**Mục lục**

[Chương 1: Mô tả bài toán 2](#_Toc187142813)

[1.1 Giới thiệu Cytoscape 2](#_Toc187142814)

[1.2 Mục tiêu nghiên cứu 2](#_Toc187142815)

[Chương 2: Thiết kế và cài đặt hệ thống 4](#_Toc187142816)

[2.1 Sơ đồ kiến trúc 4](#_Toc187142817)

[2.2 Thiết kế các thành phần 4](#_Toc187142818)

[2.3 Cài đặt 8](#_Toc187142819)

[Chương 3: Các kĩ thuật OOP 10](#_Toc187142820)

[3.1 Tính trừu tượng 10](#_Toc187142821)

[3.2 Tính kế thừa 10](#_Toc187142822)

[3.3 Tính đóng gói 11](#_Toc187142823)

[3.4 Tính đa hình 11](#_Toc187142824)

Chương 1: Mô tả bài toán

* 1. Giới thiệu Cytoscape

Cytoscape là nền tảng phần mềm mã nguồn mở, được sử dụng rộng rãi trong tin sinh học và phân tích mạng lưới. Cytoscape chuyên biệt trong việc trực quan hóa và phân tích các mạng lưới sinh học phức tạp, đặc biệt là các mạng lưới tương tác phân tử. Nền tảng này cung cấp khả năng mạnh mẽ trong việc tích hợp các mạng lưới với nhiều loại dữ liệu khác nhau, bao gồm chú thích, dữ liệu biểu hiện gen và các thông tin trạng thái khác, từ đó hỗ trợ sâu rộng cho việc nghiên cứu và khám phá tri thức sinh học.

* 1. Mục tiêu nghiên cứu

Mục tiêu của nghiên cứu này là phát triển một ứng dụng trên nền tảng Cytoscape nhằm khai thác tiềm năng của Mạng nơ-ron đồ thị (Graph Neural Networks - GNNs) trong việc phân tích mạng lưới sinh học. Cụ thể, ứng dụng sẽ thực hiện các tác vụ chính sau:

* **Nhập mạng lưới (Network Import):** Ứng dụng cho phép nhập các mạng lưới/đồ thị, bao gồm cả trường hợp có hoặc không có dữ liệu thuộc tính (attribute data) đi kèm cho các nút (node) và cạnh (edge).
* **Tích hợp mô hình GNN:** Ứng dụng tích hợp các mô hình GNNs phù hợp để thực hiện các phân tích chuyên sâu trên mạng lưới.
* **Tính toán vector nhúng nút (Node Embedding):** Sử dụng các mô hình GNNs đã tích hợp để tính toán vector nhúng (embedding vector), hay còn gọi là vector đặc trưng ẩn, cho từng nút trong mạng lưới. Các vector này mã hóa thông tin cấu trúc và thuộc tính của các nút trong không gian vector, phục vụ cho các tác vụ phân tích tiếp theo.
* **Dự đoán nhãn nút (Node Classification):** Dựa trên các vector nhúng nút, ứng dụng sẽ triển khai các thuật toán học máy để dự đoán nhãn cho các nút trong mạng lưới. Tác vụ này có ý nghĩa quan trọng trong nhiều bài toán thực tế, ví dụ như dự đoán gen gây bệnh dựa trên mạng lưới tương tác protein.
* **Dự đoán liên kết (Link Prediction):** Từ các vector nhúng nút, ứng dụng sẽ sử dụng các kỹ thuật phù hợp để dự đoán các liên kết tiềm ẩn (missing links) giữa các nút trong mạng lưới. Bài toán này có nhiều ứng dụng thực tiễn, chẳng hạn như dự đoán tác dụng phụ của thuốc, dự đoán tương tác thuốc-bệnh, hoặc gợi ý sản phẩm tiềm năng cho khách hàng trong các hệ thống thương mại điện tử.

Nghiên cứu này tập trung vào hai lớp bài toán chính là phân loại nút (node classification) và dự đoán liên kết (link prediction) bởi chúng đại diện cho nhiều bài toán quan trọng trong nhiều lĩnh vực khác nhau. Việc giải quyết hiệu quả hai lớp bài toán này bằng GNNs trên nền tảng Cytoscape sẽ mở ra tiềm năng to lớn trong việc khám phá tri thức từ dữ liệu mạng lưới, đặc biệt là trong lĩnh vực tin sinh học và y sinh.

Chương 2: Thiết kế và cài đặt hệ thống

2.1 Sơ đồ kiến trúc

A screenshot of a computer

Description automatically generated

* Cytoscape (Java): Nền tảng Cytoscape, nơi ứng dụng Java được cài đặt như một plugin.
* Ứng dụng Java thu thập dữ liệu cần thiết (ví dụ: danh sách cạnh, tên nút).
* Ứng dụng Java gửi request (POST) đến server Flask thông qua REST API, dữ liệu được truyền dưới dạng JSON.
* Server Flask nhận request, xử lý dữ liệu, gọi các hàm tương ứng trong train\_Node2Vec, predict\_node\_label\_Node2Vec, train\_GCN, predict\_node\_label\_GCN.
* Các hàm này sử dụng các thư viện như PyTorch, PyTorch Geometric, scikit-learn để huấn luyện hoặc dự đoán.
* Server Flask trả về kết quả (JSON) cho ứng dụng Java.
* Ứng dụng Java nhận kết quả, hiển thị cho người dùng.

2.2 Thiết kế các thành phần

2.2.1 Ứng dụng Cytoscape (Java)

* Các lớp Task: Các lớp này thực thi các tác vụ cụ thể. Chúng kế thừa từ AbstractTask và ghi đè phương thức run().
* CountNodesTask: Đếm số nút trong mạng lưới hiện tại sử dụng CyNetwork.getNodeCount().
* CountEdgesTask: Đếm số cạnh trong mạng lưới hiện tại sử dụng CyNetwork.getEdgeCount().
* DisplayEdgeIndicesTask: Lấy danh sách cạnh và hiển thị dưới dạng source -> target.
* PrintSelectedNodeFeaturesTask: Lấy các thuộc tính của các nút được chọn và hiển thị.
* SendEdgeIndicesTask: Gửi danh sách cạnh (dưới dạng JSON) đến endpoint /receive\_edge\_indices của server Flask.
* predictNodeNode2VecTask: Gửi tên nút được chọn (dưới dạng JSON) đến endpoint /predict\_node\_label\_Node2Vec và hiển thị kết quả dự đoán.
* SendEdgeIndicesAndNodeFeatureTask: Gửi danh sách cạnh và thuộc tính của tất cả các nút (dưới dạng JSON) đến endpoint /receive\_edge\_indices\_and\_features.
* predictNodeGCNTask: Gửi tên nút được chọn (dưới dạng JSON) đến endpoint /predict\_node\_label\_GCN và hiển thị kết quả dự đoán.
* Các lớp TaskFactory: Các lớp này (ví dụ: CountNodesTaskFactory, CountEdgesTaskFactory,...) tạo ra các instance của các lớp Task tương ứng. Chúng kế thừa từ AbstractTaskFactory và ghi đè phương thức createTaskIterator().
* CyActivator: Lớp này đăng ký các TaskFactory vào menu "Apps.MyApp" trong Cytoscape, cho phép người dùng truy cập các chức năng.
* Luồng thực thi “Train on Node2Vec”:

A screenshot of a computer

Description automatically generated

1. Người dùng chọn menu "Apps.MyApp -> Train on Node2Vec".
2. Cytoscape gọi createTaskIterator() trên SendEdgeIndicesTaskFactory.
3. SendEdgeIndicesTaskFactory tạo và trả về một TaskIterator chứa SendEdgeIndicesTask.
4. Cytoscape gọi run(taskMonitor) trên SendEdgeIndicesTask.
5. SendEdgeIndicesTask lấy mạng lưới hiện tại từ Cytoscape.
6. SendEdgeIndicesTask lấy danh sách cạnh từ mạng lưới.
7. SendEdgeIndicesTask tạo JSON payload chứa danh sách cạnh.
8. SendEdgeIndicesTask sử dụng HttpClient để gửi POST request đến /receive\_edge\_indices của server Flask.
9. Server Flask nhận request và gọi hàm train\_model\_Node2Vec(edges).
10. train\_model\_Node2Vec thực hiện huấn luyện mô hình Node2Vec:
    * Tạo ánh xạ node-to-index.
    * Chuyển danh sách cạnh thành edge index.
    * Khởi tạo và huấn luyện mô hình Node2Vec.
    * Lưu embeddings và node mapping vào file Node2Vec\_embeddings.pkl.
11. train\_model\_Node2Vec trả về kết quả cho server Flask.
12. Server Flask trả về response (thành công hoặc lỗi) cho HttpClient.
13. HttpClient trả về response cho SendEdgeIndicesTask.
14. SendEdgeIndicesTask xử lý response và hiển thị kết quả cho người dùng thông qua TaskMonitor.

* Luồng thực thi “Train on GCN”:

A screenshot of a computer screen

Description automatically generated

1. Người dùng chọn menu "Apps.MyApp" -> "Train on GCN".
2. CyActivator gọi SendEdgeIndicesAndNodeFeatureTaskFactory.createTaskIterator().
3. SendEdgeIndicesAndNodeFeatureTaskFactory tạo một instance của SendEdgeIndicesAndNodeFeatureTask.
4. Cytoscape gọi SendEdgeIndicesAndNodeFeatureTask.run().
5. SendEdgeIndicesAndNodeFeatureTask lấy danh sách cạnh và thuộc tính của các nút từ CyNetwork.
6. SendEdgeIndicesAndNodeFeatureTask tạo JSON payload chứa danh sách cạnh và các thuộc tính (features, labels, splits).
7. SendEdgeIndicesAndNodeFeatureTask gửi HTTP POST request đến /receive\_edge\_indices\_and\_features.

2.2.2 Server Flask (Python)

- API Endpoints:

* /receive\_edge\_indices (POST):
* Nhận JSON payload chứa danh sách cạnh.
* Gọi hàm train\_model\_Node2Vec(edges) để huấn luyện mô hình Node2Vec.
* Trả về thông báo thành công hoặc lỗi.
* /predict\_node\_label\_Node2Vec (POST):
* Nhận JSON payload chứa tên nút.
* Gọi hàm predict\_node\_label\_Node2Vec(node\_name) để dự đoán nhãn nút sử dụng Node2Vec.
* Trả về kết quả dự đoán.
* /receive\_edge\_indices\_and\_features (POST):
* Nhận JSON payload chứa danh sách cạnh và thuộc tính của các nút.
* Trích xuất features, labels, splits.
* Gọi hàm train\_model\_GCN(edges, features, labels, splits) để huấn luyện mô hình GCN.
* Trả về thông báo thành công hoặc lỗi.
* /predict\_node\_label\_GCN (POST):
* Nhận JSON payload chứa tên nút.
* Gọi hàm predict\_node\_label\_GCN(node\_name) để dự đoán nhãn nút sử dụng GCN.
* Trả về kết quả dự đoán.
* train\_model\_Node2Vec(edges):
* Tạo ánh xạ node-to-index.
* Chuyển danh sách cạnh thành edge index.
* Khởi tạo mô hình Node2Vec từ thư viện torch\_geometric.nn.
* Huấn luyện mô hình sử dụng torch.optim.SGD.
* Lưu embeddings và node mapping vào file Node2Vec\_embeddings.pkl.
* predict\_node\_label\_Node2Vec(node\_name):
* Load embeddings và node mapping từ Node2Vec\_embeddings.pkl.
* Kiểm tra xem node có tồn tại trong node mapping không.
* Thực hiện phân cụm K-Means trên embeddings.
* Trả về cluster label của node.
* train\_model\_GCN(edges, node\_features, labels, splits):
* Tạo node mapping và label mapping.
* Chuyển danh sách cạnh thành edge index.
* Chuyển features, labels thành tensor.
* Tạo đối tượng Data của PyTorch Geometric.
* Khởi tạo mô hình GCN từ thư viện torch\_geometric.nn.
* Huấn luyện mô hình sử dụng torch.optim.Adam và torch.nn.CrossEntropyLoss.
* Lưu model vào file GCN\_trained\_model.pth.
* Lưu embeddings, node mapping, edge index và label mapping vào file GCN\_embeddings.pkl.
* Lưu features vào file features.pkl.
  + predict\_node\_label\_GCN(node\_name):
* Load embeddings, node mapping, label mapping từ GCN\_embeddings.pkl.
* Load features từ features.pkl.
* Load model từ GCN\_trained\_model.pth.
* Tạo đối tượng Data.
* Dự đoán nhãn nút sử dụng model.
* Trả về predicted label.

2.3 Cài đặt

2.3.1 Ngôn ngữ và thư viện

- Java:

* Cytoscape API: Cung cấp các lớp và phương thức để tương tác với Cytoscape, thao tác với mạng lưới, và phát triển plugin.
* Apache HTTP Components: Sử dụng CloseableHttpClient, HttpPost, StringEntity, EntityUtils để thực hiện các HTTP request đến server Flask.
* Google Gson: Sử dụng để chuyển đổi giữa đối tượng Java và JSON.

- Python:

* Flask: Framework để xây dựng server và cung cấp các API endpoint.
* PyTorch: Thư viện deep learning để xây dựng và huấn luyện mô hình GCN.
* PyTorch Geometric: Thư viện mở rộng của PyTorch, cung cấp các công cụ để làm việc với dữ liệu đồ thị, bao gồm các lớp GCNConv và Node2Vec.
* scikit-learn: Sử dụng KMeans để phân cụm trong predict\_node\_label\_Node2Vec.
* pickle: Sử dụng để lưu và load mô hình, embeddings, và các dữ liệu khác.

2.3.2 Chuyển đổi dữ liệu giữa Java và Python

- Dữ liệu được truyền giữa ứng dụng Java và server Python dưới dạng JSON.

- Java: Sử dụng thư viện Gson để chuyển đổi giữa đối tượng Java (ví dụ: danh sách cạnh) thành chuỗi JSON và ngược lại.

- Python: Sử dụng request.get\_json() để parse JSON payload từ request và jsonify() để tạo JSON response.

2.3.3 Xử lý lỗi

- Java: Sử dụng try-catch để bắt các exception có thể xảy ra khi thực hiện HTTP request (ví dụ: lỗi kết nối) và hiển thị thông báo lỗi cho người dùng thông qua TaskMonitor.

- Python: Sử dụng try-except để bắt các exception (ví dụ: node không tồn tại, lỗi trong quá trình huấn luyện/dự đoán) và trả về thông báo lỗi tương ứng trong JSON response.

2.3.4 Load và sử dụng mô hình

- Node2Vec:

* Model được huấn luyện và lưu embeddings cùng node mapping vào file Node2Vec\_embeddings.pkl.
* Khi dự đoán, hàm predict\_node\_label\_Node2Vec sẽ load file này để lấy embeddings và node mapping.

- GCN:

* Model được huấn luyện và lưu vào file GCN\_trained\_model.pth.
* Embeddings, node mapping, edge index, label mapping được lưu vào file GCN\_embeddings.pkl.
* Features được lưu vào file features.pkl.
* Khi dự đoán, hàm predict\_node\_label\_GCN sẽ load các file này để lấy model, embeddings, features, và các thông tin cần thiết khác.

Chương 3: Các kĩ thuật OOP

3.1 Tính trừu tượng

3.1.1 Ứng dụng trong dự án

- AbstractTask và AbstractTaskFactory: Hai lớp do Cytoscape cung cấp và đóng vai trò là các lớp trừu tượng. Chúng định nghĩa các phương thức trừu tượng (như run() trong AbstractTask, createTaskIterator() trong AbstractTaskFactory) mà các lớp con bắt buộc phải cài đặt.

- Cung cấp framework: Hai lớp trừu tượng này đóng vai trò như một framework cho việc phát triển các Task và TaskFactory. Chúng định nghĩa cấu trúc tổng quát và các phương thức thiết yếu, nhưng không cung cấp chi tiết cài đặt cụ thể.

- Che giấu chi tiết cài đặt: Người sử dụng các lớp Task chỉ cần quan tâm đến việc sử dụng phương thức run() mà không cần hiểu rõ các cơ chế xử lý phức tạp bên trong.

3.1.2 Lợi ích

- Giảm thiểu độ phức tạp: Che giấu các chi tiết cài đặt phức tạp, giúp người sử dụng tập trung vào các khái niệm cốt lõi, tổng quát.

- Thống nhất giao diện: Bắt buộc các lớp con phải tuân theo một chuẩn chung đã được định nghĩa, đảm bảo tính nhất quán trong toàn hệ thống.

- Tạo tiền đề mở rộng: Dễ dàng bổ sung các chức năng mới bằng cách tạo các lớp con mới từ các lớp trừu tượng, cung cấp các cài đặt cụ thể cho các phương thức trừu tượng.

3.2 Tính kế thừa

3.2.1 Ứng dụng trong dự án

- AbstractTask và AbstractTaskFactory: Các lớp Task (như CountNodesTask, CountEdgesTask, ...) đều được thiết kế kế thừa từ lớp AbstractTask do Cytoscape cung cấp. Tương tự, các lớp TaskFactory (như CountNodesTaskFactory, CountEdgesTaskFactory, ...) đều kế thừa từ lớp AbstractTaskFactory.

- Lớp cha cung cấp phương thức cơ bản: Lớp AbstractTask định nghĩa phương thức như run(TaskMonitor monitor), phương thức này được các lớp con kế thừa và tái sử dụng. Lớp AbstractTaskFactory cung cấp phương thức createTaskIterator().

- Ghi đè phương thức (Overriding): Các lớp Task thực hiện ghi đè phương thức run() để định nghĩa các hành động cụ thể tương ứng với từng tác vụ (đếm nút, đếm cạnh,...). Các lớp TaskFactory ghi đè phương thức createTaskIterator() để tạo ra các đối tượng Task thích hợp.

3.2.2 Lợi ích

- Tái sử dụng mã nguồn hiệu quả: Các lớp con không cần phải viết lại các phương thức đã được định nghĩa ở lớp cha, làm giảm lượng mã nguồn trùng lặp.

- Khả năng mở rộng linh hoạt: Các lớp con có thể dễ dàng bổ sung các thuộc tính và hành vi mới, hoặc điều chỉnh hành vi của các phương thức kế thừa từ lớp cha cho phù hợp với yêu cầu nghiệp vụ.

- Tổ chức mã nguồn có cấu trúc: Cơ chế kế thừa tạo ra một hệ thống phân cấp lớp, giúp cho mã nguồn được tổ chức một cách có hệ thống và thuận lợi cho việc quản lý.

3.3 Tính đóng gói

3.3.1 Ứng dụng trong dự án:

- Các lớp Task: Các lớp CountNodesTask, CountEdgesTask, DisplayEdgeIndicesTask, PrintSelectedNodeFeaturesTask, SendEdgeIndicesTask, predictNodeNode2VecTask, SendEdgeIndicesAndNodeFeatureTask, và predictNodeGCNTask minh họa rõ nét cho nguyên tắc đóng gói. Các lớp này che giấu các chi tiết cài đặt cụ thể bên trong. Người sử dụng các lớp này chỉ cần gọi phương thức run(TaskMonitor monitor) mà không cần hiểu rõ cơ chế hoạt động.

- Thuộc tính private: hần lớn các thuộc tính trong các lớp đều được khai báo với bổ từ truy cập private. Việc này ngăn chặn các truy cập trực tiếp và không được kiểm soát vào các thuộc tính, đảm bảo tính toàn vẹn dữ liệu.

3.3.2 Lợi ích

- Giảm thiểu sự phụ thuộc lẫn nhau giữa các module: Các lớp khác không bị phụ thuộc vào chi tiết cài đặt bên trong của một lớp. Nhờ đó, khi có sự thay đổi trong cách thức hoạt động, các lớp sử dụng sẽ không bị ảnh hưởng.

- Bảo mật dữ liệu: Việc thiết lập các thuộc tính ở chế độ private giúp bảo vệ dữ liệu nội tại của đối tượng, ngăn ngừa việc sửa đổi dữ liệu ngoài ý muốn từ các thành phần bên ngoài.

- Nâng cao khả năng bảo trì: Mã nguồn trở nên dễ hiểu, dễ dàng sửa chữa và nâng cấp nhờ các thay đổi được tập trung trong phạm vi của từng lớp, không gây ra các ảnh hưởng dây chuyền.

3.4 Tính đa hình

3.4.1 Ứng dụng trong dự án

- Ghi đè phương thức:

* Các lớp Task thực hiện ghi đè phương thức run() của lớp AbstractTask. Khi một đối tượng Task được thực thi thông qua TaskIterator, phương thức run() của lớp Task cụ thể đó (như CountNodesTask.run() hay CountEdgesTask.run()) sẽ được gọi. Đây là biểu hiện rõ ràng của đa hình thời gian thực thi (runtime polymorphism).
* Các lớp TaskFactory ghi đè phương thức createTaskIterator() của lớp AbstractTaskFactory để trả về TaskIterator chứa các đối tượng Task cụ thể.

- TaskIterator: Đối tượng TaskIterator cho phép thực thi một chuỗi các đối tượng Task khác nhau. Mỗi đối tượng Task có thể đảm nhận một tác vụ riêng biệt, nhưng chúng đều được thực thi thông qua cùng một phương thức run().

3.4.2 Lợi ích

- Tính linh hoạt cao: Cùng một hành động (ví dụ: run()) có thể được thực thi theo nhiều cách khác nhau, tùy thuộc vào kiểu đối tượng Task cụ thể.

- Việc bổ sung các loại Task mới trở nên đơn giản mà không làm thay đổi mã nguồn hiện có (ví dụ: không cần sửa đổi logic thực thi của TaskIterator).

- Mã nguồn rõ ràng, mạch lạc: Giúp cho mã nguồn trở nên dễ đọc và dễ hiểu hơn nhờ các hành động được nhóm lại dưới cùng một tên phương thức.

Chương 4: Đánh giá

**4.1 Điểm mạnh**

- **Tích hợp thành công với Cytoscape**: Ứng dụng được tích hợp liền mạch với Cytoscape dưới dạng một plugin, cho phép người dùng sử dụng các chức năng của ứng dụng ngay trong môi trường quen thuộc của Cytoscape.

- **Giao tiếp hiệu quả với server**: Việc sử dụng REST API và định dạng JSON giúp cho việc giao tiếp giữa ứng dụng Java (Cytoscape) và server Python (Flask) diễn ra hiệu quả và linh hoạt.

- **Triển khai thành công hai mô hình học máy**: Dự án đã triển khai thành công hai mô hình Node2Vec và GCN, cho phép thực hiện các tác vụ học máy trên dữ liệu mạng. Việc tách biệt phần huấn luyện và dự đoán ra khỏi ứng dụng Cytoscape (đặt trên server Flask) giúp tận dụng được sức mạnh của các thư viện Python chuyên sâu cho học máy (PyTorch, PyTorch Geometric).

- **Ứng dụng các kỹ thuật OOP**: Việc áp dụng các nguyên tắc OOP (đóng gói, kế thừa, đa hình, trừu tượng) đã tạo ra một mã nguồn có cấu trúc tốt, dễ đọc, dễ bảo trì và dễ mở rộng.

**4.2 Hạn chế**

- Chưa hỗ trợ nhiều tùy chọn: Hiện tại, các tác vụ huấn luyện và dự đoán trên server Flask chưa cung cấp nhiều tùy chọn cho người dùng (ví dụ: thay đổi các tham số của mô hình, lựa chọn thuật toán phân cụm khác trong Node2Vec).

- Phụ thuộc vào server cục bộ: Hiện tại, ứng dụng đang giao tiếp với server Flask chạy trên localhost. Điều này gây ra hạn chế trong việc triển khai và sử dụng ứng dụng trong môi trường thực tế.

- Chưa triển khai link prediction: Hiện tại, ứng dụng chưa hỗ trợ chức năng dự đoán liên kết.

**4.3 Hướng phát triển**

**-** Triển khai chức năng link prediction.

- Cải thiện giao diện người dùng: cho phép người dùng thay đổi các tham số của mô hình.