TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI BÁO CÁO CUỐI KÌ**

**MÔN XỬ LÍ DỮ LIỆU LỚN**

**SPARK STRUCTURED STREAMING**

**VÀ XỬ LÍ TÍNH TOÁN PHÂN TÁN**

*Người hướng dẫn*: **PGS TS. LÊ ANH CƯỜNG**

*Người thực hiện*: **NGUYỄN HOÀNG THANH TRÚC - 51800144**

**PHẠM NGỌC QUỐC - 51800230**

**NGUYỄN ĐỨC TÍN - 51800248**

**HỒ CHÍ MINH, NĂM 2022**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI BÁO CÁO CUỐI KÌ**

**MÔN XỬ LÍ DỮ LIỆU LỚN**

**SPARK STRUCTURED STREAMING**

**VÀ XỬ LÍ TÍNH TOÁN PHÂN TÁN**

*Người hướng dẫn*: **PGS TS. LÊ ANH CƯỜNG**

*Người thực hiện*: **NGUYỄN HOÀNG THANH TRÚC - 51800144**

**PHẠM NGỌC QUỐC - 51800230**

**NGUYỄN ĐỨC TÍN - 51800248**

**HỒ CHÍ MINH, NĂM 2022**

LỜI CẢM ƠN

Chúng em xin cảm ơn thầy Lê Anh Cường đã tận tâm giảng dạy em trong suốt quá trình học tập môn Xử lí dữ liệu lớn. Em xin cảm ơn tất cả thầy cô trong khoa, cũng như các anh chị, các em đã nhiệt tình hỗ trợ em để em có thể hoàn thành tốt phần tiểu luận của mình. Cảm ơn nhà trường đã tạo cơ hội cho chúng em được học tập và tiếp thu kiến thức bổ ích. Bài báo cáo này còn nhiều thiếu sót, mong nhận được góp ý của thầy cô. Em xin chân thành cảm ơn.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của TS Nguyễn Văn A. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Hoàng Thanh Trúc*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Bài báo cáo tìm hiểu và trình bày về Spark Structured Streaming. Xây dựng một ứng dụng để thể hiện việc sử dụng kiến trúc Spark Streaming.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc93181148)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc93181149)

[TÓM TẮT iv](#_Toc93181150)

[MỤC LỤC 1](#_Toc93181151)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4](#_Toc93181152)

[CHƯƠNG 1 – SPARK STRUCTURED STREAMING 5](#_Toc93181153)

[1.1 Spark Streaming 5](#_Toc93181154)

[1.2 Structured Streaming 6](#_Toc93181155)

[1.3 Kiến trúc và ưu điểm của Spark Streaming 7](#_Toc93181156)

[1.4 Dữ liệu được đưa vào Streaming 7](#_Toc93181157)

[1.5 Lưu trữ kết quả ra ngoài cấu trúc streaming 8](#_Toc93181158)

[1.6 Giới thiệu về Spark SQL 8](#_Toc93181159)

[1.7 Xử lý và thao tác dữ liệu Spark SQL 8](#_Toc93181160)

[1.8 Xử lý và thao tác dữ liệu Mllib 10](#_Toc93181161)

[1.8.1 Khái niệm 10](#_Toc93181162)

[1.8.2 Một số công cụ sử dụng Spark.Mllib 10](#_Toc93181163)

[1.8.3 Thuật toán Machine Learning 11](#_Toc93181164)

[1.8.4 Các phương thức thư viện Mllib cung cấp 12](#_Toc93181165)

[1.9Xử lý và thao tác dữ liệu Mllib 12](#_Toc93181166)

[1.9.1 Word2Vec 12](#_Toc93181167)

[1.9.2 StopWordsRemover 13](#_Toc93181168)

[1.9.3 RegexTokenizer 14](#_Toc93181169)

[1.9.4 StringIndexer 15](#_Toc93181170)

[1.9.5 VectorAssembler 17](#_Toc93181171)

[1.9.6 PCA 18](#_Toc93181172)

[1.9.7 CountVectorizer 18](#_Toc93181173)

[1.9.8 Binarizer 19](#_Toc93181174)

[CHƯƠNG 2 – XỬ LÍ TÍNH TOÁN PHÂN TÁN 20](#_Toc93181175)

[2.1 P 20](#_Toc93181176)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC CHỮ VIẾT TẮT**

Ws WebSocket

HTTP Hypertext Transfer Protocol

TCP Transmission Control Protocol

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1: Cấu trúc của WebSockets 6](#_Toc71488858)

[Hình 2: Work 7](#_Toc71488859)

[Hình 3: Khi client dùng thư viện socket.io mở kết nối đến server socket 9](#_Toc71488860)

[Hình 4: Các trình duyện được hỗ trợ 10](#_Toc71488861)

[Hình 5: Giao diện trang chủ 24](#_Toc71488862)

[Hình 6: Khung chat 25](#_Toc71488863)

[Hình 7: Thoát khung chat 25](#_Toc71488864)

[Hình 8: Thông báo 26](#_Toc71488865)

**DANH MỤC BẢNG**

[Table 1: Các trạng thái của websockets 11](#_Toc71488869)

[Table 2: Các sự kiện WebSocket 13](#_Toc71488870)

CHƯƠNG 1 – SPARK STRUCTURED STREAMING

1.1 Spark Streaming

Luồng dữ liệu (Data stream) là một chuỗi dữ liệu liên tục đến liên tục. Truyền trực tuyến chia dữ liệu đầu vào liên tục chảy thành các đơn vị rời rạc để xử lý thêm. Xử lý luồng là xử lý và phân tích dữ liệu phát trực tuyến có độ trễ thấp

Spark Streaming được coi là phiên bản cũ, được thêm vào Apache Spark năm 2013, xử lý stream dựa trên RDD. Nó là phần mở rộng của Spark có thể chống chịu lỗi caovà cho phép xử lý ***B***ig ***D***ata ***R***ealtime***.*** Spark Streaming đóng vai trò cung cấp nền tảng để đẩy dữ liệu vào các mô hình phân tích tức thời, tăng hiệu năng của mô hình.

***Spark Streaming cho phép:***

+ Scaling: dễ dàng scale lên hàng nghìn node.

+ Speed: Spark Streaming có khả năng giảm độ trễn xuống mức vài trăm milliseconds.

+ Fault Tolerance: Một tính toán được chia nhỏ thành các task con cho nhiều node. Khi một task bị lỗi, các node khác sẽ thay thế thực hiện việc tính toán đó và do task đã rất nhỏ nên việc thực hiện lại này sẽ nhanh chóng hơn cách tiếp cận trên.

+ Integration: DStream đại diện cho series các RDDs trong Spark. Vì vậy, bất cứ function nào của Apache Spark đều có thể dùng để xử dữ liệu.

+ Business Analysis: Có thể các thư viện MLlib, SQL, GraphX có thể được áp dụng để phân tích dữ liệu.



Hình 1: Spark Streaming

***Dữ liệu đưa vào Spark Streaming***: Spark Streaming nhận các luồng dữ liệu đầu vào trực tiếp và chia dữ liệu thành các batches nhỏ, sau đó được xử lý bởi công cụ Spark Engine để tạo ra luồng kết quả cuối cùng theo batch dữ liệu mới.



Hình 2: Cơ chế hoạt động

Spark Streaming cung cấp một luồng trừu tượng cấp cao được gọi là luồng tùy ý hoặc DStream, đại diện cho một luồng dữ liệu liên tục. DStream có thể được tạo từ các luồng dữ liệu đầu vào từ các nguồn như Kafka và Kinesis hoặc bằng cách áp dụng các hoạt động cấp cao trên các DStream khác. Bên trong, một DStream được biểu diễn dưới dạng một chuỗi các RDD .

1.2 Structured Streaming

Structured Streaming là phiên bản mới được xây dựng dựng trên công cụ Spark SQL. Nó cũng là một công cụ xử lý luồng có khả năng mở rộng và chịu lỗi, nhưng được xử lý stream dựa trên Dataset/DataFrame.

1.3 Kiến trúc và ưu điểm của Spark Streaming

Thay vì xử lý dữ liệu truyền trực tuyến một bản ghi tại một thời điểm, Spark Streaming phân loại dữ liệu thành các bacth vi mô nhỏ sub-second. Nói cách khác, bộ thu Spark Streaming chấp nhận dữ liệu song song và đệm nó vào bộ nhớ của các nút Spark’s workers. Sau đó, Spark được tối ưu hóa độ trễ chạy các tác vụ ngắn để xử lý các bacth và xuất kết quả cho các hệ thống khác.

Không giống như mô hình toán tử liên tục truyền thống, trong đó việc tính toán được phân bổ tĩnh cho một nút, các tác vụ Spark được giao động cho các công nhân trên cơ sở địa phương dữ liệu và các tài nguyên có sẵn. Điều này cho phép cân bằng tải tốt hơn và khôi phục lỗi nhanh hơn.

Mỗi batch dữ liệu là một tập dữ liệu phân tán có khả năng phục hồi (RDD) trong Spark, là phần trừu tượng cơ bản của tập dữ liệu có khả năng chịu lỗi trong Spark. Điều này cho phép dữ liệu phát trực tuyến được xử lý bằng bất kỳ mã hoặc thư viện Spark nào.

Kiến trúc này cho phép Spark Streaming đạt được các mục tiêu: Cân bằng tải động, khôi phục lỗi, đồng bộ phân tích hàng loạt, phát trực tuyến và tương tác; phân tích nâng cao như học máy và tương tác mở rộng đến SQL Spark. Các mô hình học máy được tạo ngoại tuyến với MLlib có thể áp dụng cho dữ liệu truyền trực tuyến. Khả năng xử lý dữ liệu hàng loạt và tận dụng động cơ Spark của Spark Streaming dẫn đến thông lượng gần như cao hơn cho các hệ thống phát trực tuyến khác. Spark Streaming có thể đạt được độ trễ thấp tới vài trăm mili giây.

1.4 Dữ liệu được đưa vào Streaming

Structured Streaming coi dữ liệu stream là một bảng (table) không giới hạn và dữ liệu liên tục được thêm vào từng dòng của bảng này. Luồng dữ liệu có thể được xử lý bằng APIS lõi của Spark, SQL DataFrames hoặc API học máy và có thể được lưu vào hệ thống tệp, HDFS, cơ sở dữ liệu hoặc bất kỳ nguồn dữ liệu nào cung cấp Hadoop OutputFormat.

1.5 Lưu trữ kết quả ra ngoài cấu trúc streaming

Dữ liệu của DStream đẩy ra các hệ thống bên ngoài như cơ sở dữ liệu hoặc hệ thống tệp sử dụng Output Operations (Hoạt động đầu ra). Vì các hệ thống bên ngoài sử dụng dữ liệu đã được chuyển đổi theo sự cho phép của các hoạt động đầu ra, chúng sẽ kích hoạt việc thực thi thực tế tất cả các phép biến đổi Dstream. Spark cung cấp một số hoạt động đầu ra. Hoạt động đầu ra phổ biến nhất là saveAsTextFiles, kết xuất đầu ra dưới dạng tệp văn bản. Spark cũng cung cấp "foreachRDD", cho phép xử lý đầu ra theo bất kỳ cách nào. Có thể đẩy nó vào hàng đợi Kafka, lưu nó vào cơ sở dữ liệu hoặc POST nó vào một dịch vụ web.

Các DStream như RDD thực thi một cách lazily bởi các hoạt động đầu ra. Cụ thể, dữ liệu nhận được được xử lý mạnh mẽ bởi các hành động RDD bên trong các hoạt động đầu ra của DStream. Theo mặc định, các hoạt động đầu ra thực hiện từng lần một. Và chúng thực thi theo thứ tự được xác định trong các ứng dụng Spark.

1.6 Giới thiệu về Spark SQL

Spark SQL là một modul của Spark trong việc xử lý dữ liệu có cấu trúc (dataset, database). Không gióng như cấu trúc cơ bản RDD của Spark, Spark SQL cung cấp cho Spark nhiều thông tin chi tiết về cấu trúc và phương pháp tính toán của dữ liệu diễn ra như thế nào. Thông tin này được sử dụng để tối ưu hóa các truy vấn SQL. Nó cũng có thể giúp ta nhiều thứ, chẳng hạn như:

- Cho phép có thể tải lên dữ liệu quan hệ từ “Parquet files” và “Hive tables”

- Truy vấn dữ liệu với dữ liệu cho trước và RDD có sẵn

- Cho phép xuất dữ liệu RDD ở dạng “Parquet files” và “Hive tables”

1.7 Xử lý và thao tác dữ liệu Spark SQL

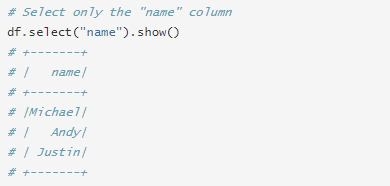
SparkSQL là một trong năm thành phần chính của Spark được thiết kế để xử lý dữ liệu có cấu trúc. Ta có thể tương tác với Spark SQL thông qua SQL, Dataframe hoặc Dataset.

Dataframe là một tập dữ liệu phân tán có cấu trúc như một tập dữ liệu 2D. Có thể hiểu đơn giản nó là bảng hay bảng tính SQL (nhưng phải là một bảng được tối ưu tốt).

Dataset rộng hơn Dataframe, có thể xem ngôn ngữ Java để dễ hiệu hơn. Nếu khai báo là Dataset<Row> trong Java Dataframe thì được được hiểu đơn giản Dataframe như là Dataset của các hàng. Ngoài ra, tập dữ liệu có thể chứa nhiều kiểu dữ liệu hơn, chẳng hạn như Dataset<String>, Dataset<Integer>.

Dưới đây là một số cách truy vấn :

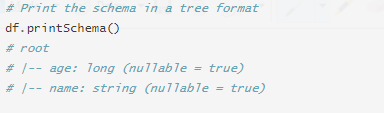
+ Truy vấn Select: giúp lấy ra các cột theo ý muốn khi mà không muốn in hết toàn bộ dữ liệu của Dataframe.



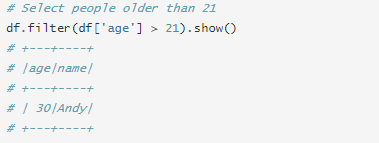
+ Truy vấn GroupBy: sẽ giúp nhóm lại các hàng mà có cùng giá trị cần nhóm và có thể thực hiện tính toán trên các giá trị mà ta nhóm lại.



+ PrintSchema: in ra cấu trúc của bảng hay là kiểu dữ liệu của từng cột.



+ Filter : Tương tự như Select, nhưng Select là lọc cột còn Filter là lọc hàng.



1.8 Xử lý và thao tác dữ liệu Mllib

1.8.1 Khái niệm

Spark MLlib là một modul nằm trên Spark Core thể hiện các nguyên bản về máy học dưới dạng API. Học máy thường xử lý một lượng lớn dữ liệu để đào tạo một mô hình.

Khung máy tính cơ sở từ Spark là một lợi ích to lớn. Trên hết, MLlib cung cấp hầu hết các thuật toán thống kê và học máy phổ biến. Điều này giúp đơn giản hóa đáng kể nhiệm vụ làm việc trên một dự án máy học quy mô lớn.

Spark MLlib được sử dụng để thực hiện học máy trong Apache Spark. MLlib chứa các thuật toán và tiện ích phổ biến. MLlib trong Spark là một thư viện của học máy có thể mở rộng để thảo luận về các thuật toán chất lượng cao và nhanh chóng.

1.8.2 Một số công cụ sử dụng Spark.Mllib

Spark.Mllib là API học máy chính cho Spark. Thư viện Spark.Mllib cung cấp một API cấp cao hơn được xây dựng trên DataFrames để xây dựng các pipeline cho machine learning. Một số công cụ như:

+ Thuật toán ML

+ Featurization

+ Pipelines

+ Persistence

+ Utilities

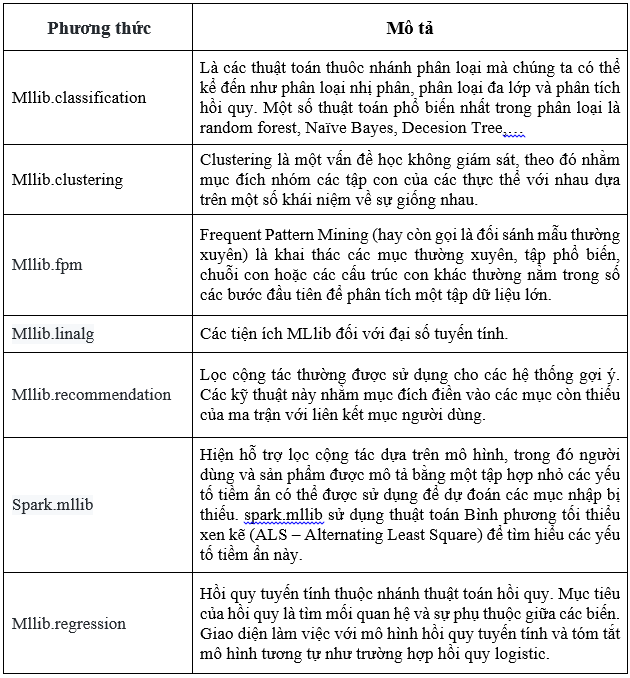
1.8.3 Thuật toán Machine Learning

Thuật toán ML là cốt lõi của MLlib. Điều này bao gồm các thuật toán học tập phổ biến như phân loại, hồi quy, phân cụm và lọc cộng tác. MLlib chuẩn hóa API để nhiều thuật toán có thể dễ dàng được kết hợp thành một đường dẫn hoặc quy trình làm việc duy nhất. Khái niệm chính là API đường ống, được lấy cảm hứng từ dự án đào tạo scikit - learning.

Transformer: Một thuật toán biến đổi một Dataframe thành một khung dữ liệu Dataframe khác. Về mặt lý thuyết, bằng cách thêm một hoặc nhiều cột, ta thực hiện phương thức transform( ) được sử dụng để chuyển đổi một Dataframe thành một Dataframe khác.

Estimator: là một thuật toán phù hợp trên Dataframe để tạo Transformer. Về mặt kỹ thuật, Estimator triển khai phương thức fit() và chấp nhận DataFrame tạo ra một mô hình là một transformer

1.8.4 Các phương thức thư viện Mllib cung cấp



Hình 3: Các phương thức thư viện Mllib cung cấp

1.9 Xử lý và thao tác dữ liệu Mllib

1.9.1 Word2Vec

Word2Vec là công cụ ước tính lấy chuỗi các từ đại diện cho tài liệu và đào tạo Word2VecModel. Mô hình ánh xạ mỗi từ với một vectơ có kích thước cố định duy nhất. Word2VecModel biến mỗi tài liệu thành một vectơ bằng cách sử dụng giá trị trung bình của tất cả các từ trong tài liệu, vectơ này sau đó có thể được sử dụng làm các tính năng để dự đoán, tính toán độ tương đồng của tài liệu .

Trong ví dụ sau, ta bắt đầu với một tập hợp các tài liệu, mỗi tài liệu được biểu diễn dưới dạng một chuỗi các từ. Đối với mỗi tài liệu, chuyển đổi nó thành một vector đặc trưng. Vectơ đặc trưng này sau đó có thể được chuyển cho một thuật toán học tập

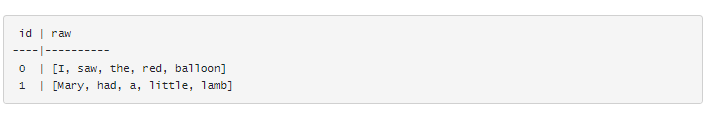


1.9.2 StopWordsRemover

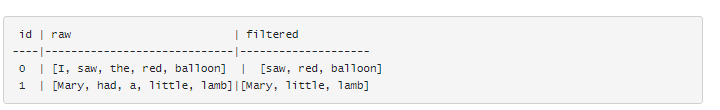
Stop words là những từ cần được loại trừ khỏi đầu vào, thường là vì các từ này xuất hiện thường xuyên và không mang nhiều ý nghĩa.

Stop AdWordsRemover nhận làm đầu vào một chuỗi các chuỗi (ví dụ: đầu ra của Tokenizer) và loại bỏ tất cả Stop words khỏi chuỗi đầu vào. Stop words mặc định cho một số ngôn ngữ có thể truy cập được bằng cách gọi StopWordsRemover.loadDefaultStopWords (ngôn ngữ), các tùy chọn có sẵn là “tiếng Tây Ban Nha”, “tiếng Hà Lan”, “tiếng Anh”, “tiếng Phần Lan”, “tiếng Pháp”, “tiếng Đức”, “tiếng Hungari”, “Tiếng Ý”, “tiếng Na Uy”, “tiếng Bồ Đào Nha”, “tiếng Nga”, “tiếng Tây Ban Nha”, “tiếng Thụy Điển” và “tiếng Thổ Nhĩ Kỳ”. Một tham số boolean có phân biệt chữ hoa chữ thường cho biết liệu các kết quả phù hợp có phân biệt chữ hoa chữ thường hay không.

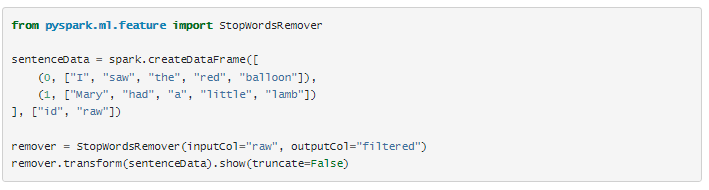
Giả sử rằng chúng ta có DataFrame sau với cột ID và raw:



Áp dụng StopWordsRemover với raw làm cột đầu vào và được lọc làm cột đầu ra:



Các stop words “I”, “the”, “had” và “a” đã được lọc bỏ.



1.9.3 RegexTokenizer

Tokenization là quá trình lấy văn bản (chẳng hạn như một câu) và chia nó thành các thuật ngữ riêng lẻ (thường là các từ). Một lớp Tokenizer đơn giản cung cấp chức năng này.

RegexTokenizer cho phép mã hóa nâng cao hơn dựa trên đối sánh biểu thức chính quy (regex). Theo mặc định, tham số “pattern” (regex, default: "\\ s +") được sử dụng làm dấu phân cách để tách văn bản đầu vào. Ngoài ra, người dùng có thể đặt tham số "khoảng trống" thành false cho biết "mẫu" regex biểu thị "mã thông báo" thay vì chia nhỏ khoảng trống và tìm tất cả các lần xuất hiện phù hợp dưới dạng kết quả mã hóa.



1.9.4 StringIndexer

StringIndexer mã hóa một cột chuỗi nhãn thành một cột chỉ số nhãn. StringIndexer có thể mã hóa nhiều cột. Các chỉ số nằm trong [0, numLabels) và bốn tùy chọn sắp xếp được hỗ trợ:

+ “frequencyDesc”: thứ tự giảm dần theo tần suất nhãn (nhãn thường xuyên nhất được gán 0)

+ “frequencyAsc”: thứ tự tăng dần theo tần suất nhãn (nhãn thường xuyên nhất được gán 0)

+ “AlphabetDesc”: thứ tự bảng chữ cái giảm dần.

+ “alphabetAsc”: thứ tự bảng chữ cái tăng dần (mặc định = “frequencyDesc”).

Lưu ý rằng trong trường hợp tần số bằng nhau khi ở dưới "frequencyDesc"/"frequencyAsc", các chuỗi được sắp xếp thêm theo bảng chữ cái.

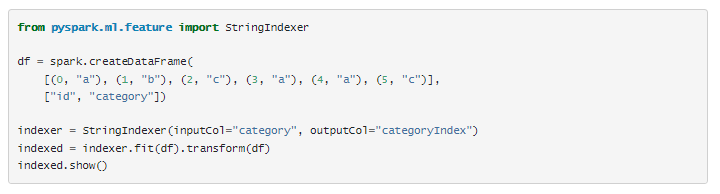
Các nhãn không nhìn thấy sẽ được đặt ở numLabels nếu người dùng chọn giữ chúng. Nếu cột đầu vào là số, ta ép kiểu thành chuỗi và lập chỉ mục các giá trị của chuỗi. Khi các thành phần của đường ống dẫn xuống như Estimator hoặc Transformer sử dụng nhãn được lập chỉ mục theo chuỗi này, ta phải đặt cột đầu vào của thành phần thành tên cột string-indexed này. Trong nhiều trường hợp, bạn có thể đặt cột đầu vào bằng setInputCol.

Giả sử rằng chúng ta có DataFrame sau với ID cột và category:



Category là một cột chuỗi có ba nhãn: “a”, “b” và “c”. Áp dụng StringIndexer với category làm cột đầu vào và categoryIndex làm cột đầu ra, sẽ được kết quả như sau:

“a” nhận chỉ số 0 vì nó là thường xuyên nhất, tiếp theo là “c” với chỉ số 1 và “b” với chỉ số 2.

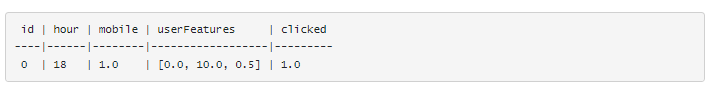


1.9.5 VectorAssembler

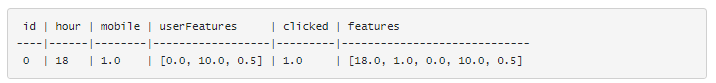
VectorAssembler là một máy biến áp kết hợp một danh sách các cột nhất định thành một cột vectơ duy nhất. Nó rất hữu ích để kết hợp các tính năng thô và các tính năng được tạo bởi các máy biến áp đặc trưng khác nhau thành một vectơ đặc trưng duy nhất, để đào tạo các mô hình ML như Hồi quy Logistic và Decision Trees.

VectorAssembler chấp nhận các kiểu cột đầu vào sau: tất cả các kiểu số, kiểu boolean và kiểu vectơ. Trong mỗi hàng, giá trị của các cột đầu vào sẽ được nối thành một vectơ theo thứ tự được chỉ định.

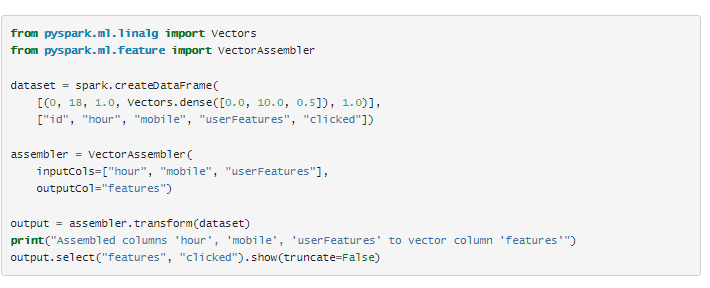
Giả sử rằng ta có DataFrame với các cột id, hour, mobile, userFeatures và clicked:



userFeatures là một cột vectơ chứa ba tính năng người dùng. Muốn kết hợp các Tính năng theo hour, mobile và userFeatures thành một vectơ đặc trưng duy nhất được gọi là các tính năng và sử dụng nó để dự đoán người dùng có clicked hay không. Nếu đặt các cột đầu vào của VectorAssembler thành hour, mobile, userFeatures và cột đầu ra thành các tính năng, sau khi chuyển đổi, chúng ta sẽ nhận được DataFrame sau:

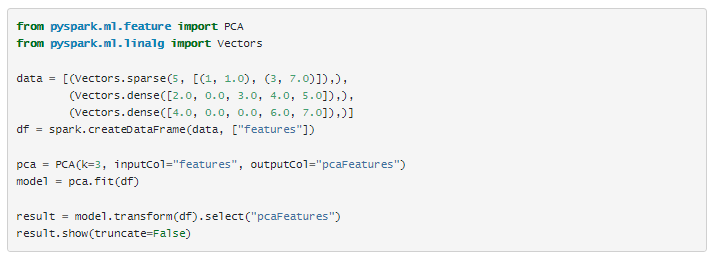


[VectorAssembler Python](https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.feature.VectorAssembler.html):



1.9.6 PCA

PCA là một thủ tục thống kê sử dụng phép biến đổi trực giao để chuyển đổi một tập hợp các quan sát của các biến có thể tương quan thành một tập giá trị của các biến không tương quan tuyến tính được gọi là các thành phần chính. Một lớp PCA đào tạo một mô hình để chiếu các vectơ lên không gian chiều thấp bằng cách sử dụng PCA. Ví dụ dưới đây cho thấy cách chiếu các vectơ đặc trưng 5 chiều thành các thành phần chính 3 chiều.



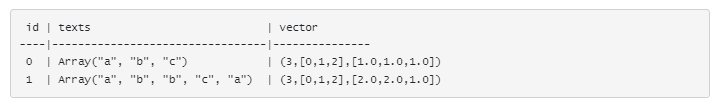
1.9.7 CountVectorizer

CountVectorizer và CountVectorizerModel nhằm mục đích giúp chuyển đổi một bộ sưu tập các tài liệu văn bản thành các vectơ của số lượng mã thông báo. Khi không có từ điển tiên nghiệm, CountVectorizer có thể được sử dụng làm Công cụ ước tính để trích xuất từ vựng và tạo CountVectorizerModel. Mô hình tạo ra các biểu diễn thưa thớt cho các tài liệu trên từ vựng, sau đó có thể được chuyển cho các thuật toán khác như LDA.

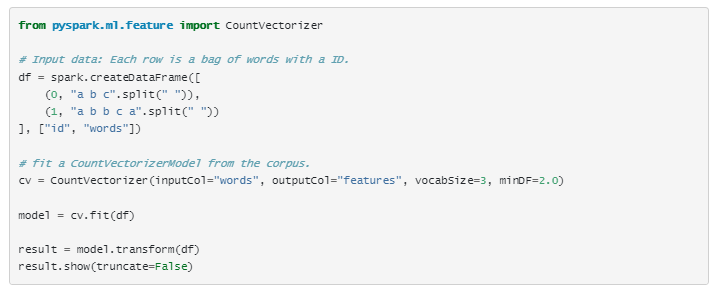
Giả sử rằng chúng ta có DataFrame sau với cột id và texts:



Mỗi hàng trong văn bản là một tài liệu kiểu Array[String]. Việc gọi phù hợp với CountVectorizer tạo ra CountVectorizerModel với từ vựng (a, b, c). Sau đó, cột đầu ra "vector" sau khi chuyển đổi chứa:

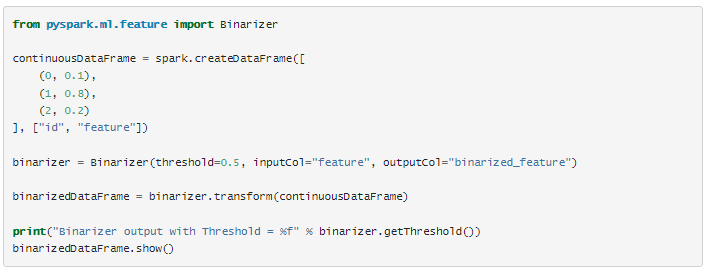


CountVectorizer Python:



1.9.8 Binarizer

Binarization là quá trình ngưỡng các đối tượng số thành đối tượng địa lý nhị phân (0/1). Binarizer lấy các tham số chung inputCol và outputCol, cũng như ngưỡng để mã hóa nhị phân. Giá trị tính năng lớn hơn ngưỡng được mã hóa thành 1,0; các giá trị bằng hoặc nhỏ hơn ngưỡng được mã hóa thành 0,0. Cả hai loại Vector và Double đều được hỗ trợ cho inputCol.



CHƯƠNG 2 – XỬ LÍ TÍNH TOÁN PHÂN TÁN

2.1 P

**TÀI LIỆU THAM KHẢO**

1. [A new high-level API for streaming] by Matei Zaharia, Tathagata Das, Michael Armbrust
2. <http://itechseeker.com/tutorials/apache-spark/lap-trinh-spark-voi-scala/spark-structured-streaming/>
3. <https://spark.apache.org/docs/latest/streaming-programming-guide.html>
4. <http://itechseeker.com/tutorials/apache-spark/lap-trinh-spark-voi-scala/spark-sql-dataset-va-dataframes/>
5. <https://spark.apache.org/docs/latest/ml-features?fbclid=IwAR2NDbvVkYkAeXX3ZivE8WH9d3co7LWJSRIKfPE2fiIAllbDDQowGJhgMOY#countvectorizer>