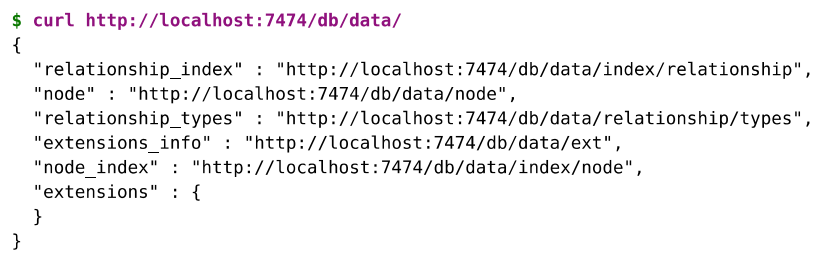
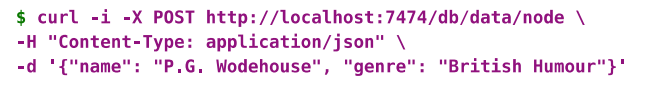
## 7.3 TÌM HIỂU VỀ REST, INDEXES VÀ ALGORITHMS

* Rest:
* Tổng quan: Rest là interface có thể sử dụng để tạo node(nút) và relationship(quan hệ giữa các nút). Sau đó dùng rest để đánh chỉ mục và thực hiện tìm kiếm các node mong muốn. Ở đây chúng ta sẽ cùng tìm hiểu cách sử dụng Gremlin query để truy vấn thông qua Rest API.
* Hiểu về rest interface: Bước đầu kiểm tra xem Rest server có đang chạy hay không bằng
* Chạy đoạn mã để kiểm tra server hoạt động hay không, kết quả sẽ trả về dạng json mổ tả url được gọi.



* Tạo node bặng rest:
* Gửi một phương thức post đến /db/data/node path với dữ liệu json:

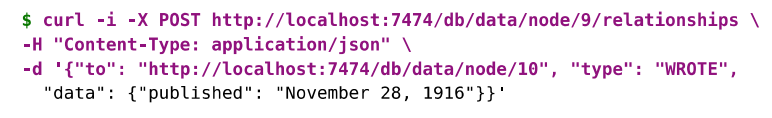


Nếu muốn them thuộc tính cho một node. Có thể dùng GET để them thuộc tính vào URL của node hoặc với nhưng property đặc biệt thì them vào tên thuộc tính.

Ví dụ: Curl <http://localhost:7474/db/data/node/9/properties/genre> “British Humour”.

Tương tự thêm nhiều node với các thuộc tính [“name” : “Jeeves Takes Charge”, “tyle” : “short story”].

Thêm mối qua hệ giữa hai node thể hiện P. G. Wodehouse viết truyện ngắn “Jeeves Takes Charge”.

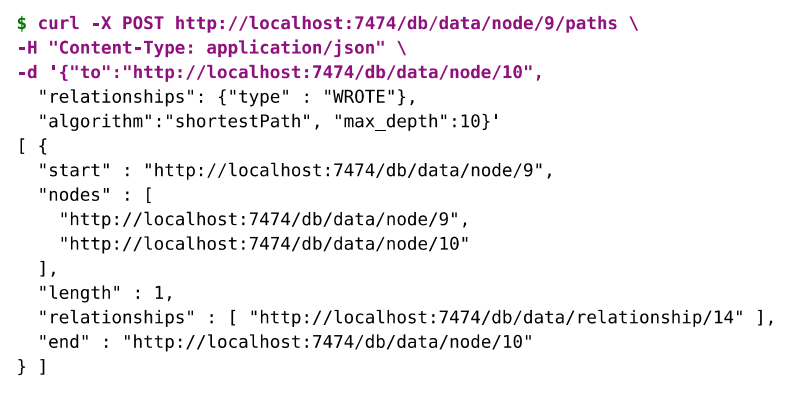


Điểm hay của REST interface là mô tả cách tạo quan hệ sớm trong thân của thuộc tính metadata’s create\_relationship

* Tìm đường (Finding your path):

Thông qua rest, bạn có thể tìm đường giữa hai node bằng cách gửi một request dữ liệu để bắt đầu đường dẫn URL bằng phương thức POST với kiểu JSON thể hiện thông tin node muốn tìm, loại relationship, thuật toán tìm kiếm được sử dụng.

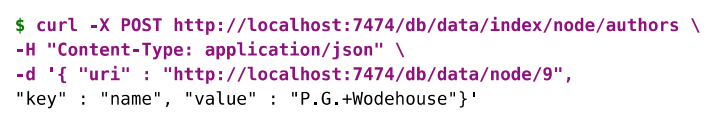
Ví dụ: Chúng ta đang tìm một quan hệ WROTE từ 1 node dùng thuật toán ShortesPath



Ngoài ra có thể dùng các thuật toán như allPaths, allSimplePaths, diskstra.

* Indexes( Đánh chỉ mục): giống như nhiều cơ sở dữ liệu khách, NEO4J đánh chỉ mục để hỗ trợ việc tìm kiếm nhanh hơn. Nhưng không giống những csdl đó, NEO4J đánh chỉ mục phúc tạp hơn vi cần nhiều chỉ mục để thể hiện mối quan hệ với các node khác nhau, chỉ vì vậy mà Indexes Service được tách riêng.

Một cách chỉ mục đơn giản là sử dụng key value hoặc loại hash. Khi sử dụng key value thì key dùng lưu chỉ mục, value lưu REST URL chỉ đến node tương ứng trên đồ thị. Có thể có nhiều indexes vậy nên bạn hãy them tên indexes vào URL:



Để lấy được dữ liệu chỉ đơn giản gọi index là “name” đã đánh ở ví dụ trên:

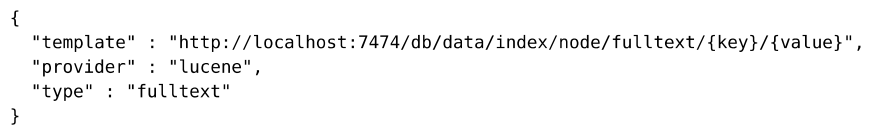


Tương tự như key-value, chúng ta có thể xây dựng index với full-text search. Ví dụ dưới đây đánh chỉ mục để lấy tất cả các cuốn sách có tên bắt đầu với “Jeeves”.

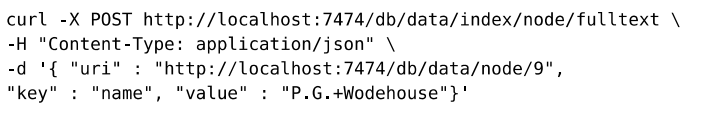
Trước hết xây dựng inverted index:



Sau đó POST sẽ tra về JSON:



Nếu add Wodehouse vào fulltext chung ta được:



Bây giờ việc tìm kiếm Lucene đơn giản gọi:



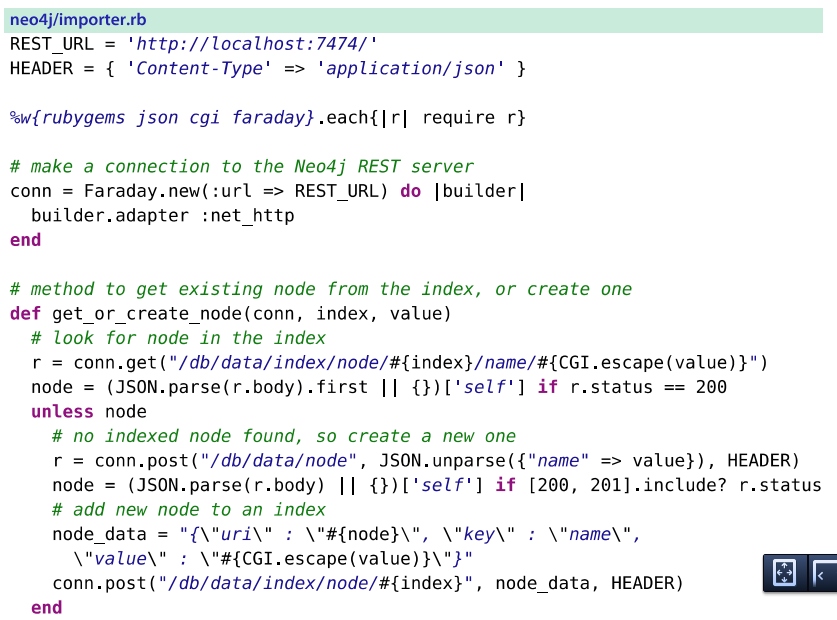
Chỉ mục cũng có thể được xây dựng trên cạnh nối các node, thay các node bang relationship.

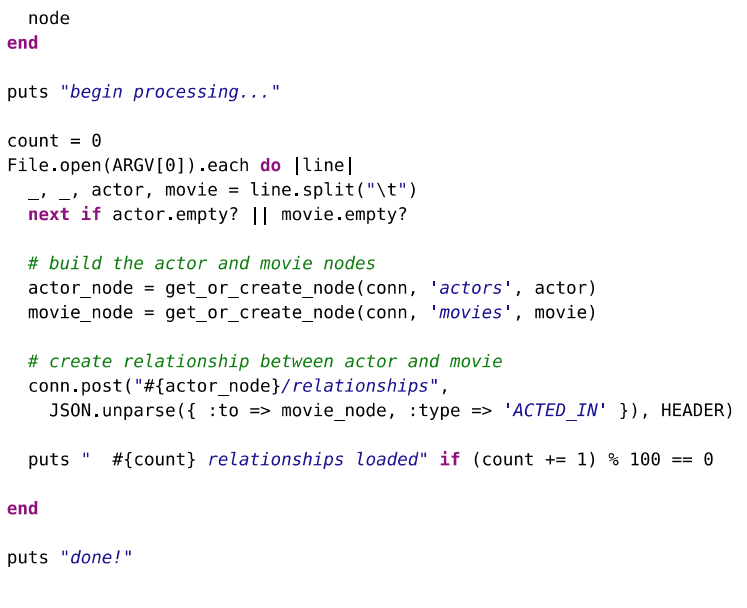
* Rest and Gremlin:

Chúng ta đang phân vân là nên dùng Rest interface hay là truy vấn bằng Gremlin, nhưng thực sự thì REST đã cài sẳn plugin Gremlin, cho nên có thể gửi thông qua rest các câu lệnh thực hiện trên Gremlin console. Nó giúp bạn có thể dễ dàng sử dụng, linh động với hai cách với Gremlin sử dụng sức mạnh của truy vấn, REST là khả năng triễn khai và ngôn ngữ linh động

* BIG DATA:

Tìm hiểu về hiểu quả khi sử dụng NEO4J với big data. Thử với dữ liệu từ trang web phim Freebase.com. Chúng ta sẽ sữ dụng “perfomance”tab-separated. Dowload script từ đường dẫn về, chúng tạo ra các dòng với các quan hệ giữa các node. Có thể được tìm kiếm bằng chỉ mục name. Dữ liệu có thể tồn tại số lượng lớn thông tin phim, nên phải cài đặt ruby hoặc JSON để chạy script.



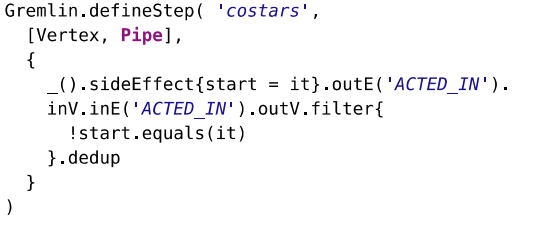


Sau khi cài đặt thiết lập xong ruby thì chỉ việc chỉ đến file lưu dữ liệu vừa tải và chạy:

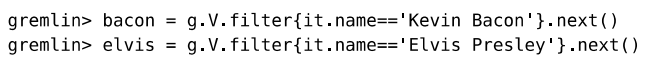


**Fancy Algorithms:** Với big data vừa chạy chúng ta sẽ dừng Rest một lúc để quay trở lại với Gremlin.

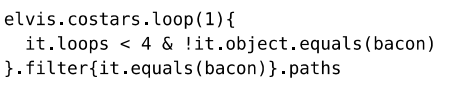
Thuật toán Kenvin Bacon: Là thuật toán đồ thì nội tiếng. Đây là thuật toán tìm khoảng cách ngắn nhất giữa hai actor được sử dụng trong ngành điện ảnh đầu tiên. Chúng ta sẽ xây dựng ví dụ tìm bạn diễn của một diễn viên. Trước hệt chạy Gremlin console và chạy đồ thị sau đó xây dựng costars( bạn diễn) custom theo code dưới đây:



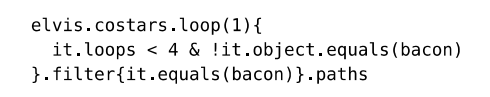
Trong NEO4J chúng ta không dùng nhiều truy vấn cho viết thiết lập dữ liệu. Điểm hay là từ node đầu tiên sẽ chạy qua node gần đó nhất (khoảng cách cạnh và nút). Dưới đây thể hiện điểm bắt đầu và điểm kết thúc.



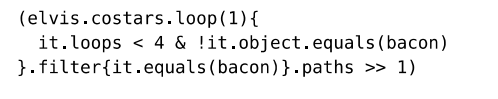
Thêm điều kiện break khỏi vòng lặp:



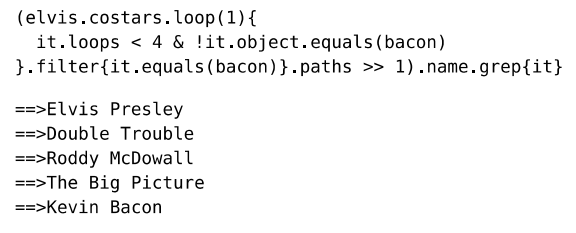
Chỉ để chắc chắn chúng ta không muốn tiếp lục lặp thuật toán Kenvin Bacon cho lần thứ 2, chuyển nút bacon qua vòng lặp ngắn. Nói cách khác là vòng lặp xãy ra nếu chưa đặt 4 lần, chúng ta chưa tìm được bacon node. Sau đó chúng ta xuất đường dẫn để đến được mỗi nut bacon tiếp theo.



Như vậy, chúng ta chỉ đẩy đường dẫn đầu của list nhiều đường dẫn khả dụng, tức là đường ngắn nhất được đến tại vị trí 1. Dùng >> để đẩy item đầu tiên ra khỏi list tất cả các node.



Cuối cùng chúng ta có thể lấy tên của mỗi đĩnh đầu và lọc ra bất ki cạnh không có giá trị bằng Groovy grep command.



Thuật toán Random walk:

Khi tìm kiếm đối tượng tốt từ một tập dữ liệu lớn, cách hữu dụng là “random walk”. Bắt đầu với một số :

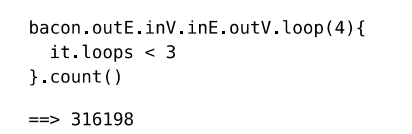
Rand = new Random();

Sau đó lọc ra tỉ lệ đích trên tổng số. Nếu chúng ta muôn lây về chỉ 1/3 của Kenvin Bacon xấp xĩ 60 bộ film, chúng ta có thể lọc bất kì số random nào nhỏ hơn 0.33.

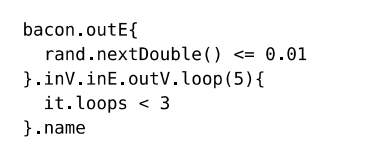
Bacon.outE.filter{rand.nextDouble()<=0.33}.inV.name

Số lượng nên là những nơi bao quanh 20 random.

Gỉa sử đỉnh thứ 2 từ Kevin Bacon. Bạn diễn của bạn diễn anh ấy hoàn toàn là một list(nhiều hơn 300,000 ).



Nếu bạn cần 1 % của list trên, thêm một filter. Chú ý thêm filter là một bước, do vậy bạn sẽ cần thêm một bước nữa vào số vòng lặp.

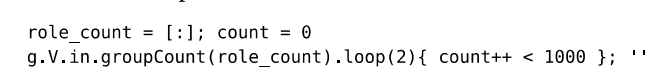


**Thuật toán Centrality Park:**

Centrality là một các ướng lượng những node độc lập với cả đồ thị. Ví dụ: Nếu chúng ta muốn ướng lượng độ quan trọng của mỗi nút trong mạng la dựa vào khoảng cách của nó so với tất cả các node còn lại. Nó yêu cầu sử dụng thuật toán centrality.

Thuật toán centrality nỗi tiếng nhất chỉ có thể là Google’s PageRand, nhưng có một vài kiểu. Chúng ta sẽ thực thi một version đơn giản gọi là eigenvector centrality, no chỉ đếm số cạnh vào ra liên quan đến một node. Chúng ta sẽ cho mỗi actor 1 con số liên quan đến bao nhiêu vai diễn họ có.

Chúng ta cần một bản đồ cho groupCount() để thực hiện và một count để đặt giá trị lặp cực đại.



Muốn sắp xếp output dùng lệnh: role\_count.sort{a,b -> a.value ⬄ b.value}

Kết quả cuối cùng sẽ là actor với số lược vai diễn lớn nhất.

**Thuật toán External Algorithms:**

Tự viết thuật toán thì ổn, nhưng đa số những việc đã có sẵn cách làm. Java Universal Network/Graph (JUNG) Framework la một tập hợp các thuật toán đồ thì phổ biến. Nhờ có dự án Gremlin/Blueprint, chúng ta dễ dàng truy cập được nhưng thuật toán của JUNG. Ví dụ: PageRank, HITS, Voltage, centrality, graph-as-a-matrix.

Để dùng JUNG chúng ta cần gói Neo4j vào trong một đồ thị JUNG mới. Để truy cập đồ thị JUNG graph, chúng ta cần thực hiện một trong hai cách: Tải và cài đặt Blueprint và JUNG (file jar) vào thư mục Neo4j server libs sau đó khởi động lại server, hoặc tải gói cài đặt trước Gremlin console.

Gỉa sử bạn đã tải xong gremlin console, đóng neo4j server và khởi đọng Gremlin. Bạn sẽ phải tạo đối tượng Neo4jGraph và chỉ nó đến thư mục cái đặt data/graph.

G = new Neo4jGraph(‘/users/x/neo4j-enterprise-1.7/data/graph.db’)

Chúng ta sẽ giữ Gremlin graph tên là g. Neo4jGraph object cần được gói lại trong GraphJung object:

J= new GraphJung( g )

Một phần lí do Kenvin Bacon được chọn giống nhưu đường dẫn đích cơ bản là vì mối qua hệ gần với các actor khác. Một điều qua trọng, anh ấy không cần nhiều vai diễn nhưng đơn giản là có liên kết với những người nhiều vai diễn.

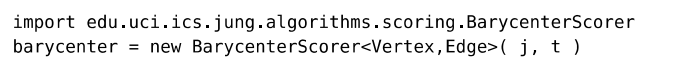
Có một câu hỏi phát sinh: chúng ta có thể tìm một actor tốt hơn Kenvin Bacon?

JUNG chưa thuật toán tính điểm gọi là BarycenterScorer cái cho một số điểm đến mỗi đĩnh dựa vào khoảng cách đến các đỉnh khác. Nếu Kenvin Bacon là lựa chọn tốt nhất, chúng ta sẽ mông đợi điểm của anh ấy sẽ là nhỏ nhất, nghĩa là anh ấy gần nhất so với các actor còn lại.

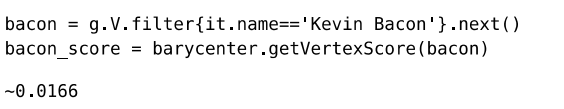
Thuật toán JUNG của chúng ta nên áp dụng cho tập nhiều actor, do vậy chúng ta có thể cài một transformer để tìm chỉ một node. EdgeLabelTransformer cho phép chỉ những node đó với một cạnh của ACTED\_IN đến thuật toán:

T = new EdgeLableTransformer([ACTED\_IN’)] as Set, false)

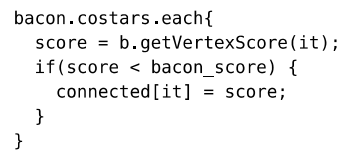
Tiếp theo chúng ta cần import thuật toán, truyền vào GraphJung and Transformer.



Với nó, chúng ta có thể lấy điểm BarycenterScorer của nhiều node. Hãy tìm ra những điểm đó:



Khi chúng ta có điểm của Kenvi Bacon chúng ta có thể duyệt qua mỗi đỉnh và lưu bất kì đỉnh nào có điểm thấp hơn. Nó có thể mất một thời gian dài để chạy điểm BarycenterScorer cho mỗi actor trong database. Do vậy, chúng ta chỉ chạy thuật toán với mỗi bạn diễn của Kenvin. Nó có thể mất vài phút, tùy thuộc vào phần cứng. BarycenterScorer thí nhanh, nhưng chậm ở quá trình chuyễn đổi giữa các bạn diện được thêm vào:



Tất cả các key tồn tại trong bản đồ kết nối biểu hiện một lựa chọn tốt hơn Kenvin Bacon. Nhưng nó tốt để có một cái tên chúng ta dễ đọc, do vậy output tất cả các node và lấy một cái chúng ta thích.

Với j graph vừa tạo ở trên, chúng ta có thể chạy bất kì thuật toán JUNG nào trong dataset, nhưn: PageRank.

Dang sách tất cả các thuật toán JUNG có thể tìm trên Javadoc API online.