Giới thiệu về Apache Mahout

Học máy thân thiện với thương mại, có khả năng mở rộng để xây dựng các ứng dụng thông minh

Mức độ: Trung bình

Grant Ingersoll, Kỹ sư phần mềm cao cấp, Center for Natural Language Processing at Syracuse University

18 12 2009

Một khi lĩnh vực độc quyền của các viện nghiên cứu và các tổng công ty có ngân sách nghiên cứu lớn là các ứng dụng thông minh học được từ các dữ liệu và đầu vào của người dùng đang trở nên phổ biến hơn. Nhu cầu về các kỹ thuật học máy như phân cụm, lọc cộng tác và phân loại chưa bao giờ lớn hơn bây giờ, do nó cho phép thấy được sự tương đồng giữa các nhóm đông người hoặc tự động thêm vào khối lượng lớn nội dung Web. Dự án Apache Mahout làm cho việc xây dựng các ứng dụng thông minh dễ dàng hơn và nhanh hơn. Người đồng sáng lập Mahout Grant Ingersoll giới thiệu các khái niệm cơ bản về học máy rồi trình diễn cách sử dụng Mahout để phân cụm các tài liệu, bình luận và tổ chức nội dung.

Dần dần sự thành công của các công ty và những cá nhân trong thời đại thông tin phụ thuộc vào cách họ chuyển số lượng lớn dữ liệu sang thông tin hành động một cách nhanh và hiệu quả như thế nào. Cho dù đó là để xử lý hàng trăm hoặc hàng ngàn thư điện tử (e-mail) cá nhân một ngày hoặc đoán biết ý định của người dùng từ hàng triệu tỷ byte (petabyte) của các weblog, sự cần thiết có các công cụ có thể tổ chức và tăng cường dữ liệu chưa bao giờ lại lớn đến như vậy. Điểm then chốt của giả thuyết và triển vọng của lĩnh vực *học máy* và dự án mà bài viết này giới thiệu là: Apache Mahout (xem Tài nguyên).

Học máy là một lĩnh vực của trí tuệ nhân tạo, đề cập các kỹ thuật cho phép các máy tính cải thiện kết quả đầu ra của chúng dựa trên kinh nghiệm có trước. Lĩnh vực này liên quan chặt chẽ đến việc khai thác dữ liệu và thường sử dụng các kỹ thuật từ thống kê, lý thuyết xác suất, nhận dạng và một loạt các lĩnh vực khác. Mặc dù học máy không phải là một lĩnh vực mới, nó phát triển chắc chắn. Nhiều công ty lớn, gồm cả IBM®, Google, Amazon, Yahoo! và Facebook, đã triển khai thực hiện các thuật toán học máy trong các ứng dụng của họ. Càng ngày càng có nhiều công ty sẽ được hưởng lợi từ việc sử dụng học máy trong các ứng dụng của mình để tìm hiểu về người dùng và những tình huống quá khứ.

Sau khi đưa ra một tổng quan ngắn về khái niệm học máy, tôi sẽ giới thiệu cho bạn về các đặc tính, lịch sử và các mục tiêu của dự án Apache Mahout. Sau đó, tôi sẽ cho bạn thấy cách sử dụng Mahout để thực hiện một số nhiệm vụ học máy thú vị bằng cách sử dụng tập dữ liệu Wikipedia có sẵn sàng miễn phí.

101 cách học máy

Học máy sử dụng một loạt đầy đủ từ việc chơi trò chơi để phát hiện gian lận đến phân tích thị trường chứng khoán. Nó được sử dụng để xây dựng các hệ thống như các hệ thống ở Netflix và Amazon để đề xuất các sản phẩm cho người dùng dựa trên các việc mua hàng trước đây hoặc các hệ thống để tìm tất cả các bài viết thời sự tương tự trong một ngày cụ thể. Nó cũng có thể được dùng để phân loại tự động các trang Web theo thể loại (thể dục thể thao, kinh tế, chiến tranh và v.v) hoặc để đánh dấu e-mail là thư rác. Việc sử dụng học máy có nhiều thứ hơn là tôi có thể trình bày trong bài viết này. Nếu bạn quan tâm đến việc khám phá lĩnh vực này sâu hơn, tôi khuyên bạn nên tham khảo phần Tài nguyên.

Một số cách tiếp cận đến học máy thường sử dụng để giải quyết các vấn đề. Tôi sẽ tập trung vào hai vấn đề được sử dụng chung nhất — học *có giám sát* và học *không giám sát* — vì chúng là những vấn đề chính được Mahout hỗ trợ.

Học có giám sát được giao nhiệm vụ học một chức năng từ dữ liệu huấn luyện được ghi nhãn để dự đoán giá trị của bất kỳ đầu vào hợp lệ nào. Các ví dụ thường gặp của học có giám sát gồm việc phân loại các e-mail là thư rác, ghi nhãn các trang Web theo thể loại của chúng và nhận dạng chữ viết tay. Nhiều thuật toán được sử dụng để tạo những trình học (learner) có giám sát, phổ biến nhất là các mạng thần kinh, Support Vector Machines (SVMs-Các máy Vectơ hỗ trợ) và các trình phân loại (classifier) Naive Bayes.

Học không giám sát, như bạn có thể đoán, được giao nhiệm vụ có ý nghĩa về dữ liệu mà không có các ví dụ bất kỳ về cái gì là đúng hay sai. Nó hầu như thường được sử dụng để phân cụm đầu vào tương tự thành các nhóm hợp lý. Nó

cũng có thể dùng để giảm số lượng chiều trong một tập dữ liệu để chỉ tập trung vào các thuộc tính có ích nhất hoặc để phát hiện các xu hướng. Các cách tiếp cận phổ biến để học không giám sát gồm k-Means, phân cụm theo phân cấp và bản đồ tự tổ chức.

Đối với bài này, tôi sẽ tập trung vào ba nhiệm vụ học máy cụ thể mà Mahout hiện nay triển khai thực hiện. Chúng ngẫu nhiên là ba lĩnh vực thường được sử dụng trong các ứng dụng thực tế:

- Lọc cộng tác
- Phân cụm
- Phân loại

Tôi sẽ xem xét sâu hơn mỗi một nhiệm vụ này ở mức khái niệm trước khi khảo sát việc thực hiện của chúng trong Mahout.

Lọc cộng tác

Lọc cộng tác (*Collaborative filtering*-CF) là một kỹ thuật, được Amazon và những hãng khác phổ biến, sử dụng thông tin của người dùng như sự xếp loại, sự nhấn chuột và sự mua hàng để đưa ra những bình luận cho người dùng của trang Web khác. CF thường được sử dụng để bình luận các mục cho người dùng như là sách, âm nhạc và phim ảnh, nhưng nó cũng được sử dụng trong các ứng dụng khác, nơi nhiều diễn viên cần phải cộng tác để thu hẹp bớt dữ liệu. Có thể bạn đã nhìn thấy CF đang hoạt động trên Amazon, như trong Hình 1:

Hình 1. Ví dụ về lọc cộng tác trên Amazon

Dựa vào một tập những người dùng và các mục, các ứng dụng CF cung cấp các bình luận cho người dùng hiện tại của hệ thống. Bốn cách tạo các bình luận điển hình:

- **Dựa vào-người dùng**: Bình luận các mục bằng cách tìm những người dùng tương tự. Điều này thường khó mở rộng do đặc tính động của người dùng.
- **Dựa vào-mục**: Tính toán sự giống nhau giữa các mục và đưa ra các bình luận. Các mục thường không thay đổi nhiều, do đó, điều này thường có thể được ước tính ngoại tuyến (offline).
- **Nghiêng về một phía**: Một cách tiếp cận bình luận dựa vào mục rất nhanh chóng và đơn giản có thể áp dụng khi người dùng đã đưa ra sự xếp loại (và không chỉ theo logic đúng/sai).
- **Dựa vào-mô hình**: Đưa ra các bình luận dựa vào phát triển mô hình của những người dùng và sự xếp loại của chúng.

Tất cả các cách tiếp cận của CF kết thúc việc tính toán một khái niệm tương đương giữa người dùng và các mục đã đánh giá của họ. Có nhiều cách để tính toán sự tương đương và hầu hết các hệ thống CF cho phép bạn gắn vào các phép đo khác nhau để bạn có thể xác định phép đo nào hoạt động tốt nhất đối với dữ liệu của bạn.

Phân cụm

Dựa vào các tập dữ liệu lớn, cho dù chúng là văn bản hoặc số, để tự động nhóm với nhau, hoặc *phân cụm*, các mục tương tự thường rất có ích. Ví dụ, căn cứ vào tất cả các tin tức trong ngày từ tất cả các tờ báo ở Hoa Kỳ, bạn có thể muốn tự động nhóm tất cả các bài viết về câu chuyện giống nhau cùng với nhau, sau đó bạn có thể chọn tập trung vào các nhóm và các câu chuyện cụ thể mà không cần phải vất vả lướt qua nhiều việc không liên quan. Một ví dụ khác: căn cứ vào đầu ra từ bộ cảm biến trên máy trên toàn bộ thời gian, bạn có thể nhóm các kết quả đầu ra để xác định hoạt động bình thường so với hoạt động có vấn đề, vì các hoạt động bình thường sẽ nhóm tất cả lại với nhau và các hoạt động bất thường sẽ ở trong các nhóm khác.

Có nhiều cách tiếp cận để tính toán các cụm, mỗi cụm có sự thỏa hiệp riêng của mình. Một số cách tiếp cận thực hiện từ dưới lên, xây dựng các cụm lớn hơn từ cái nhỏ hơn, trong khi những cách khác lại chia một cụm lớn thành nhiều cụm càng nhỏ hơn càng tốt. Cả hai đều có tiêu chuẩn để thoát ra khỏi quá trình này tại một số điểm trước khi chúng phân ra thành một sự biểu diễn cụm tầm thường (tất cả các mục trong một cụm hoặc tất cả các mục trong cụm riêng của chúng). Các cách tiếp cận phổ biến gồm phân cụm k-Means và phân cụm theo hệ thống phân cấp. Như tôi sẽ trình bày sau, Mahout đi kèm với một số cách tiếp cận phân cụm khác nhau.

Phân loại

Mục tiêu của sự *phân loại* (thường được gọi là phân lớp-*classification*) là để ghi nhãn các tài liệu vô hình, theo đó nhóm chúng lại với nhau. Nhiều cách tiếp cận phân loại theo cách học máy tính toán một loạt các số liệu thống kê để kết hợp các đặc tính của một tài liệu với nhãn cụ thể, vì thế tạo một mô hình có thể được sử dụng sau này để phân loại các tài liệu vô hình. Ví dụ, một cách tiếp cận đơn giản cho việc phân loại có thể theo dõi các từ có liên kết với một nhãn, cũng như số lần những từ được nhìn thấy với một nhãn nhất định. Sau đó, khi một tài liệu mới được phân loại, các từ trong tài liệu này được tìm kiếm trong mô hình này, các khả năng xảy ra được tính toán và kết quả tốt nhất được đưa ra, thường là cùng với một điểm số cho thấy sự tin tưởng là kết quả chính xác.

Các đặc tính phân loại có thể gồm các từ, các trọng số đối với những từ đó (ví dụ, dựa trên tần suất), các phần của bài phát biểu, v.v. Tất nhiên, các đặc tính thực sự có thể là bất cứ điều gì đó trợ giúp liên kết một tài liệu với một nhãn và có thể được tích hợp vào thuật toán.

Lĩnh vực học máy lớn và mạnh mẽ. Thay vì tập trung nhiều hơn vào lý thuyết, mà nó không thể thực hiện hoàn toàn thích hợp ở đây, tôi sẽ tiến lên và đi sâu nghiên cứu Mahout và cách sử dụng của nó.

Giới thiệu về Mahout

Apache Mahout là một dự án mã nguồn mở mới của Apache Software Foundation (ASF-Quỹ phần mềm Apache) với mục tiêu chính là tạo các thuật toán học máy có khả năng mở rộng, các thuật toán này là miễn phí sử dụng theo giấy phép Apache. Dự án này đang bước vào năm thứ hai của mình, với bản phát hành công khai trong phạm vi của nó. Mahout bao gồm các việc thực hiện để phân cụm, phân loại, CF và lập trình tiến hóa. Hơn nữa, nó khôn khéo sử dụng thư viện Apache Hadoop để cho phép Mahout mở rộng hiệu quả trong đám mây này (xem Tài nguyên).

Lịch sử Mahout

Dự án Mahout được bắt đầu bởi một số người tham gia vào cộng đồng Lucene Apache (tìm kiếm mã nguồn mở) với một sự quan tâm tích cực trong việc học máy và mong muốn về thực hiện mạnh mẽ, có đầy đủ các tài liệu cần thiết, có khả năng mở rộng của các thuật toán học máy phổ biến cho việc phân cụm và phân loại. Cộng đồng này ban đầu được báo "Map-Reduce for Machine Learning on Multicore" (Map-Reduce cho học máy theo đa lõi) của Ng và cộng sự (xem Tài nguyên) nhưng đã phát triển để trình bày các cách tiếp cận học máy rộng hơn. Mahout cũng nhằm mục đích:

Tên có nghĩa gì?

Mahout (người quản tượng) là một người nuôi và điều khiển một con voi. Tên Mahout xuất phát từ việc sử dụng (trước đây) dự án của Apache Hadoop — trong đó con voi màu vàng là biểu tượng của nó — với khả năng mở rộng và có khả năng chịu lỗi.

- Xây dựng và hỗ trợ một cộng đồng những người dùng và những người đóng góp sao cho mã này vượt trên bất kỳ tác động nào của người đóng góp cụ thể, bất kỳ công ty, hoặc quỹ tài trợ đại học.
- Tập trung vào trường hợp sử dụng thực tế, thế giới thực, đối lập với nghiên cứu hay các kĩ thuật mới.

- Cung cấp các tài liệu và ví dụ có chất lượng.

Các đặc tính

Mặc dù tương đối mới trong thuật ngữ mã nguồn mở, Mahout đã có một số lượng lớn các chức năng, đặc biệt liên quan đến việc phân cụm và CF. Các đặc tính chính của Mahout là:

- Taste CF. Taste là một dự án mã nguồn mở cho CF được khởi đầu bởi Sean Owen trên SourceForge và được tặng cho Mahout vào năm 2008.
- Một số việc thực hiện phân cụm của Map-Reduce có sẵn, gồm k-Means, fuzzy k-Means, Canopy, Dirichlet và Mean-Shift.
- Các việc thực hiện phân loại Naive Bayes phân tán và Naive Bayes phụ.
- Các khả năng của hàm phù hợp phân tán đối với lập trình tiến hóa.
- Các thư viện ma trận và vectơ.
- Các ví dụ về tất cả các thuật toán ở trên.

Một vài lời về Map-Reduce

Map-Reduce là một API lập trình phân tán được Google phát minh ra và được thực hiện trong dự án Apache Hadoop. Được kết hợp với một hệ thống tệp phân tán, thường nó làm cho các vấn đề song song hóa trở nên dễ dàng hơn bằng cách cung cấp cho các lập trình viên một API rõ ràng để mô tả công việc tính toán song song. (Xem phần Tài nguyên để biết thêm thông tin.)

Bắt đầu với Mahout

Bắt đầu và thực hiện với Mahout là tương đối đơn giản. Để khởi động, bạn cần phải cài đặt các điều kiện cần trước sau đây:

- JDK 1.6 hoặc cao hơn
- Ant 1.7 hoặc cao hơn
- Maven 2.0.9 hoặc 2.0.10, nếu bạn muốn xây dựng mã nguồn Mahout

Bạn cũng cần mã mẫu của bài viết này (xem phần Tải về), trong đó có một bản sao của Mahout và các phụ thuộc của nó. Hãy thực hiện theo các bước sau để cài đặt mã mẫu:

1. `unzip sample.zip`
2. `cd apache-mahout-examples`
3. `ant install`

Bước 3 tải về các tệp Wikipedia cần thiết và biên dịch mã. Các tệp Wikipedia thường dùng xấp xỉ 2,5GB, vì thế thời gian để tải về sẽ phụ thuộc vào băng thông của bạn.

Xây dựng một máy bình luận

Mahout thiện đang cung cấp các công cụ để xây dựng một máy bình luận thông qua các thư viện Taste — một máy nhanh và linh hoạt cho CF. Taste hỗ trợ bình luận dựa vào người dùng và dựa vào mục và đi kèm với nhiều sự lựa chọn xây dựng các bình luận, cũng như các giao diện cho bạn để định nghĩa riêng của bạn. Taste gồm năm thành phần chính để làm việc với `User` (những người dùng), `Item` (các mục) và `Preference` (các sở thích):

- `DataModel`: Lưu trữ cho các `User`, các `Item`, và các `Preference`.
- `UserSimilarity`: Giao diện định nghĩa sự tương đương giữa hai người dùng.
- `ItemSimilarity`: Giao diện định nghĩa sự tương đương giữa hai mục.
- `Recommender`: Giao diện để cung cấp các bình luận.
- `UserNeighborhood`: Giao diện để tính toán một vùng lân cận của những người dùng tương tự có thể được `Recommender` (Những người bình luận) sử dụng.

Các thành phần này và các việc thực hiện của chúng giúp cho nó có thể xây dựng các hệ thống bình luận phức tạp cho

hoặc các bình luận dựa trên thời gian thực hoặc các bình luận ngoại tuyến (offline). Các bình luận dựa trên thời gian thực thường có thể làm việc với vài nghìn người dùng, trong khi các bình luận ngoại tuyến có thể mở rộng lớn hơn nhiều. Taste thậm chí đi kèm với các công cụ có sử dụng Hadoop để tính toán các bình luận ngoại tuyến. Trong nhiều trường hợp, đây là một tiếp cận hợp lý để cho phép bạn đáp ứng các yêu cầu của một hệ thống lớn có rất nhiều người dùng, các mục và các sở thích.

Để giải thích việc xây dựng một hệ thống đơn giản, tôi cần một số người dùng, các mục, và các sự xếp loại. Với mục đích này, tôi ngẫu nhiên đã tạo một tập lớn các `User` và các `Preference` cho các tài liệu Wikipedia (các `Item` theo cách nói Taste) khi sử dụng mã trong `cf.wikipedia.GenerateRatings` (có trong mã nguồn với mã mẫu) rồi bổ sung điều này bằng một tập các sự xếp loại thủ công xung quanh một chủ đề cụ thể (Abraham Lincoln) để tạo tệp recommendations.txt cuối cùng có trong ví dụ này. Ý tưởng đằng sau cách tiếp cận này là chỉ ra cách CF có thể hướng dẫn những người hâm mộ một chủ đề cụ thể tới các tài liệu quan tâm khác trong chủ đề đó. Trong ví dụ, dữ liệu là 990 (có nhãn 0-989) người dùng ngẫu nhiên, những người đã chỉ định các sự xếp loại ngẫu nhiên cho tất cả các bài viết trong bộ sưu tập và 10 người dùng (được ghi nhãn 990-999), những người đã đánh giá một hoặc nhiều bài trong số 17 bài viết trong bộ sưu tập có chứa cụm từ *Abraham Lincoln*.

Để bắt đầu, tôi sẽ giải thích cách tạo các bình luận cho một người dùng dựa vào tập xếp loại trong tệp recommendations.txt. Như là trường hợp với hầu hết các lần sử dụng Taste, bước đầu tiên là nạp dữ liệu có chứa các bình luận và lưu trữ nó trong một `DataModel`. Taste đi kèm với một số việc thực hiện khác nhau của `DataModel` để làm việc với các tệp và các cơ sở dữ liệu. Với ví dụ này, tôi sẽ giữ cho tình hình đơn giản và sử dụng lớp `FileDataModel`, nó dự kiến mỗi dòng phải có dạng: mã nhận dạng (ID) của người dùng, ID của mục, sở thích — ở đây cả ID người dùng lẫn ID mục là các chuỗi, trong khi sở thích có thể là một chuỗi kép. Căn cứ vào một mô hình, tôi cần phải cho Taste biết cách nó nên so sánh những người dùng bằng cách khai báo một việc thực hiện `UserSimilarity`. Tùy thuộc vào việc thực hiện `UserSimilarity` được sử dụng, bạn cũng có thể phải nói cho cho Taste cách suy ra các sở thích nếu thiếu một giá trị cài đặt rõ ràng cho người dùng. Liệt kê 1 đặt tất cả những từ này vào trong mã (`cf.wikipedia.WikipediaTasteUserDemo` trong mã mẫu có chứa liệt kê đầy đủ.)

Hãy coi chừng dữ liệu có sẵn!

Ví dụ được trình bày ở đây có chứa dữ liệu có sẵn hoàn toàn. Tôi đã tự thực hiện tất cả xếp loại, mô phỏng 10 người dùng thực tế, những người muốn có thông tin về Abraham Lincoln. Trong khi tôi tin rằng khái niệm đằng sau dữ liệu này là đáng quan tâm, thì chính dữ liệu và các giá trị được sử dụng lại không đáng tin. Nếu bạn muốn dữ liệu thực, tôi khuyên bạn nên kiểm tra dự án GroupLens tại Đại học Minnesota và tài liệu hướng dẫn Taste (xem Tài nguyên). Tôi đã chọn chỉnh sửa dữ liệu vì muốn sử dụng chỉ một tập dữ liệu xuyên suốt tất cả các ví dụ.

Liệt kê 1. Tạo mô hình và định nghĩa sự tương đương của người dùng

```
//create the data model
FileDataModel dataModel = new FileDataModel(new File(recsFile));
UserSimilarity userSimilarity = new PearsonCorrelationSimilarity(dataModel);
// Optional:
userSimilarity.setPreferenceInferrer(new AveragingPreferenceInferrer(dataModel));
```

Trong Liệt kê 1, tôi sử dụng `PearsonCorrelationSimilarity`, đánh giá sự tương quan giữa hai biến, nhưng cũng có các phép đo `UserSimilarity` khác. Sự lựa chọn của một phép đo sự tương đương phụ thuộc vào kiểu dữ liệu hiện tại và việc thử nghiệm của bạn. Đối với dữ liệu này, tôi đã tìm thấy sự kết hợp này để làm việc tốt nhất trong khi vẫn giải thích các vấn đề. Bạn sẽ tìm thấy nhiều thông tin hơn về việc chọn một phép đo sự tương đương tại trang Web Mahout (xem Tài nguyên).

Để hoàn thành ví dụ này, tôi xây dựng một `UserNeighborhood` và một `Recommender`. `UserNeighborhood` nhận biết những người dùng tương tự như người dùng của tôi và được giao quyền cho `Recommender`, nó sau đó thực hiện việc tạo một danh sách xếp loại các mục được bình luận. Liệt kê 2 thu hút được những ý tưởng trong mã:

Liệt kê 2. Tạo các bình luận

```
//Get a neighborhood of users
UserNeighborhood neighborhood =
        new NearestNUserNeighborhood(neighborhoodSize, userSimilarity, dataModel);
//Create the recommender
Recommender recommender =
        new GenericUserBasedRecommender(dataModel, neighborhood, userSimilarity);
User user = dataModel.getUser(userId);
System.out.println("-----");
```

```
System.out.println("User: " + user);
//Print out the users own preferences first
TasteUtils.printPreferences(user, handler.map);
//Get the top 5 recommendations
List<RecommendedItem> recommendations =
        recommender.recommend(userId, 5);
TasteUtils.printRecs(recommendations, handler.map);
```

Bạn có thể chạy ví dụ đầy đủ trên dòng lệnh bằng cách thực hiện lệnh `ant user-demo` trong thư mục chứa mẫu. Chạy lệnh này in các sở thích và các bình luận cho người dùng tưởng tượng 995, người ngẫu nhiên là một người hâm mộ của Lincoln. Liệt kê 3 cho thấy kết quả từ việc chạy lệnh `ant user-demo`:

Liệt kê 3. Kết quả từ bình luận của người dùng

```
 [echo] Getting similar items for user: 995 with a neighborhood of 5
     [java] 09/08/20 08:13:51 INFO file.FileDataModel: Creating FileDataModel
            for file src/main/resources/recommendations.txt
     [java] 09/08/20 08:13:51 INFO file.FileDataModel: Reading file info...
     [java] 09/08/20 08:13:51 INFO file.FileDataModel: Processed 100000 lines
     [java] 09/08/20 08:13:51 INFO file.FileDataModel: Read lines: 111901
     [java] Data Model: Users: 1000 Items: 2284
     [java] -----
     [java] User: 995
     [java] Title: August 21 Rating: 3.930000066757202
     [java] Title: April Rating: 2.203000068664551
     [java] Title: April 11 Rating: 4.230000019073486
     [java] Title: Battle of Gettysburg Rating: 5.0
     [java] Title: Abraham Lincoln Rating: 4.739999771118164
     [java] Title: History of The Church of Jesus Christ of Latter-day Saints
              Rating: 3.430000066757202
     [java] Title: Boston Corbett Rating: 2.009999990463257
     [java] Title: Atlanta, Georgia Rating: 4.429999828338623
     [java] Recommendations:
     [java] Doc Id: 50575 Title: April 10 Score: 4.98
     [java] Doc Id: 134101348 Title: April 26 Score: 4.860541
     [java] Doc Id: 133445748 Title: Folklore of the United States Score: 4.4308662
     [java] Doc Id: 1193764 Title: Brigham Young Score: 4.404066
     [java] Doc Id: 2417937 Title: Andrew Johnson Score: 4.24178
```

Từ các kết quả trong Liệt kê 3, bạn có thể thấy rằng hệ thống đã bình luận một số bài viết với các mức tin tưởng khác nhau. Trong thực tế, mỗi một mục này đã được những người hâm mộ Lincoln đánh giá, nhưng không phải do người dùng 995. Nếu bạn muốn xem kết quả của những người dùng khác, chỉ cần dùng tham số `-Duser.id=`*`USER-ID`* trên dòng lệnh, ở đây *`USER-ID`* (mã nhận dạng người dùng) là một số nằm giữa `0` và `999`. Bạn cũng có thể thay đổi quy mô của vùng lân cận bằng cách dùng tham số `-Dneighbor.size=`*`X`*, ở đây *`X`* là số nguyên lớn hơn 0. Trong thực tế, việc thay đổi quy mô vùng lân cận tới `10` đưa ra kết quả rất khác nhau, kết quả này chịu ảnh hưởng bởi thực tế là một trong những người dùng ngẫu nhiên ở trong vùng lân cận. Để xem các vùng lân cận của người dùng và các mục nói chung, hãy thêm `-Dcommon=true` vào dòng lệnh.

Bây giờ, nếu bạn đã tình cờ nhập một số không nằm trong phạm vi của người dùng, bạn có thể thấy rằng ví dụ này để lộ ra một `NoSuchUserException`. Thật vậy, ứng dụng của bạn sẽ cần biết làm gì khi một người dùng mới nhập vào hệ thống. Ví dụ, bạn chỉ có thể hiển thị 10 bài viết phổ biến nhất, một sự lựa chọn ngẫu nhiên các bài báo, hoặc một sự lựa chọn một của các bài viết "khác nhau" — hoặc với vấn đề đó, chẳng có gì cả.

Như tôi đã đề cập trước đó, cách tiếp cận dựa trên người dùng thường không mở rộng được. Trong trường hợp này, tốt hơn là sử dụng một cách tiếp cận dựa trên mục-mục. Rất may, Taste giúp cho việc sử dụng tiếp cận dựa trên mục-mục cũng dễ dàng. Mã cơ bản để tạo và chạy với sự tương đương của mục-mục không khác nhau nhiều, như bạn có thể thấy trong Liệt kê 4:

Liệt kê 4. Ví dụ về sự tương đương mục-mục (từ `cf.wikipedia.WikipediaTasteItemItemDemo`)

```
//create the data model
FileDataModel dataModel = new FileDataModel(new File(recsFile));
//Create an ItemSimilarity
ItemSimilarity itemSimilarity = new LogLikelihoodSimilarity(dataModel);
//Create an Item Based Recommender
ItemBasedRecommender recommender =
        new GenericItemBasedRecommender(dataModel, itemSimilarity);
//Get the recommendations
List<RecommendedItem> recommendations =
        recommender.recommend(userId, 5);
```

```
TasteUtils.printRecs(recommendations, handler.map);
```

Cũng như trong Liệt kê 1, tôi tạo một `DataModel` từ tệp các bình luận, nhưng cùng lúc này, thay cho việc tạo phiên bản một cá thể `UserSimilarity`, tôi tạo một `ItemSimilarity` khi sử dụng `LogLikelihoodSimilarity`, để giúp xử lý các sự kiện hiếm thấy. Sau đó, tôi đưa `ItemSimilarity` tới một `ItemBasedRecommender` và yêu cầu bình luận. Vậy đó! Bạn có thể chạy cái này trong mã mẫu thông qua lệnh `ant item-demo`. Từ đây, tất nhiên, bạn muốn thiết lập hệ thống của bạn để thực hiện các tính toán này ngoại tuyến và bạn cũng có thể nghiên cứu các phép đo `ItemSimilarity` khác. Lưu ý rằng, do tính ngẫu nhiên của các dữ liệu trong ví dụ này, các bình luận có thể không được như mong đợi. Trong thực tế, điều quan trọng là chắc chắn rằng bạn đánh giá kết quả của mình trong thời gian thử nghiệm và thử nghiệm các số liệu thông kê của sự tương đương khác nữa, do nhiều số liệu thống kê chung có trường hợp cạnh ngoài nào đó, nên trường hợp này ngừng hoạt động khi dữ liệu có sẵn không đủ để đưa ra các bình luận phù hợp.

Quay lại ví dụ người dùng mới, vấn đề phải làm gì nếu thiếu các sở thích người dùng đã trở nên dễ dàng tiếp cận hơn rất nhiều một khi người dùng chuyển hướng đến một mục. Trong trường hợp đó, bạn có thể tận dụng lợi thế của các tính toán mục-mục và yêu cầu các `ItemBasedRecommender` cho các mục hầu giống với mục hiện hành. Liệt kê 5 chứng tỏ điều này trong mã:

Liệt kê 5.Chương trình trình diễn các mục tương đương (từ `cf.wikipedia.WikipediaTasteItemRecDemo`)

```
//create the data model
FileDataModel dataModel = new FileDataModel(new File(recsFile));
//Create an ItemSimilarity
ItemSimilarity itemSimilarity = new LogLikelihoodSimilarity(dataModel);
//Create an Item Based Recommender
ItemBasedRecommender recommender =
        new GenericItemBasedRecommender(dataModel, itemSimilarity);
//Get the recommendations for the Item
List<RecommendedItem> simItems
        = recommender.mostSimilarItems(itemId, numRecs);
TasteUtils.printRecs(simItems, handler.map);
```

Bạn có thể chạy Liệt kê 5 từ dòng lệnh bằng cách thực hiện lệnh `ant sim-item-demo`. Sự khác biệt thực sự duy nhất với Liệt kê 4 là Liệt kê 5, thay vì yêu cầu các bình luận, lại yêu cầu các mục hoàn toàn giống với mục đầu vào.

Từ đây, bạn cần phải có đủ các thứ để bắt đầu với Taste. Để tìm hiểu thêm, hãy tham khảo tài liệu hướng dẫn Taste và danh sách gửi thư mahout-user@lucene.apache.org (xem Tài nguyên). Tiếp theo, tôi sẽ xem xét cách tìm các bài viết tương tự bằng cách sử dụng một số khả năng của phân cụm Mahout.

Phân cụm với Mahout

Mahout hỗ trợ một số việc thực hiện thuật toán phân cụm, tất cả đều được viết bằng Map-Reduce, mỗi lần thực hiện có tập riêng của nó về các mục tiêu và tiêu chuẩn:

- **Canopy**: Một thuật toán phân cụm nhanh thường được sử dụng để tạo ra các phôi ban đầu cho các thuật toán phân cụm khác.
- **k-Means** (và **fuzzy k-Means**): Phân cụm các mục thành k nhóm dựa trên khoảng cách các mục từ trọng tâm hoặc trung tâm của lần lặp lại trước.
- **Mean-Shift**: Thuật toán không yêu cầu bất kỳ kiến thức *cần có trước* về số lượng các nhóm và có thể tạo ra các nhóm có dạng tùy ý.
- **Dirichlet**: Phân cụm dựa trên sự pha trộn của nhiều mô hình xác suất cung cấp cho nó lợi thế là không cần phải ghi nhớ trước một khung nhìn cụ thể của các nhóm.

Theo quan điểm thực tế, các tên và các lần thực hiện không quan trọng như các kết quả mà chúng tạo nên. Với ý nghĩ đó, tôi sẽ hiển thị cách k-Means làm việc và để lại những cái khác cho bạn khám phá. Hãy ghi nhớ rằng mỗi thuật toán

có các yêu cầu riêng của mình để làm cho nó hoạt động hiệu quả.

Theo phác thảo (có nhiều chi tiết hơn để tiếp tục), các bước có liên quan trong việc phân cụm dữ liệu khi sử dụng Mahout là:

1. Chuẩn bị đầu vào. Nếu phân cụm văn bản, bạn cần chuyển đổi biểu diễn văn bản thành biểu diễn số.
2. Chạy thuật toán lựa chọn phân cụm nhờ sử dụng một trong nhiều trình điều khiển Hadoop đã có sẵn trong Mahout.
3. Đánh giá các kết quả.
4. Lặp lại nếu cần thiết.

Đầu tiên và trước hết, các thuật toán phân cụm yêu cầu dữ liệu theo một định dạng phù hợp để xử lý. Trong việc học máy, dữ liệu thường được biểu diễn như là *véc tơ*, đôi khi được gọi là một *vectơ đặc tính*. Trong việc phân cụm, một vectơ là một mảng của các trọng số biểu diễn dữ liệu. Tôi sẽ giải thích việc phân cụm bằng cách sử dụng các vectơ được tạo từ các tài liệu Wikipedia, nhưng các vectơ có thể đến từ các lĩnh vực khác, chẳng hạn như dữ liệu cảm biến hoặc hồ sơ người dùng. Mahout đi kèm với hai biểu diễn `Vector`: `DenseVector` và `SparseVector`. Tùy thuộc vào dữ liệu của bạn, bạn sẽ cần phải chọn việc thực hiện phù hợp để đạt được hiệu năng tốt. Nói chung, các vấn đề dựa trên văn bản thưa thớt, làm cho `SparseVector` là sự lựa chọn đúng đối với chúng. Mặt khác, nếu hầu hết các giá trị cho hầu hết các vectơ là khác không, thì một `DenseVector` là thích hợp hơn. Nếu bạn không chắc chắn, hãy thử cả hai và nhận thấy cái nào hoạt động nhanh hơn trên một tập con của dữ liệu của bạn.

Để tạo ra các vectơ từ nội dung Wikipedia (mà tôi đã làm cho bạn):

1. Lập chỉ mục nội dung trong Lucene, chắc chắn lưu trữ các vectơ kỳ hạn cho lĩnh vực mà bạn muốn tạo các vectơ từ đó. Tôi sẽ không trình bày các chi tiết về việc này — nó nằm ngoài phạm vi của bài viết — nhưng tôi sẽ cung cấp một số gợi ý ngắn gọn cùng với một số tài liệu tham khảo trên Lucene. Lucene đi kèm với một lớp gọi là `EnWikiDocMaker` (trong gói `contrib/benchmark`), có thể đọc trong một kho chứa tạm thời tệp Wikipedia và đưa ra các tài liệu để lập chỉ mục trong Lucene.

2. Tạo vectơ từ chỉ mục Lucene bằng cách sử dụng lớp `org.apache.mahout.utils.vectors.lucene.Driver` có trong mô đun `utils` của Mahout. Chương trình điều khiển này đi kèm với rất nhiều tuỳ chọn để tạo các vectơ. Trang wiki Mahout có tên Creating Vectors (Tạo các Vectơ) từ Text (Văn bản) có nhiều thông tin (xem Tài nguyên).

Các kết quả của việc chạy hai bước này là một tệp giống như tệp n2.tar.gz mà bạn đã tải xuống trong phần Bắt đầu với Mahout. Để có đầy đủ, tệp n2.tar.gz gồm các vectơ được tạo từ việc lập chỉ mục của tất cả các tài liệu trong tệp "các khối dữ liệu" của Wikipedia đã được tự động tải về bằng cách phương thức `ant install` ở trên. Các vectơ đã được chuẩn hoá nhờ sử dụng chuẩn Ơ-clit (hoặc chuẩn L^2; xem Tài nguyên). Khi sử dụng Mahout, bạn có thể sẽ muốn thử tạo các vectơ theo một số cách để xem cách nào mang lại các kết quả tốt nhất.

Dựa vào một tập các vectơ, bước tiếp theo là chạy thuật toán phân cụm k-Means. Mahout cung cấp các chương trình điều khiển cho tất cả các thuật toán phân cụm, gồm cả thuật toán k-Means, có tên thích hợp là `KMeansDriver`. Trình điều khiển dễ sử dụng khi là một một chương trình độc lập không có Hadoop, như đã giải thích bằng cách chạy `ant k-means`. Xin cứ tự nhiên kiểm tra đích Ant k-means trong build.xml để có thêm thông tin các đối số mà `KMeansDriver` chấp nhận. Sau khi quá trình này hoàn tất, bạn có thể in ra các kết quả bằng cách dùng lệnh `ant dump`.

Sau khi bạn đã chạy thành công trong chế độ độc lập, bạn có thể tiến hành chạy ở chế độ phân tán trên Hadoop. Để làm như vậy, bạn cần có Mahout Job JAR, nằm trong thư mục hadoop trong mã mẫu. Một Job Jar đóng gói tất cả các mã và các phụ thuộc vào trong chỉ một tệp JAR để dễ dàng nạp vào Hadoop. Bạn cũng sẽ cần phải tải về Hadoop 0.20 và làm theo các chỉ dẫn trên hướng dẫn Hadoop để chạy lần đầu tiên trong chế độ phân tán giả (có nghĩa là từng cụm một) và sau đó là chế độ phân tán đầy đủ. Để biết thêm thông tin, xem trang Web Hadoop và Tài nguyên, cũng như các tài nguyên điện toán đám mây của IBM (xem Tài nguyên).

Đánh giá các kết quả của bạn

Có nhiều cách tiếp cận để đánh giá kết quả phân cụm của bạn. Nhiều người bắt đầu chỉ đơn giản bằng cách sử dụng kiểm tra thủ công và thử nghiệm đặc biệt. Tuy nhiên, để thực sự hài lòng, điều cần thiết là sử dụng các kỹ thuật đánh giá sâu hơn như là việc phát triển một tiêu chuẩn vàng khi dùng một số người am hiểu kỹ thuật. Để tìm hiểu thêm về việc đánh giá kết quả của bạn, xem Tài nguyên. Với các ví dụ của tôi, tôi đã sử dụng kiểm tra thủ công để xem một số kết quả đã được cụm với nhau thực sự có ý nghĩa không. Nếu tôi được đưa điều này vào trong sản xuất, tôi sẽ sử dụng một quy trình nghiêm ngặt hơn nhiều.

Phân loại nội dung với Mahout

Mahout hiện đang hỗ trợ hai phương pháp tiếp cận liên quan đến việc phân loại/sắp xếp có hệ thống nội dung dựa trên số liệu thống kê Bayesian. Cách tiếp cận đầu tiên là một trình phân loại Naive Bayes kích hoạt Map-Reduce đơn giản. Các trình phân loại Naive Bayes được biết là nhanh và khá chính xác, mặc dù các giả định rất đơn giản (và thường không chính xác) của chúng về các dữ liệu hoàn toàn độc lập. Các trình phân loại Naive Bayes thường không hoạt động khi kích thước của các ví dụ huấn luyện cho mỗi lớp không được cân bằng hoặc khi dữ liệu đủ độc lập. Phương pháp thứ hai, được gọi là Naive Bayes phụ, cố gắng chỉnh sửa một số vấn đề với cách tiếp cận Naive Bayes trong khi vẫn duy trì tính đơn giản và tốc độ của nó. Tuy nhiên, với bài viết này, tôi sẽ chỉ hiển thị cách tiếp cận Naive Bayes, bởi vì nó giải thích được vấn đề tổng thể và các đầu vào trong Mahout.

Tóm lại, một trình phân loại Naive Bayes là một quá trình hai phần gồm việc theo dõi các đặc tính (các từ) có liên kết với một tài liệu và loại riêng và sau đó sử dụng thông tin đó để dự đoán loại nội dung mới, vô hình. Bước đầu tiên, được gọi là *huấn luyện*, tạo một mô hình bằng cách xem các ví dụ đã được phân loại về nội dung và sau đó theo dõi các khả năng mà mỗi từ được kết hợp với một phần nội dung cụ thể. Bước thứ hai, được gọi là *phân loại*, sử dụng mô hình được tạo trong huấn luyện và nội dung của một tài liệu mới, vô hình, cùng với Định lý Bayes, để dự đoán loại tài liệu được thông qua. Vì vậy, để chạy trình phân loại Mahout, đầu tiên bạn cần phải huấn luyện theo mô hình và sau đó sử dụng mô hình đó để phân loại nội dung mới. Phần tiếp theo sẽ giải thích cách thực hiện điều này bằng cách sử dụng tập dữ liệu Wikipedia.

Chạy trình phân loại Naive Bayes

Trước khi bạn có thể chạy các trình huấn luyện và trình phân loại, bạn cần thực hiện một công việc chuẩn bị nhỏ để thiết lập một bộ các tài liệu về huấn luyện và một bộ các tài liệu về thử nghiệm. Bạn có thể chuẩn bị các tệp Wikipedia (từ những thứ bạn đã tải về thông qua đích `install` bằng cách chạy `ant prepare-docs`. Việc này chia tách các tệp đầu vào Wikipedia khi sử dụng lớp `WikipediaDatasetCreatorDriver` có trong các ví dụ Mahout. Các tài liệu được chia tách dựa trên tài liệu có một loại giống với một trong các loại quan tâm hay không. Các loại quan tâm có thể là bất cứ loại Wikipedia hợp lệ nào (hoặc thậm chí là chuỗi con bất kỳ của một loại Wikipedia). Chẳng hạn, trong ví dụ này, tôi có hai loại: Khoa học và Lịch sử. Vì vậy, bất kỳ loại Wikipedia nào có một loại chứa từ *khoa học* hoặc *lịch sử* (đó không phải là một sự tương đương chính xác) sẽ được đặt vào một thùng có các tài liệu khác cho loại đó. Ngoài ra, mỗi tài liệu được gắn thẻ (tokenized) và được chuẩn hóa để loại bỏ dấu chấm câu, đánh dấu Wikipedia và các đặc tính khác mà không cần thiết cho công việc này. Các kết quả cuối cùng sẽ được lưu trong một tệp có ghi nhãn với tên loại đó, một tài liệu cho mỗi dòng, đó là định dạng đầu vào mà Mahout trông đợi. Cũng vậy, việc chạy `ant prepare-test-docs` thực hiện cùng công việc với các tài liệu thử nghiệm. Điều quan trọng là các tài liệu thử nghiệm và huấn luyện không trùng nhau, có thể nghiêng về các kết quả. Theo lý thuyết, việc sử dụng các tài liệu huấn luyện để thử nghiệm kết quả trong các kết quả hoàn hảo, nhưng ngay cả điều này không có khả năng trong thực hành.

Sau khi các bộ huấn luyện và thử nghiệm được thiết lập, bây giờ là lúc để chạy các lớp `TrainClassifier` thông qua đích `ant train`. Điều này sẽ cần một lượng lớn lần đăng nhập từ cả hai Mahout và Hadoop. Sau khi hoàn thành, `ant test` chọn các tài liệu thử nghiệm mẫu và cố gắng phân loại chúng bằng cách sử dụng mô hình đã được xây dựng trong lúc huấn luyện. Kết quả từ một thử nghiệm như vậy trong Mahout là một cấu trúc dữ liệu gọi là một ma trận nhầm lẫn (*confusion matrix*). Một ma trận nhầm lẫn mô tả có bao nhiêu kết quả đã được phân loại đúng và có bao nhiêu được phân loại không đúng cho mỗi loại.

Tóm lại, bạn chạy các bước sau để tạo ra kết quả phân loại:

1. `ant prepare-docs`
2. `ant prepare-test-docs`
3. `ant train`
4. `ant test`

Chạy tất cả lệnh này (`classifier-example` của đích Ant bắt giữ tất cả chúng trong một cuộc gọi), đưa ra ma trận tóm tắt và ma trận nhầm lẫn được hiển thị trong Liệt kê 6:

Liệt kê 6.Các kết quả từ việc chạy trình phân loại Bayes cho lịch sử và khoa học

```
[java] 09/07/22 18:10:45 INFO bayes.TestClassifier: history
```

```
                                95.458984375    3910/4096.0
[java] 09/07/22 18:10:46 INFO bayes.TestClassifier: science
                                15.554072096128172      233/1498.0
[java] 09/07/22 18:10:46 INFO bayes.TestClassifier: ================
[java] Summary
[java] -------------------------------------------------------
[java] Correctly Classified Instances          :       4143
                                                     74.0615%
[java] Incorrectly Classified Instances        :       1451
                                                     25.9385%
[java] Total Classified Instances              :       5594
[java]
[java] =======================================================
[java] Confusion Matrix
[java] -------------------------------------------------------
[java] a           b       <--Classified as
[java] 3910        186      |  4096        a     = history
[java] 1265        233      |  1498        b     = science
[java] Default Category: unknown: 2
```

Các kết quả của quá trình trung gian được lưu trữ trong thư mục có tên là wikipedia trong thư mục cơ bản.

Với một tập kết quả trong tay, câu hỏi rõ ràng là: "Tôi đã thực hiện như thế nào?" Bản tóm tắt nói rõ rằng đại khái tôi đã nhận được khoảng 75 phần trăm đúng và 25 phần trăm không đúng. Thoạt nhìn điều này có vẻ khá hợp lý, đặc biệt là bởi vì nó có nghĩa là tôi đã làm tốt hơn so với cách dự đoán ngẫu nhiên. Tuy nhiên, việc kiểm tra gần đây cho thấy rằng tôi đã thực sự làm tốt khi dự báo về lịch sử (khoảng 95 phần trăm đúng) và thực sự kém lúc dự báo về khoa học (khoảng 15 phần trăm). Trong việc tìm kiếm lý do tại sao lại như vậy, việc xem xét nhanh các tệp đầu vào cho huấn luyện cho thấy rằng tôi có nhiều ví dụ về lịch sử hơn so với khoa học (kích thước tệp là gần gấp đôi), đó có lẽ là một vấn đề tiềm năng.

Với thử nghiệm này, bạn có thể thêm tùy chọn `-Dverbose=true` vào `ant test`, cho phép đưa ra thông tin về mỗi đầu vào thử nghiệm và nó đã được ghi nhãn đúng hay không. Nghiên cứu kỹ kết quả này, bạn có thể tra cứu tài liệu và xem xét lý do nó đã được phân loại không đúng. Tôi cũng có thể thử các tham số đầu vào khác nhau và cũng có nhiều dữ liệu khoa học và huấn luyện lại mô hình này để xem tôi có thể cải thiện kết quả này không.

Điều cũng quan trọng là suy nghĩ về lựa chọn đặc tính để huấn luyện mô hình. Đối với các ví dụ này, tôi đã sử dụng từ `WikipediaTokenizer` của Apache Lucene để gắn thẻ các tài liệu ban đầu, nhưng tôi đã không cố gắng nhiều để loại bỏ các thuật ngữ phổ biến hoặc các thuật ngữ tạp nham mà có thể đã bị gắn thẻ sai. Nếu tôi đang xem xét để đưa trình phân loại này vào sản xuất, tôi sẽ kiểm tra kĩ hơn các đầu vào và các giá trị cài đặt khác, cố gắng tăng thêm một chút về hiệu năng.

Chỉ cần để xem các kết quả khoa học đã có một sự may mắn không, tôi đã thử một tập các loại khác nhau: Đảng Cộng hòa và Đảng Dân chủ. Trong trường hợp này, tôi muốn dự báo có hay không có một tài liệu mới về Đảng Cộng hòa hay Dân chủ. Để cho phép bạn tự mình thử nghiệm, tôi tạo ra repubs-dems.txt in src/test/resources. Sau đó tôi chạy các bước phân loại thông qua:

```
ant classifier-example -Dcategories.file=./src/test/resources/repubs-dems.txt -Dcat.dir=rd
```

Hai giá trị `-D` đơn giản trỏ tới tệp và tên loại của thư mục để đặt các kết quả trung gian trong thư mục wikipedia. Ma trận tóm tắt và ma trận nhầm lẫn từ hoạt động này trông giống như Liệt kê 7:

Liệt kê 7. Các kết quả từ việc chạy trình phân loại Bayes cho Đảng Cộng hòa và Đảng Dân chủ

```
 [java] 09/07/23 17:06:38 INFO bayes.TestClassifier: --------------
 [java] 09/07/23 17:06:38 INFO bayes.TestClassifier: Testing:
                             wikipedia/rd/prepared-test/democrats.txt
 [java] 09/07/23 17:06:38 INFO bayes.TestClassifier: democrats      70.0
                                                                    21/30.0
 [java] 09/07/23 17:06:38 INFO bayes.TestClassifier: --------------
 [java] 09/07/23 17:06:38 INFO bayes.TestClassifier: Testing:
                            wikipedia/rd/prepared-test/republicans.txt
 [java] 09/07/23 17:06:38 INFO bayes.TestClassifier: republicans    81.3953488372093
                                                                    35/43.0
 [java] 09/07/23 17:06:38 INFO bayes.TestClassifier:
 [java] Summary
 [java] -------------------------------------------------------
 [java] Correctly Classified Instances          :         56           76.7123%
 [java] Incorrectly Classified Instances        :         17           23.2877%
 [java] Total Classified Instances              :         73
```

```
[java]
[java] =======================================================
[java] Confusion Matrix
[java] -------------------------------------------------------
[java] a           b       <--Classified as
[java] 21          9        |  30          a     = democrats
[java] 8           35       |  43          b     = republicans
[java] Default Category: unknown: 2
```

Mặc dù kết quả cuối cùng là về cùng dạng đúng, bạn có thể thấy rằng tôi đã làm một công việc tốt hơn về việc quyết định giữa hai loại. Một việc kiểm tra nhanh của thư mục wikipedia/rd/prepared chứa các tài liệu đầu vào cho thấy rằng hai tệp huấn luyện đã được cân bằng hơn nhiều về mặt các ví dụ huấn luyện. Việc kiểm tra cũng cho thấy tôi có một vài ví dụ tổng thể so với hoạt động lịch sử/khoa học, bởi vì mỗi tệp nhỏ hơn nhiều so với bộ huấn luyện lịch sử hoặc khoa học. Nhìn chung, kết quả ít nhất có vẻ cân bằng hơn. Các bộ huấn luyện lớn hơn có khả năng bù trừ sự khác biệt giữa đảng Cộng hòa và đảng Dân chủ, mặc dù nếu nó không bù trừ, điều đó có thể ngụ ý rằng một nhóm là tốt hơn lúc bám vào thông báo của nó trên Wikipedia — nhưng tôi sẽ để lại việc đó cho các học giả chính trị quyết định.

Bây giờ tôi đã chỉ ra cách chạy phân loại trong chế độ độc lập, các bước tiếp theo là chọn mã cho đám mây và chạy trên một nhóm Hadoop. Cũng như với mã phân lớp, bạn sẽ cần có Mahout Job JAR. Ngoài ra, tất cả các thuật toán mà tôi đã đề cập trước đó là Map-Reduce-sẵn sàng và chỉ nên làm việc khi chạy trong quá trình nộp Job (công việc) được nêu ra trong hướng dẫn Hadoop.

Có gì tiếp theo với Mahout?

Apache Mahout đã đi một chặng đường dài trong hơn một năm, với các khả năng đáng kể về phân cụm, phân loại và CF, nhưng nó cũng có rất nhiều khả năng để phát triển. Sắp xảy ra đến nơi là thực hiện Map-Reduce của vô số quyết định ngẫu nhiên cho việc phân loại, các quy tắc kết hợp, Latent Dirichlet Allocation để nhận biết các chủ đề trong các tài liệu và nhiều tùy chọn phân loại hơn sử dụng HBase và các tùy chọn lưu trữ có sẵn khác. Ngoài những lần thực hiện mới này, hãy tìm kiếm thêm các chương trình trình diễn, tài liệu hướng dẫn và sửa lỗi.

Cuối cùng, đúng như một người quản tượng thực sự sử dụng sức mạnh và khả năng của con voi, vì thế, Apache Mahout cũng có thể giúp bạn tận dụng sức mạnh và khả năng của con voi màu vàng, đó là Apache Hadoop. Tiếp theo khi có nhu cầu phân cụm, phân loại hoặc bình luận nội dung, đặc biệt là ở quy mô lớn, hãy xem Apache Mahout.

Lời cảm ơn

Đặc biệt cảm ơn các ủy viên hội đồng Mahout Ted Dunning và Sean Owen về việc xem lại và sự hiểu biết sâu sắc của họ về bài viết này.

Tải về

Mô tả	Tên	Kích thước	Phương thức tải
Sample code	j-mahout.zip	90MB	**HTTP**

→ Thông tin về phương thức tải

Tài nguyên

Học tập

- **Học máy**
  - Học máy: Trang Wikipedia chứa một số thông tin bắt đầu có ích cũng như nhiều tài liệu tham khảo tốt để

tìm hiểu thêm về học máy, gồm các cách tiếp cận như việc học có giám sát.

- *Programming Collective Intelligence* (Toby Segaran, O'Reilly, 2007): Cuốn sách này là một điểm khởi đầu tuyệt vời cho nhiều nhiệm vụ học máy.
- Trí tuệ nhân tạo | Học máy: Hãy tham dự lớp của Andrew Ng tại Stanford.
- Đánh giá về phân cụm: Tìm hiểu thêm về đánh giá việc phân cụm. Cũng xem thêm cuộc thảo luận về danh sách gửi thư của Mahout.
- Định lý Bayes: Tìm hiểu kỹ Định lý Bayes hoạt động như thế nào.
- L^p space: Hiểu các chỉ tiêu L^p.

- **Apache Mahout and Apache Lucene**
  - Trang chủ dự án Mahout: Khám phá tất cả những gì về Mahout.
  - "Map-Reduce cho việc học máy theo đa lõi": Đọc bài hướng dẫn trợ giúp khởi chạy Mahout.
  - "MapReduce: làm đơn giản việc xử lý trên các cụm lớn" (Google Research Publications): Đọc bản chính trên Map-Reduce.
  - Taste: Khám phá tài liệu hướng dẫn Taste.
  - Apache Lucene: Tìm hiểu thêm về Lucene.
  - Apache Lucene trên developerWorks: Khám phá Lucene trong những bài viết này.
  - Tạo các Vectơ từ văn bản: Đọc mục này trong Mahout Wiki để tìm hiểu thêm về việc chuyển đổi dữ liệu của bạn sang lớp `Vector` của Mahout.
  - Phân cụm dữ liệu của bạn: Xem trang Mahout Wiki này để tìm hiểu thêm về cách phân cụm dữ liệu của bạn.

- **Apache Hadoop:**
  - Apache Hadoop: Tìm hiểu thêm về Hadoop.
  - Hướng dẫn bắt đầu nhanh về Hadoop: Tìm hiểu cách để chạy một Hadoop Job.
  - HBase: Hiểu cơ sở dữ liệu Hadoop.
- Duyệt qua hiệu sách công nghệ với các sách về các chủ đề kỹ thuật này và khác.
- Điện toán đám mây: Truy cập vào vùng Điện toán đám mây của developerWorks.
- Vùng công nghệ Java của developerWorks: Tìm hàng trăm bài viết về mọi khía cạnh của lập trình Java.

Lấy sản phẩm và công nghệ

- Tải về Hadoop 0.20.0.
- Tải về một tập con của Wikipedia.
- Tải về một tập con của Wikipedia như là các vectơ.
- Nhận được dữ liệu đánh giá phim thực từ dự án GroupLens.

Thảo luận

- Tham gia vào cộng đồng Mahout tại mahout-user@lucene.apache.org.
- Dành tâm trí cho cộng đồng My developerWorks.

Đôi nét về tác giả

Grant Ingersoll là người sáng lập và thành viên của hội đồng kỹ thuật tại Lucid Imagination. Các mối quan tâm lập trình của Grant gồm phục hồi thông tin, học máy, phân loại văn bản và trích xuất. Grant là người đồng sáng lập của dự án học máy Apache Mahout, cũng như một ủy viên hội đồng và người phát ngôn cho cả dự án Lucene Apache và Apache Solr. Ông cũng là đồng tác giả của Taming Text (Cung cấp nhân công, sắp tới) gồm các công cụ mã nguồn mở để xử lý ngôn ngữ tự nhiên.