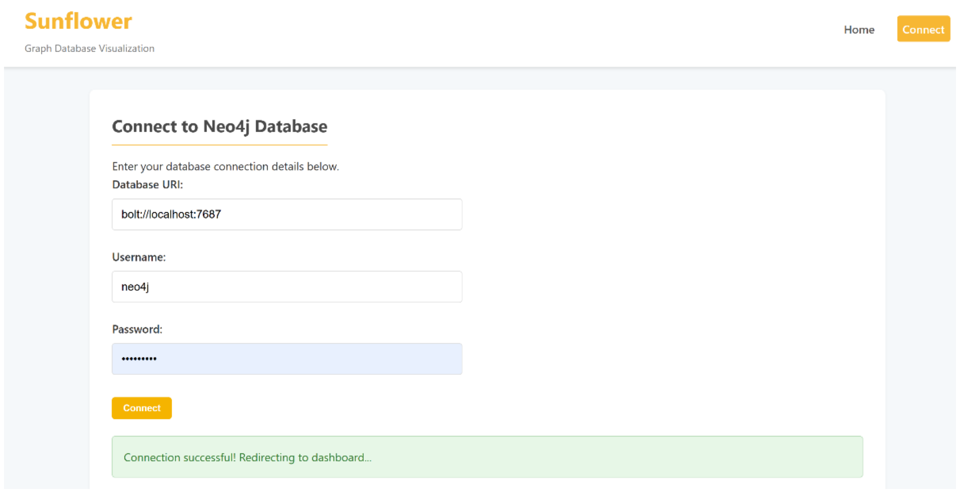
Basic Concept

Sunflower automatically identifies and visualizes meaningful patterns in graph data. It finds recurring structures in your database (like directors who repeatedly work with the same actors) and presents them in an interactive visual format. Sunflower allows non-technical users to discover insights without writing complex queries, making the "hidden wisdom" in connected data visible and accessible to everyone.

Sunflower Workflow

1. Connect to Database

- Connect to the Neo4j database containing The Matrix and other movie data



Connect to Neo4j Database

Enter your database connection details below.

Database URI: bolt://localhost:7687

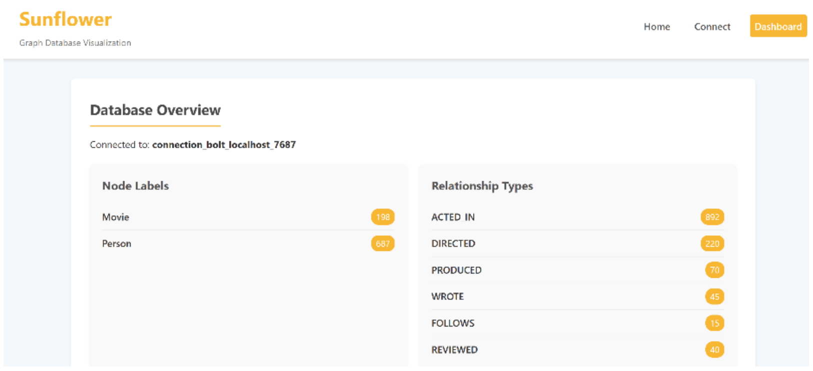
Username: neo4j

Password: •••••••••

Connection successful! Redirecting to dashboard...

2. Generate Interface

- Use Module Generator to create the Python interface



Database Overview

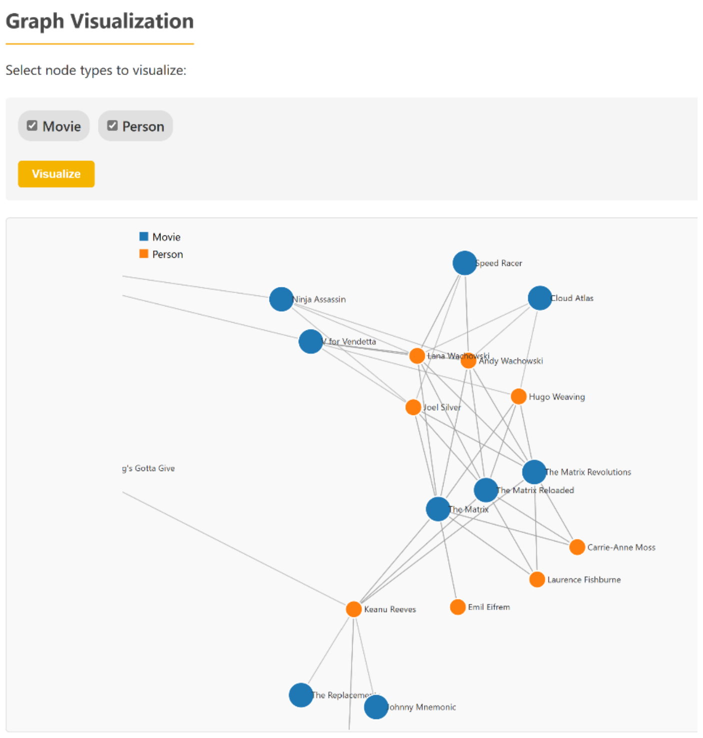
Connected to: **connection_bolt_localhost_7687**

Node Labels	
Movie	198
Person	687

Relationship Types	
ACTED_IN	892
DIRECTED	220
PRODUCED	70
WROTE	45
FOLLOWS	15
REVIEWED	40

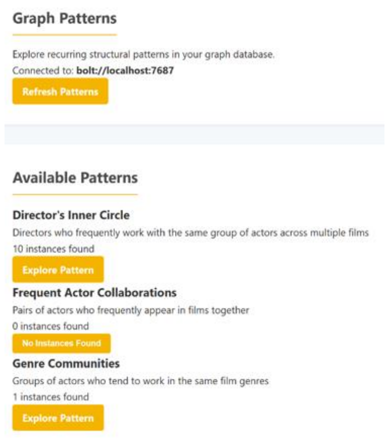
3. Find Patterns

- Sunflower automatically searches for patterns like:
  - Directors who frequently work with the same actors
  - Actors who frequently work together
  - Movies with similar characteristics



4. Pattern Catalog

- Shows discovered patterns with examples:
  - Director's Inner Circle: "Lana Wachowski works repeatedly with certain actors"


Graph Patterns

Explore recurring structural patterns in your graph database.
Connected to: **bolt://localhost:7687**

Refresh Patterns

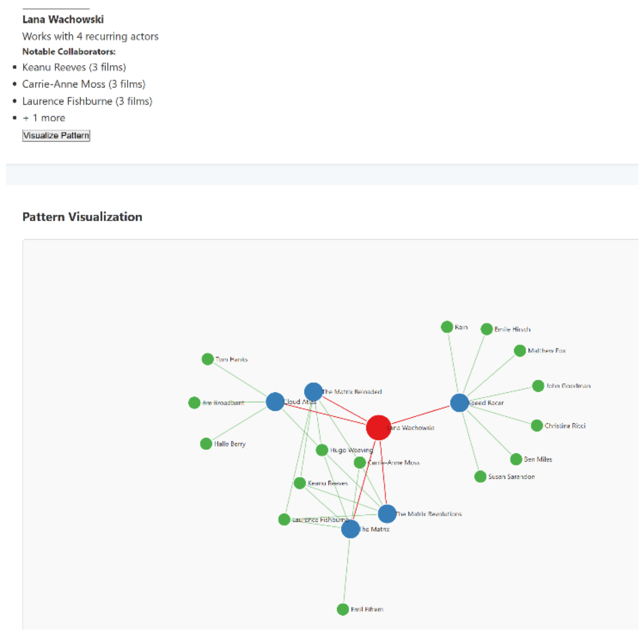
Available Patterns

Director's Inner Circle
Directors who frequently work with the same group of actors across multiple films
10 instances found
Explore Pattern

Frequent Actor Collaborations
Pairs of actors who frequently appear in films together
0 instances found
No Instances Found

Genre Communities
Groups of actors who tend to work in the same film genres
1 instances found
Explore Pattern

5. **Choose Pattern**
    - User selects "Director's Inner Circle" to explore Lana Wachowski's repeated collaborators



Lana Wachowski

Works with 4 recurring actors

Notable Collaborators:

- Keanu Reeves (3 films)
- Carrie-Anne Moss (3 films)
- Laurence Fishburne (3 films)
- + 1 more

Visualize Pattern

Pattern Visualization

6. **Interactive Visualization**
    - Shows Lana Wachowski at the center
    - Connected to her films: The Matrix, Cloud Atlas, Jupiter Ascending
    - Shows actors who appeared in multiple Wachowski films
    - Reveals that Lana Wachowski frequently works with actors Keanu Reeves

Key Files and Structure of Sunflower

Here's a summary of Sunflower's main components based on the files you've shared:

Core Application Files

1. **app.py**
    - The main Flask application that handles routing and serves as the entry point
    - Integrates with Module Generator to create Python interfaces for Neo4j databases
    - Sets up routes for database connection, dashboard, patterns exploration, and API endpoints

    - Manages session state to persist database connections and pattern information
2. **db_manager.py**
    - Manages database connections and generated modules
    - Handles loading/saving connection information
    - Provides methods to connect to Neo4j and generate Python modules
    - Extracts database schema information from generated modules
3. **pattern_detector.py**
    - Core class that detects patterns in Neo4j databases
    - Implements methods to find predefined patterns like Director's Inner Circle, Actor Collaborations, and Genre Communities
    - Provides visualization data for pattern instances
    - Includes deduplication logic to prevent duplicate pattern instances
4. **pattern_library.py**
    - Defines the pattern types that Sunflower can detect
    - Contains Cypher queries that identify each pattern in the database
    - Provides metadata (name, description) for each pattern type

Frontend Templates

1. **index.html** - Welcome page with introduction and features
2. **connect.html** - Connection form for Neo4j database credentials
3. **dashboard.html** - Overview of database structure and general visualization
4. **patterns.html** - Catalog of detected patterns in the database
5. **pattern_detail.html** - Detailed view of a specific pattern with visualization

JavaScript Files

1. **connect.js** - Handles form submission for database connection
2. **dashboard.js** - Creates interactive graph visualizations of database data
3. **pattern-visualization.js** - Renders pattern-specific visualizations using D3.js

Configuration Files

1. **connections.json** - Stores saved database connection information (with masked passwords)
2. **styles.css** - Contains all styling for the application

Architecture

1. **Database Connection Stage:**
   - User provides Neo4j credentials
   - Module Generator creates a Python interface for the database
   - Connection is saved for future use
2. **Pattern Detection Stage**:
   - Patterns are discovered using predefined Cypher queries
   - Results are processed and deduplicated
   - Pattern instances are cached in the session
3. **Visualization Stage**:
   - Users select patterns to explore from the catalog
   - Pattern instances are visualized using D3.js force-directed graphs
   - Different node types (directors, actors, movies, genres) are color-coded
   - Interactive elements allow exploration of the graph

This architecture demonstrates a clean separation of concerns with database connectivity, pattern detection logic, and visualization rendering handled by distinct components. The application follows a "pattern-first" approach to graph exploration, making complex patterns in Neo4j databases accessible without requiring knowledge of Cypher query language.

IV. Summary

1. The Problem Module Generator Solves

- Without Module Generator, you'd need to write complex Cypher queries:
- `MATCH (m:Movie)WHERE m.title = "The Matrix"RETURN m`
- With Module Generator, you write simple Python:
- `The Matrix = moviegraph.movie(title="The Matrix")`

2. The Problem Sunflower Solves

- Without Sunflower, discovering patterns requires writing complex analyses
- With Sunflower, patterns like "Lana Wachowski's collaborators" are automatically found and visualized

3. Example Insights from The Matrix

- Discover that Wachowski works with many of the same crew and actors across films

  - Discover that Keanu Reeves and Carrie-Anne Moss went on to collaborate again in "The Matrix Revolutions"
  - Find connections between The Matrix and other successful blockbusters