ĐẠI HỌC SƯ PHẠM HÀ NỘI

Khoa Công Nghệ Thông Tin

BÁO CÁO BÀI TẬP LỚN

MÔN BIG DATA – Cora Dataset

Giáo viên hướng dẫn: Lê Thị Tú Kiên

Sinh viên:

1. Bùi Thị Hà – 705105025
2. Trần Vũ Phương Uyên – 705105144

Lớp: 70B, 70C

HÀ NỘI, 5/2023

Nội dung

1 Giới thiệu

Vấn đề phân loại bài báo đã là một thách thức lâu dài trong lĩnh vực nghiên cứu học thuật. Khả năng phân loại nhanh chóng và chính xác các bài báo nghiên cứu thành các chủ đề tương ứng có thể hỗ trợ rất nhiều cho các nhà nghiên cứu trong việc tìm kiếm tài liệu liên quan cho công việc của họ. Bộ dữ liệu Cora là bộ dữ liệu điểm chuẩn nổi tiếng trong lĩnh vực này, chứa các tài liệu nghiên cứu từ nhiều chủ đề khác nhau như học máy, hệ thống cơ sở dữ liệu và xử lý ngôn ngữ tự nhiên.

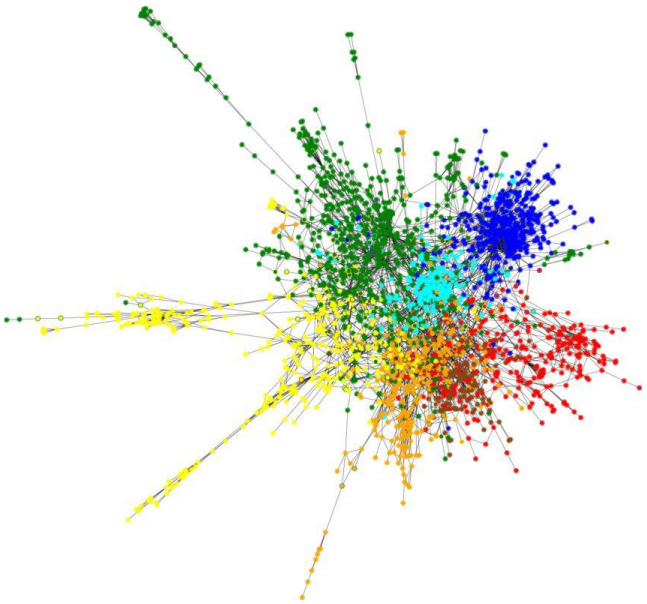

Figure 1: Trực quan hoá cho bộ dữ liệu Cora

Một cách tiếp cận đầy hứa hẹn để giải quyết thách thức phân loại bài viết là sử dụng cơ sở dữ liệu đồ thị như Neo4j. Cơ sở dữ liệu đồ thị được thiết kế để lưu trữ và phân tích hiệu quả các mạng dữ liệu lớn và phức tạp, làm cho chúng rất phù hợp với nhiệm vụ phân loại bài viết. Đặc biệt, Neo4j là một cơ sở dữ liệu đồ thị phổ biến cung cấp một loạt các công cụ và thuật toán mạnh mẽ để phân tích đồ thị.

Một công cụ trong Neo4j đặc biệt hữu ích cho các nhiệm vụ phân loại bài viết là Node Classification Pipeline. Công cụ này sử dụng kết hợp thuật toán đồ thị và kỹ thuật máy học để phân loại các nút (tức là bài báo) trong đồ thị (tức là bộ dữ liệu Cora) dựa trên các thuộc tính của chúng (ví dụ: từ khóa, tác giả, tiêu đề, v.v.). Quy trình phân loại bài báo có thể được định cấu hình để sử dụng các thuật toán khác nhau, chẳng hạn như Random Forest hoặc Gradient Boosted Trees, để thực hiện tác vụ phân loại.

Bằng cách sử dụng quy trình phân loại Nút của Neo4j, chúng tôi có thể đạt được độ chính xác cao trong việc phân loại bài viết đồng thời tận dụng khả năng mở rộng và hiệu suất của cơ sở dữ liệu đồ thị. Hơn nữa, quy trình phân loại Node có thể dễ dàng tùy chỉnh để phù hợp với các nhiệm vụ phân loại cụ thể, cho phép các nhà nghiên cứu điều chỉnh công cụ theo nhu cầu cụ thể của họ. Nhìn chung, quy trình phân loại Node của Neo4j là một công cụ mạnh mẽ để phân loại bài viết và có khả năng hỗ trợ rất nhiều cho các nhà nghiên cứu trong công việc của họ.

2 Mô tả bài toán

Bộ dữ liệu Cora bao gồm 2708 bài báo khoa học được phân loại thành một trong bảy chủ đề. Mạng lưới trích dẫn bao gồm 5429 liên kết. Mỗi bài báo trong bộ dữ liệu được mô tả bằng một vectơ từ có giá trị 0/1 cho biết sự vắng mặt/hiện diện của từ tương ứng trong từ điển. Từ điển bao gồm 1433 từ độc đáo.

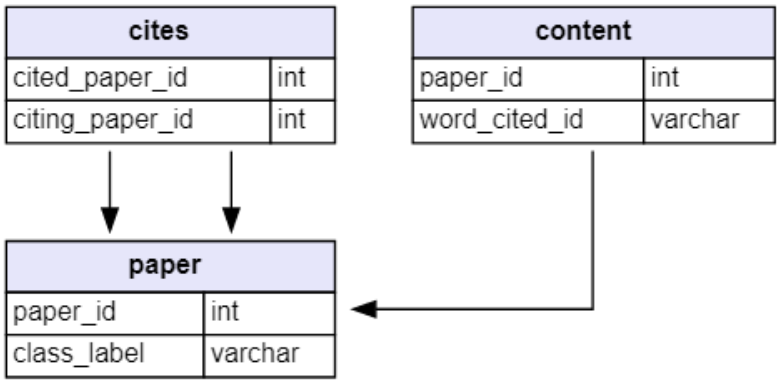

Figure 2: Cấu trúc của bộ dữ liệu Cora

Nhiệm vụ của chúng ta là tiến hành phân loại bài báo trên bộ dữ liệu Cora này. Chúng ta sẽ nạp dữ liệu vào Neo4j, xây dựng một quy trình phân loại sử dụng các kỹ thuật học máy, đánh giá độ chính xác và lưu kết quả vào Neo4j.

Bộ dữ liệu đã được các bên thứ ba cung cấp dưới dạng phù hợp hơn cho chúng ta như sau:

- Bao gồm 2 text file, cora.content và cora.cites.

- File cora.content chứa thông tin của các nút trong đồ thị. Mỗi nút là một bài báo, tương ứng với một dòng trong file, trong đó hai trường đầu tiên là id bài báo và chủ đề, 1433 trường còn lại là vector đặc trưng cho nội dung của bài báo.

- File cora.cites chứa thông tin về các cạnh của đồ thị. Mỗi dòng trong file gồm hai id (id1, id2) thể hiện rằng bài báo với id1 có tham chiếu (hay trích dẫn) tới bài báo với id2.

3 Các bước thực hiện

3.1 Nạp dữ liệu vào Neo4j

3.1.1 Nạp tập đỉnh

Mỗi bài báo thuộc một trong 7 chủ đề, ta đánh số chúng từ 0 đến 6. Sau khi thêm các đỉnh vào, ta đặt 10 đỉnh là chưa gán nhãn `UnclassifiedPaper` để làm dữ liệu kiểm tra.

```
LOAD CSV FROM "https://raw.githubusercontent.com/neo4j/graph-data-
science/master/test-utils/src/main/resources/cora.content" AS row
WITH
{
    `Neural_Networks`: 0,
    `Rule_Learning`: 1,
    `Reinforcement_Learning`: 2,
    `Probabilistic_Methods`: 3,
    `Theory`: 4,
    `Genetic_Algorithms`: 5,
    `Case_Based`: 6
} AS subject_to_id,
toInteger(row[0]) AS extId,
row[1] AS subject,
toIntegerList(row[2..]) AS features
MERGE (p:Paper {
  extId: extId, subject: subject_to_id[subject], features: features
})
WITH p
LIMIT 10
REMOVE p:Paper
SET p:UnclassifiedPaper
```

3.1.2 Nạp tập cạnh

```
LOAD CSV FROM "https://raw.githubusercontent.com/neo4j/graph-data-
science/master/test-utils/src/main/resources/cora.cites" AS row
MATCH (n), (m)
WHERE n.extId = toInteger(row[0]) AND m.extId = toInteger(row[1])
```

```
MERGE (n)-[:CITES]->(m)
```

Với dữ liệu được tải trên Neo4j, giờ đây chúng ta có thể chiếu một đồ thị bao gồm tất cả các nút và mối quan hệ `CITES` dưới dạng vô hướng (và với aggregation `SINGLE`, để bỏ qua các mối quan hệ lặp lại do thêm hướng nghịch đảo).

```
G, _ = gds.graph.project(
    "cora-graph",
    {"Paper": {"properties": ["features", "subject"]},
"UnclassifiedPaper": {"properties": ["features"]}},
    {"CITES": {"orientation": "UNDIRECTED", "aggregation": "SINGLE"}},
)
```

Việc thêm mối quan hệ nghịch hướng là cần thiết để xây dựng đồ thị vô hướng trong ngữ cảnh của bài toán phân loại bài viết. Đồ thị vô hướng coi mối quan hệ giữa các nút là hai chiều, nghĩa là các mối quan hệ có giá trị như nhau theo cả hai hướng. Trong ngữ cảnh của bộ dữ liệu Cora, mối quan hệ CITES thể hiện việc trích dẫn một bài báo này bởi một bài báo khác. Vì một trích dẫn là một mối quan hệ có hướng, nên cần phải thêm mối quan hệ có hướng nghịch đảo để tạo ra một đồ thị vô hướng xử lý cả ấn phẩm trích dẫn và ấn phẩm được trích dẫn như nhau.

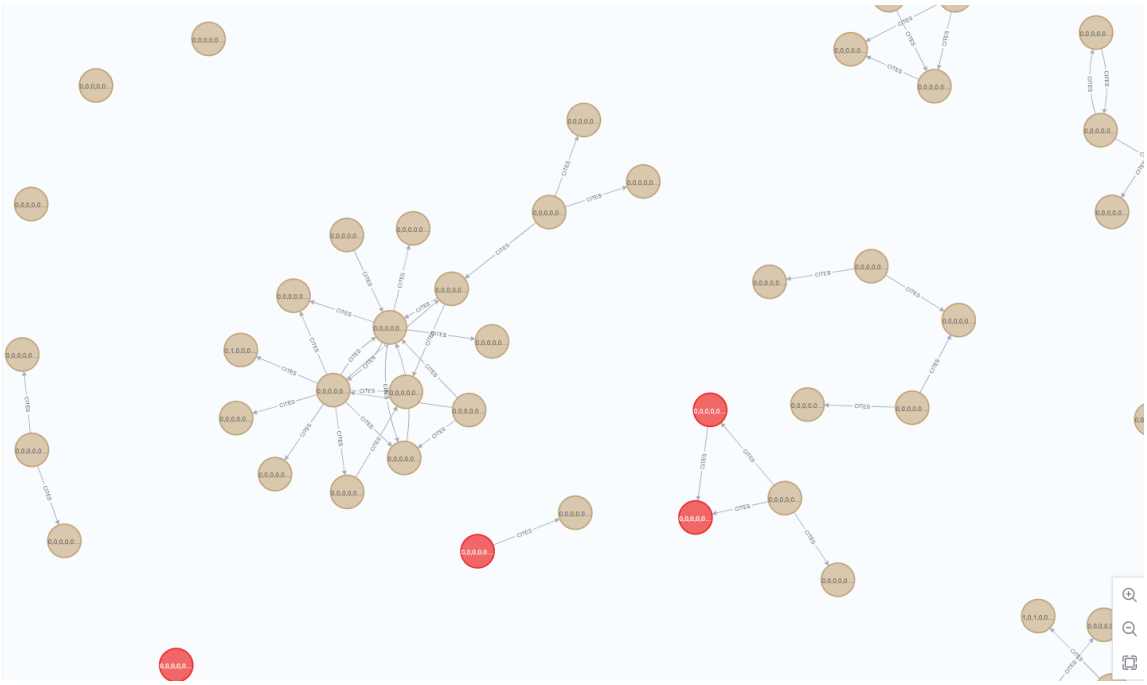

Figure 3: Trực quan hoá đồ thị đã được nạp (ở đây chỉ bao gồm một phần nhỏ của đồ thị, các nút màu đỏ là các nút đã bị xoá nhãn)

3.2 Cấu hình pipeline

Khi tập dữ liệu đã được tải, chúng ta có thể xác định quy trình học máy phân loại nút.

```
# Create the pipeline
node_pipeline, _ = gds.beta.pipeline.nodeClassification.create("cora-pipeline")
```

Bây giờ chúng ta có thể cấu hình pipeline. Các bước bao gồm:

1. Chọn mộn tập các thuộc tính để sử dụng làm features cho mô hình học máy

2. Đặt tỷ lệ chia train/test và số lượng folds cho k-fold cross-validation

3. Chọn các mô hình học máy đề cử

4. Cài đặt auto-tunning.

```
# "Mark" some node properties that will be used as features
node_pipeline.selectFeatures(["features"])
```

```
# If needed, change the train/test split ratio and the number of folds
# for k-fold cross-validation
node_pipeline.configureSplit(testFraction=0.2, validationFolds=5)

# Add a model candidate to train
node_pipeline.addLogisticRegression(maxEpochs=200, penalty=(0.0, 0.5))

# Explicit set the number of trials for autotuning (default = 10)
node_pipeline.configureAutoTuning(maxTrials=5)
```

3.3 Huấn luyện pipeline

Pipeline đã được cài đặt sẵn sàng cho việc lựa chọn và huấn luyện một mô hình. Trước hết cần phải ước lượng việc huấn luyện có đủ tài nguyên hay không.

```
# Estimate the resources needed for training the model
node_pipeline.train_estimate(
    G,
    targetNodeLabels=["Paper"],
    modelName="cora-pipeline-model",
    targetProperty="subject",
    metrics=["F1_WEIGHTED"],
    randomSeed=42,
    concurrency=4,
)
```

Sau đó thực hiện quá trình huấn luyện. Độ đo để đánh giá là F1_weighted.

```
# Perform the actual training
model, stats = node_pipeline.train(
    G,
    targetNodeLabels=["Paper"],
    modelName="cora-pipeline-model",
    targetProperty="subject",
    metrics=["F1_WEIGHTED"],
    randomSeed=42,
    concurrency=4,
)
```

Sau đó chúng ta có thể xem kết quả huấn luyện bằng cách in ra giá trị độ đo đánh giá:

```
# Print F1_WEIGHTED metric
```

```
stats["modelInfo"]["metrics"]["F1_WEIGHTED"]["test"]
```

Kết quả thu được là F1_weighted = 0.73

```
0.7287325951256631
```

3.4 Dự đoán sử dụng pipeline

Sau khi được huấn luyện, mô hình đã sẵn sàng để phân loại dữ liệu chưa được phân loại.

```
model.predict_mutate(
    G,
    mutateProperty="predictedClass",
    modelName="cora-pipeline-model",
    predictedProbabilityProperty="predictedProbabilities",
    targetNodeLabels=["UnclassifiedPaper"],
)

predicted = gds.graph.streamNodeProperty(G, "predictedClass",
["UnclassifiedPaper"])
```

Ta ghép nhãn dự đoán với nhãn thực tế để so sánh kết quả

```
# Retrieve node information from Neo4j using the node IDs from the
prediction result
nodes = gds.util.asNodes(predicted.nodeId.to_list())

# Create a new DataFrame containing node IDs along with node properties
nodes_df = pd.DataFrame([(node.id, node["subject"]) for node in nodes],
columns=["nodeId", "subject"])

# Merge with the prediction result on node IDs, to check the predicted
value
# against the original subject
predicted.merge(nodes_df, on="nodeId")
```

	nodeId	propertyValue	subject
0	0	0	0
1	1	5	1
2	2	2	2
3	3	2	2
4	4	3	3
5	5	5	3
6	6	6	4
7	7	0	0
8	8	0	0
9	9	4	4

Kết quả cho thấy các nhãn dự đoán (ở cột propertyValue) hầu hết giống với nhãn thực tế (subject).

3.5 Ghi kết quả vào Neo4j

Giờ đây chúng ta có thể ghi kết quả dự đoán trở lại cơ sở dữ liệu Neo4j.

```
gds.graph.nodeProperties.write(
    G,
    node_properties=["predictedClass"],
    node_labels=["UnclassifiedPaper"],
)
```

4 Kết luận

Trong bài tập lớn này, chúng tôi đã thực hiện nghiên cứu vận dụng hệ quản trị cơ sở dữ liệu Neo4j để nạp bộ dữ liệu Cora về mạng lưới tham chiếu của các bài báo khoa học, sau đó xây dựng một pipeline học máy để dự đoán chủ đề của các bài báo đó. Kết quả cho thấy mô hình của chúng tôi đạt độ chính xác khoảng 70% trên quá trình huấn luyện cũng như kiểm tra.

Qua bài tập này, chúng tôi rút ra được nhiều kinh nghiệm và kiến thức về dữ liệu đồ thị, hệ quản trị dữ liệu đồ thị và các công cụ phân tích khoa học trên chúng. Trong tương lai, chúng tôi sẽ nghiên cứu và sử dụng nhiều kỹ thuật hơn để đạt được kết quả tốt hơn cho bài toán, cũng như cập nhật kiến thức mới nhất trong lĩnh vực này.