ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

____***____

BÁO CÁO MÔN HỌC KỸ THUẬT VÀ CÔNG NGHỆ DỮ LIỆU LỚN

ĐỀ TÀI

NAIVE BAYES VÀ LẬP TRÌNH MAPREDUCE HÓA TRONG PHÂN LỚP VĂN BẢN

Giảng viên hướng dẫn:

TS.Trần Hồng Việt

Ths.Ngô Minh Hương

Nhóm sinh viên thực hiện:

Ngô Huy Hoàn - 22022590

Nguyễn Hải Nam - 22022600

Tạ Nguyên Dũng - 22022546

Trần Phạm Hoàng - 22022669

HÀ NỘI, THÁNG 12/2024

MỞ ĐẦU

Dữ liệu luôn đóng vai trò quan trọng trong sự phát triển của nền văn minh nhân loại. Từ thời cổ đại, con người đã biết quan sát các hiện tượng tự nhiên và xã hội, ghi chép lại để rút ra các quy luật phục vụ cho cuộc sống. Tuy nhiên, do hạn chế về phương tiện lưu trữ, dữ liệu khi đó thường được ghi chép thủ công trên giấy tờ, dẫn đến quy mô dữ liệu còn nhỏ và khó quản lý. Đến thời kỳ cách mạng công nghệ 4.0, dữ liệu đã bùng nổ với tốc độ chưa từng có, tạo ra một khối lượng thông tin khổng lồ. Việc xử lý và phân tích dữ liệu để tìm ra những thông tin hữu ích trở thành một thách thức lớn, đòi hỏi các công cụ và phương pháp hiện đại, hiệu quả.

Trong bối cảnh đó, Hadoop nổi lên như một giải pháp quan trọng để lưu trữ và xử lý dữ liệu lớn (Big Data). Với khả năng triển khai mô hình MapReduce, Hadoop không chỉ giúp tối ưu hóa việc xử lý dữ liệu mà còn mở ra nhiều ứng dụng trong các lĩnh vực khác nhau, bao gồm bài toán phân lớp văn bản. Thuật toán Naive Bayes, một phương pháp đơn giản nhưng hiệu quả trong phân loại dữ liệu, đã được áp dụng rộng rãi. Khi tích hợp với Hadoop, thuật toán này có thể được triển khai trên quy mô lớn, giúp tăng tốc độ xử lý và cải thiện hiệu suất. Báo cáo này sẽ tập trung phân tích việc ứng dụng Hadoop để MapReduce hóa thuật toán Naive Bayes trong bài toán phân lớp văn bản, đồng thời thảo luận về các kết quả đạt được cũng như định hướng phát triển trong tương lai.

Mục lục

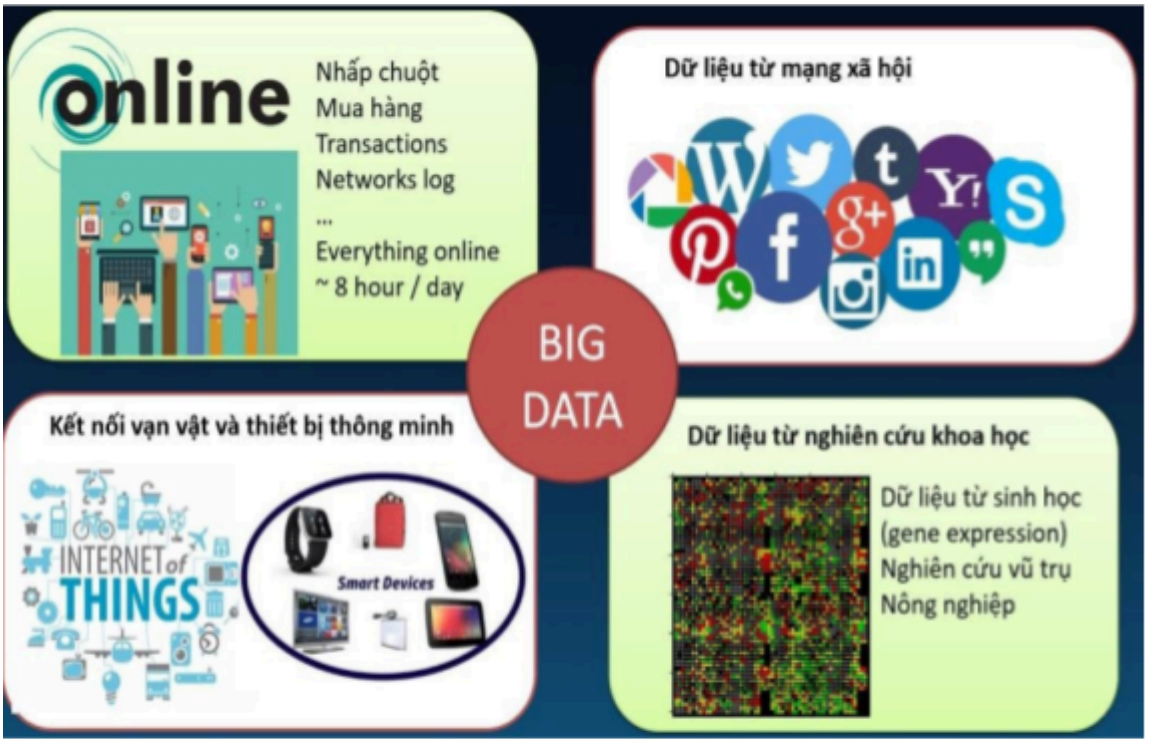
CHƯƠNG 1: TỔNG QUAN VỀ DỮ LIỆU LỚN VÀ HỆ SINH THÁI HADOOP

1.1 GIỚI THIỆU VỀ DỮ LIỆU LỚN

1.1.1 Định nghĩa

Theo wikipedia: Dữ liệu lớn (Big data) là một thuật ngữ chỉ bộ dữ liệu lớn hoặc phức tạp mà các phương pháp truyền thống không đủ các ứng dụng để xử lý dữ liệu này

Theo Gartner: Dữ liệu lớn là những nguồn thông tin có đặc điểm chung khối lượng lớn, tốc độ nhanh và dữ liệu định dạng dưới nhiều hình thức khác nhau, do đó muốn khai thác được đòi hỏi phải có hình thức xử lý mới để đưa ra quyết định, khám phá và tối ưu hóa quy trình.



Hình 1.1: Các nguồn dữ liệu lớn trong Big Data

1.1.2 Tính chất của dữ liệu lớn

Đặc trưng của dữ liệu lớn được thể hiện qua 5V:

Volume: Đặc điểm nổi bật nhất của Big Data là **khối lượng dữ liệu khổng lồ**. Nguồn dữ liệu đến từ nhiều nguồn khác nhau: mạng xã hội, cảm biến IoT, giao dịch tài chính, dữ liệu log, video, ảnh, văn bản, v.v. Ví dụ: Facebook xử lý hàng petabyte dữ liệu mỗi ngày.

Velocity: **Tốc độ tạo và xử lý dữ liệu** là yếu tố quan trọng. Dữ liệu từ cảm biến, giao dịch thời gian thực, hoặc các nền tảng mạng xã hội cần được xử lý ngay lập tức hoặc trong thời gian rất ngắn. Ví dụ: Một nền tảng như Twitter phải xử lý hàng triệu tweet mỗi phút.

Variety: Big Data thể hiện sự đa dạng về nguồn gốc và định dạng dữ liệu, bao gồm **dữ liệu có cấu trúc** như bảng cơ sở dữ liệu, **dữ liệu bán cấu trúc** như JSON và XML, **dữ liệu phi cấu trúc** như hình ảnh, video, âm thanh hay email. Sự phong phú này đặt ra yêu cầu cao về tính linh hoạt của các hệ thống nhằm đảm bảo khả năng xử lý và phân tích hiệu quả.

Veracity: **Độ tin cậy của dữ liệu** là một thách thức lớn. Dữ liệu có thể chứa lỗi, thiếu sót, hoặc không đầy đủ. Xử lý dữ liệu "nhiễu" và đảm bảo tính chính xác, đáng tin cậy là rất quan trọng để phân tích đúng đắn. Ví dụ: Trích xuất thông tin từ các bài đăng mạng xã hội có thể gặp phải thông tin sai lệch.

Value: **Giá trị** là mục tiêu cuối cùng của Big Data. Dữ liệu lớn chỉ thực sự hữu ích nếu nó có thể được khai thác để mang lại giá trị cho doanh nghiệp hoặc xã hội, như tối ưu hóa hoạt động kinh doanh, hiểu khách hàng sâu sắc hơn, hoặc hỗ trợ quyết định nhanh chóng. Ví dụ: Các công ty thương mại điện tử như Amazon sử dụng Big Data để cá nhân hóa trải nghiệm khách hàng, tăng doanh thu.

1.1.3 Các loại dữ liệu lớn

Dựa trên kiểu dữ liệu

Big Data được phân loại thành ba cấu trúc chính sau:

Dữ liệu có cấu trúc (Structured Data): Dữ liệu được tổ chức trong các bảng, hàng, và cột, dễ dàng lưu trữ và xử lý bằng các hệ quản trị cơ sở dữ liệu (RDBMS). Ví dụ: Thông tin khách hàng (tên, địa chỉ, số điện thoại), bảng giao dịch ngân hàng.

Dữ liệu bán cấu trúc (Semi-Structured Data): Dữ liệu không có cấu trúc hoàn toàn, nhưng có định dạng và các trường thông tin nhất định giúp tổ chức. Ví dụ: Tệp JSON, XML, Log file của ứng dụng hoặc hệ điều hành.

Dữ liệu phi cấu trúc (Unstructured Data): Không có cấu trúc cố định, khó phân tích trực tiếp mà không qua bước xử lý ban đầu. Ví dụ: Hình ảnh, video, âm thanh, Email, tài liệu văn bản.

Dựa trên nguồn gốc dữ liệu

Big Data có thể được chia thành:

Dữ liệu giao dịch (Transactional Data): Dữ liệu từ các hệ thống giao dịch như ngân hàng, bán lẻ, thương mại điện tử. Ví dụ: Lịch sử mua sắm trên Amazon.

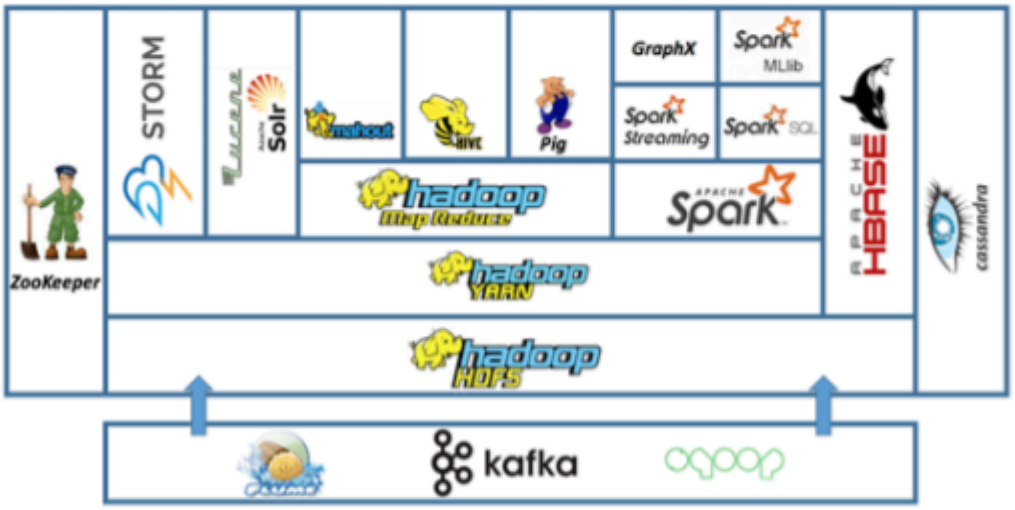
Dữ liệu từ máy móc (Machine Data): Dữ liệu được tạo bởi các thiết bị hoặc hệ thống tự động. Ví dụ: Dữ liệu cảm biến từ IoT, log server, và dữ liệu máy móc sản xuất.

Dữ liệu từ con người (Human-Generated Data): Dữ liệu từ hoạt động trực tuyến hoặc offline của con người. Ví dụ: Bài đăng mạng xã hội, bình luận, đánh giá sản phẩm, video YouTube.

1.2 GIỚI THIỆU VỀ HADOOP

1.2.1 Tổng quan về Hadoop

Hadoop là một framework mã nguồn mở được phát triển bởi Apache Software Foundation. Nó được thiết kế để hỗ trợ lưu trữ và xử lý lượng dữ liệu lớn (Big Data) trên các hệ thống phân tán (distributed systems). Hadoop có khả năng mở rộng cao và thường được sử dụng trong các môi trường cần xử lý dữ liệu lớn và phức tạp.



Hình 1.2: Hệ sinh thái Hadoop

Hadoop sử dụng mô hình phân tán để lưu trữ và xử lý dữ liệu trên các máy tính thông thường. Thay vì lưu trữ tất cả dữ liệu trên một máy chủ duy nhất, Hadoop phân chia dữ liệu thành các khối và lưu chúng trên nhiều máy tính. Điều này giúp tăng khả năng mở rộng, độ tin cậy và hiệu năng.

Hadoop phân tán dữ liệu trên nhiều nút máy tính và xử lý nó song song trên các nút. Điều này giúp giảm thời gian xử lý và tăng hiệu suất.

Hadoop được sử dụng rộng rãi trong các lĩnh vực như khoa học dữ liệu, trí tuệ nhân tạo, tài chính, ý tế, và nhiều lĩnh vực khác.

1.2.2 Thành phần của Hadoop

Hadoop bao gồm ba thành phần chính:

Hadoop Distributed File System (HDFS): Là hệ thống tệp phân tán, cho phép lưu trữ dữ liệu lớn trên nhiều máy chủ. Dữ liệu được chia thành các khối (blocks) và phân phối trên các máy tính khác nhau để đảm bảo khả năng chịu lỗi và tăng tốc độ truy xuất.

MapReduce: Mô hình lập trình và framework xử lý dữ liệu song song. Nó chia nhỏ công việc thành các bước "Map" (lọc và phân loại) và "Reduce" (tổng hợp kết quả). Nó hoạt động dựa trên 2 pha chính là Map và Reduce. Pha Map sẽ chia nhỏ dữ liệu thành các cặp key-value, sau đó pha Reduce sẽ nhóm các cặp key-value lại với nhau để tính toán kết quả cuối cùng.

YARN (Yet Another Resource Negotiator): Là thành phần quản lý tài nguyên trong Hadoop. Nó quản lý và phân phối tài nguyên (CPU, bộ nhớ) cho các ứng dụng chạy trên cluster Hadoop.

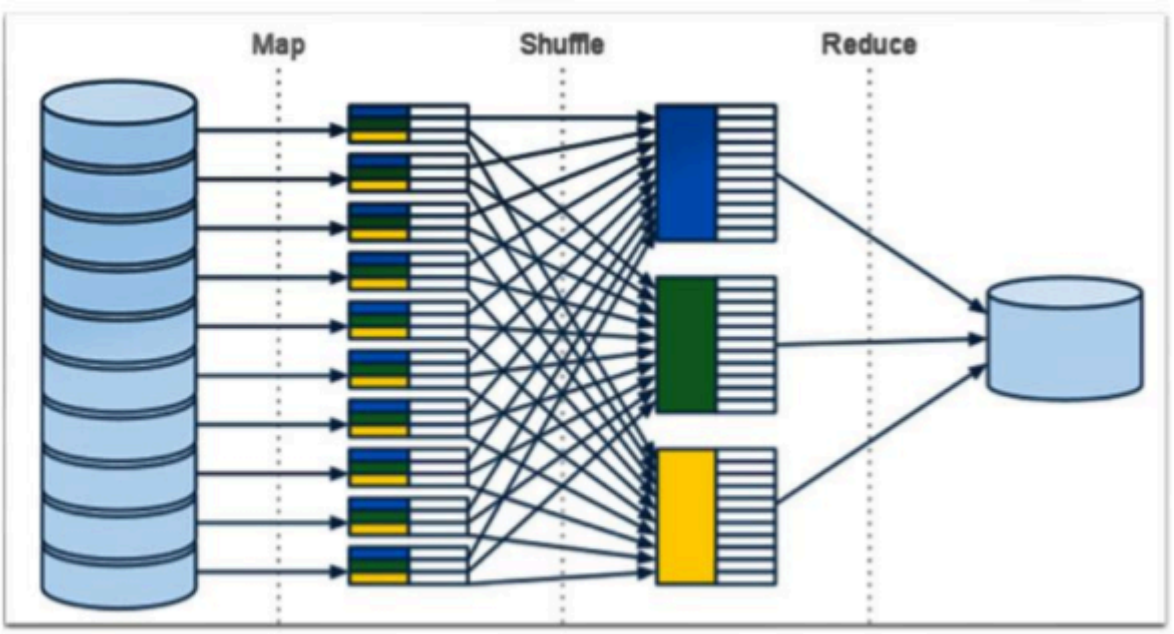
1.2.3 Tổng quan về MapReduce

MapReduce là một framework xử lý dữ liệu phân tán, cho phép xử lý song song một lượng lớn dữ liệu với khả năng chịu lỗi cao trên hàng ngàn máy tính trong cụm. Được phát triển bởi Google và sau đó phát hành bởi Apache Software Foundation như một phần của hệ sinh thái Hadoop, MapReduce chia dữ liệu thành các phần nhỏ, xử lý song song trên các nút máy tính trong mạng Hadoop. Khung làm việc này được thiết kế để xử lý dữ liệu độc lập, giúp tăng tốc độ xử lý thông qua việc phân tách và xử lý song song.

Map-Reduce thực hiện 2 chức năng chính đó là Map và Reduce:

Map: Sẽ thực hiện đầu tiên, có chức năng tải, phân tích dữ liệu đầu vào và được chuyển đổi thành tập dữ liệu theo cặp key/value.

Reduce: Sẽ nhận kết quả đầu ra từ tác vụ Map, kết hợp dữ liệu lại với nhau thành tập dữ liệu nhỏ hơn.



Hình 1.4: Sơ đồ MapReduce

Ưu điểm của mô hình MapReduce:

Khả năng mở rộng cao: Nó có thể hoạt động trên các cụm máy tính phân tán với hàng ngàn node.

Tự động xử lý phân tán: Hệ thống tự động chia nhỏ công việc và phân phối đến các node, giảm gánh nặng cho lập trình viên trong việc quản lý phân tán.

Độ tin cậy và độ chịu lỗi: Khi một node gặp sự cố, hệ thống sẽ tự động phân phối lại công việc đến các node khác.

Đơn giản và dễ sử dụng: Sử dụng 2 hàm chính Map (phân chia công

việc) và Reduce (tổng hợp kết quả).

Tương thích với nhiều loại dữ liệu: MapReduce hỗ trợ nhiều định dạng dữ liệu khác nhau và có thể xử lý dữ liệu có cấu trúc, bán và phi cấu trúc.

Nhược điểm của mô hình MapReduce:

Không tối ưu cho xử lý thời gian thực: MapReduce không thích hợp cho các ứng dụng yêu cầu thời gian thực hoặc độ trễ thấp, quá trình đọc, ghi dữ liệu trên HDFS và tính toán qua nhiều bước có thể mất nhiều thời gian.

Giới hạn trong tính linh hoạt: Việc chỉ dựa vào hai giai đoạn (Map và Reduce) có thể làm phức tạp một số tác vụ xử lý dữ liệu không phù hợp với mô hình này, ví dụ như xử lý đồ thị hoặc thuật toán đòi hỏi nhiều vòng lặp.

Hiệu suất thấp cho dữ liệu nhỏ: Với khối lượng dữ liệu nhỏ, chi phí của việc thiết lập và quản lý phân tán lớn hơn lợi ích đạt được.

Khó khăn trong debug và kiểm thử: Vì là hệ thống phân tán, việc debug lỗi trong MapReduce có thể phức tạp hơn so với các chương trình chạy trên một máy duy nhất.

CHƯƠNG 2: THUẬT TOÁN NAIVE BAYES TRONG PHÂN LOẠI VĂN BẢN

2.1 PHÂN LOẠI VĂN BẢN

2.1.1 Khái niệm phân loại văn bản

Phân loại văn bản là một kỹ thuật trong học máy dùng để gán các danh mục đã được xác định trước cho văn bản đầu vào. Kỹ thuật này cho phép sắp xếp, tổ chức và phân tích các loại văn bản khác nhau, từ tài liệu, bài báo đến các bình luận trên mạng xã hội.

Ví dụ:

- Phân loại email thành spam và không spam.
- Phân loại tin tức theo các chủ đề như kinh tế, chính trị, hoặc thể thao.
- Phân tích cảm xúc trên các bài đăng mạng xã hội, xác định nội dung là tích cực, tiêu cực, hoặc trung lập.

Đây là một lĩnh vực quan trọng trong xử lý ngôn ngữ tự nhiên (NLP), với ứng dụng rộng rãi trong phân tích cảm xúc, phát hiện thư rác, hoặc tổ chức tài liệu.

Tại sao nên sử dụng phân loại văn bản học máy? Một số lý do hàng đầu:

Khả năng mở rộng: Phân tích và sắp xếp theo cách thủ công chậm và kém chính xác hơn nhiều. Học máy có thể tự động phân tích hàng triệu cuộc khảo sát , nhận xét, email, v.v., với chi phí thấp, thường chỉ trong vài phút. Các công cụ phân loại văn bản có thể mở rộng theo mọi nhu cầu kinh doanh, dù lớn hay nhỏ.

Phân tích thời gian thực: Có những tình huống quan trọng mà các công

ty cần xác định càng sớm càng tốt và hành động ngay lập tức (ví dụ: khủng hoảng PR trên mạng xã hội). Phân loại văn bản bằng máy học có thể theo dõi đề cập đến thương hiệu của chúng ta một cách liên tục và theo thời gian thực, do đó, ta sẽ xác định được thông tin quan trọng và có thể thực hiện hành động ngay lập tức.

Tiêu chí nhất quán: Con người chú thích mắc lỗi khi phân loại dữ liệu văn bản do mất tập trung, mệt mỏi, nhàm chán và tính chủ quan của con người tạo ra các tiêu chí không nhất quán. Mặt khác, học máy áp dụng cùng một lăng kính và tiêu chí cho tất cả dữ liệu và kết quả. Khi mô hình phân loại văn bản huấn luyện đúng cách, nó sẽ hoạt động với độ chính xác vượt trội.

2.1.2 Phân loại văn bản hoạt động như nào?

Có rất nhiều phương pháp để phân loại văn bản, bao gồm các thuật toán học máy như Support Vector Machine (SVM), Decision Tree, và các mô hình học sâu (Deep Learning) như LSTM hay BERT. Tuy nhiên, trong báo cáo này, nhóm chúng em sẽ tập trung sử dụng Naive Bayes.

Naive Bayes là một thuật toán xác suất dựa trên Định lý Bayes, với giả định rằng các đặc trưng trong dữ liệu là độc lập với nhau. Đây là phương pháp đơn giản nhưng hiệu quả, đặc biệt phù hợp với các bài toán phân loại văn bản nhờ khả năng xử lý nhanh chóng và hiệu quả trên tập dữ liệu lớn.

Trong báo cáo này, Naive Bayes sẽ được áp dụng để dự đoán danh mục văn bản và đánh giá hiệu suất thông qua các chỉ số như độ chính xác, độ nhạy và F1-score.

2.1.3 Ứng dụng của phân loại văn bản

Phân loại văn bản có nhiều ứng dụng thực tiễn trong các lĩnh vực khác nhau, từ kinh doanh, giáo dục đến truyền thông và công nghệ. Một trong những ứng dụng phổ biến nhất là trong việc phân loại email, giúp tự động xác định và lọc các thư rác (spam), từ đó cải thiện trải nghiệm người dùng và bảo mật thông tin.

Trong lĩnh vực truyền thông, phân loại văn bản trong truyền thông giúp sắp xếp bài báo theo chủ đề như kinh tế, thể thao, chính trị, và hỗ trợ phân tích cảm xúc từ đánh giá, bình luận hay bài đăng mạng xã hội, giúp doanh nghiệp cải thiện sản phẩm. Nó cũng được sử dụng trong hệ thống quản lý tài liệu để tổ chức và tìm kiếm hiệu quả.

Trong môi trường số, công nghệ này còn đóng vai trò quan trọng trong việc phát hiện thông tin độc hại như tin giả, phát ngôn thù ghét hoặc nội dung không phù hợp, góp phần xây dựng môi trường mạng an toàn và lành mạnh. Những ứng dụng đa dạng này cho thấy tầm quan trọng và tiềm năng phát triển mạnh mẽ của phân loại văn bản trong thời đại số hóa.

2.2 NAIVE BAYES TRONG PHÂN LOẠI VĂN BẢN

2.2.1 Tổng quan về Naive Bayes

Thuật toán Naive Bayes là một bộ phân loại dựa trên Định lý Bayes, sử dụng giả định rằng các đặc trưng trong dữ liệu là độc lập với nhau. Mặc dù giả định này có thể không hoàn toàn chính xác trong thế giới thực, thuật toán Naive Bayes vẫn được ứng dụng rộng rãi nhờ tính đơn giản, hiệu quả và khả năng dự đoán nhanh chóng.

Thuật toán này hoạt động bằng cách tính toán xác suất của mỗi lớp dựa trên

các đặc trưng của dữ liệu đầu vào. Thuật toán này tìm ra nhãn làm cho xác suất hậu nghiệm(Posterior) trở nên cực đại dựa trên tích của xác suất tiên nghiệm(Prior) và ước lượng hợp lý(Likelihood).

Naive Bayes thường được áp dụng trong các bài toán phân loại, đặc biệt là phân loại văn bản như lọc thư rác, phân tích cảm xúc, và phân loại các tài liệu theo chủ đề. Ưu điểm nổi bật của thuật toán là tốc độ và khả năng xử lý các tập dữ liệu lớn, với việc dự đoán nhanh chóng và chính xác.

Tại sao nó được gọi là Naive Bayes?

Tên gọi "Naive Bayes" xuất phát từ hai yếu tố: "Naive" (ngây thơ) phản ánh giả định đơn giản hóa về tính độc lập của các đặc trưng, và "Bayes" ám chỉ đến nhà thống kê Thomas Bayes, người đã phát triển Định lý Bayes, cơ sở lý thuyết của thuật toán này.

2.2.2 Nguyên lý hoạt động của Naive Bayes

Định lý Bayes cho phép ta tính toán xác suất xảy ra của một sự kiện dựa trên thông tin liên quan đến sự kiện khác đã xảy ra. Định lý Bayes được phát biểu như sau:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \qquad (2.1)$$

Trong đó, A và B là các sự kiện, và $P(B) \neq 0$.

- P(A) là xác suất trước (prior probability), tức là xác suất của A trước khi có thông tin từ bằng chứng B.
- P(B) là xác suất biên (marginal probability), là xác suất xảy ra của bằng

chứng B.

- P(A ∣ B) là xác suất hậu nghiệm (posterior probability), tức là xác suất của sự kiện A sau khi có thông tin từ bằng chứng B.
- P(B ∣ A) là xác suất có điều kiện (likelihood), thể hiện khả năng xảy ra của B khi giả thuyết A là đúng.

Định lý Bayes là một công cụ quan trọng trong lý thuyết xác suất và học máy, giúp cải thiện khả năng dự đoán bằng cách điều chỉnh các giả thuyết dựa trên dữ liệu mới.

Áp dụng vào bài toán phân loại

Trong bài toán phân loại, với tập dữ liệu X = [$x_1, x_2,..., x_n$] , trong đó X là vectơ đặc trưng có kích thước n và c là biến lớp, áp dụng Định lý Bayes để tính toán xác suất của lớp c khi biết các đặc trưng X.

$$P(c \mid X) = \frac{P(X \mid c)P(c)}{P(X)} \qquad (2.2)$$

Giả sử các đặc trưng [$x_1, x_2,..., x_n$] độc lập với nhau. Nếu hai sự kiện A và B là độc lập, thì ta có:

$$P(A, B) = P(A)P(B) \qquad (2.3)$$

Dựa vào giả định này, ta có thể viết lại xác suất hậu nghiệm như sau:

$$P(c \mid x_1, x_2,..., x_n) = \frac{P(x_1|c)\, P(x_2|c)... P(x_n|c)\, P(c)}{P(x_1)\, P(x_2)... P(x_n)} \qquad (2.4)$$

Do $P(x_1)\, P(x_2)... P(x_n)$ là hằng số đối với lớp y, ta có thể bỏ qua phần mẫu trong

phân số, từ đó viết lại công thức:

$$P(c \mid x_1, x_2, \ldots, x_n) \propto P(c) \prod_{i=1}^{n} P(x_i \mid c) \qquad (2.5)$$

Khi đó, để xây dựng một bộ phân loại, ta tính toán xác suất của $P(y)$ và $P(x_i \mid y)$ cho tất cả các giá trị có thể của lớp y, và chọn lớp có xác suất cao nhất. Ta có:

$$c = argmax_c P(c) \prod_{i=1}^{n} P(x_i \mid c) \qquad (2.6)$$

Để tính toán công thức trên, ta chỉ cần tính $P(c)$ (xác suất của lớp) và $P(x_i \mid c)$ (xác suất có điều kiện của các đặc trưng đối với lớp c).

2.2.2 Ưu và nhược điểm của Naive Bayes

Ưu điểm

Đơn giản và nhanh chóng: Dễ dàng triển khai và tính toán, phù hợp với các ứng dụng yêu cầu xử lý thời gian thực.

Hiệu quả với dữ liệu nhỏ: Hoạt động tốt ngay cả khi ít dữ liệu đào tạo.

Khả năng mở rộng: Xử lý tốt với các tập dữ liệu lớn nhờ tính toán nhanh và không yêu cầu tài nguyên lớn.

Ứng dụng rộng rãi: Đặc biệt hiệu quả trong phân loại văn bản, lọc thư rác, và phân tích cảm xúc.

Không đòi hỏi tinh chỉnh nhiều: Không cần nhiều tham số phức tạp hoặc điều chỉnh thủ công.

Nhược điểm

Giả định tính độc lập: Dựa trên giả định các đặc điểm độc lập, điều này hiếm khi đúng trong thực tế và có thể làm giảm độ chính xác.

Không xử lý tốt dữ liệu có đặc điểm phụ thuộc mạnh: Khi các đặc điểm có mối quan hệ chặt chẽ, kết quả dự đoán dễ bị sai lệch.

Không phù hợp với dữ liệu số liên tục: Đòi hỏi xử lý trước, như chuyển đổi dữ liệu thành dạng phân loại hoặc sử dụng các kỹ thuật ước lượng phân phối xác suất.

Không cạnh tranh với các phương pháp tiên tiến: Về độ chính xác, thường kém hơn các mô hình hiện đại như rừng ngẫu nhiên hay máy tăng cường độ dốc trong các bài toán phức tạp.

Dễ bị ảnh hưởng bởi dữ liệu mất cân bằng: Nếu dữ liệu chứa các lớp có tần suất xuất hiện chênh lệch lớn, thuật toán có thể ưu tiên lớp có tần suất cao hơn.

2.2.3 Ứng dụng của Naive Bayes

Real-time Prediction: Naive Bayes chạy khá nhanh nên nó thích hợp áp dụng nhiều vào các ứng dụng chạy thời gian thực, như hệ thống cảnh báo phát hiện sự cố.

Multi-class Prediction: Nhờ vào định lý Bayes mở rộng, thuật toán có thể được ứng dụng vào các loại dự đoán đa mục tiêu, tức là các ứng dụng có khả năng dự đoán nhiều giả thuyết mục tiêu.

Text Classification/ Spam Filtering/ Sentiment Analysis: Naive Bayes rất thích hợp cho các hệ thống phân loại văn bản hay xử lý ngôn ngữ tự nhiên nhờ tính chính xác cao hơn so với nhiều thuật toán khác. Ngoài ra, các hệ thống chống thư rác cũng rất ưa chuộng sử dụng thuật toán này. Các hệ thống phân tích tâm lý thị trường cũng áp dụng Naive Bayes để tiến hành phân tích tâm lý người dùng, xác định xu hướng ưa chuộng hoặc không ưa chuộng sản phẩm thông qua việc phân tích thói quen và hành động của khách hàng.

Recommendation System: Naive Bayes được sử dụng rộng rãi để xây dựng các hệ thống gợi ý.

2.2.4 Naive Bayes trong phân loại văn bản

Văn bản là dữ liệu phi cấu trúc, nhưng có thể áp dụng Naive Bayes nhờ vào giả định độc lập: Các token trong văn bản được giả định là độc lập với nhau và chỉ phụ thuộc vào lớp. Giả định này cho phép mở rộng hoặc thu hẹp số lượng token, phù hợp với độ dài không cố định của văn bản.

Trong bài toán phân loại văn bản, với tập dữ liệu D bao gồm các văn bản đã được gán nhãn và một văn bản cần phân loại T= $[t_1, t_2,.., t_n]$, trong đó t_i là các token của văn bản, áp dụng Bayes để tính xác suất của lớp y khi biết các token T.

Để xác định lớp c_T của văn bản T, ta chọn lớp có xác suất cao nhất:

$$c = argmax_c P(c) \prod_{i=1}^{n} P(t_i \mid c)$$

Ta chỉ cần tính $P(c)$ (xác suất của lớp) và $P(t_i \mid c)$ (xác suất có điều kiện của các đặc trưng đối với lớp c).

Xác suất của lớp P(c): Là tần suất xuất hiện của lớp c trong tập dữ liệu:

$$P(c) = \frac{N_c}{N} \qquad (2.7)$$

Trong đó:

- N_c: Số lượng văn bản thuộc lớp c.

- N: Tổng số văn bản trong tập dữ liệu.

Xác suất của token $P(t_i \mid c)$*:* Là tần suất xuất hiện của token t_i trong lớp c:

$$P(t_i \mid c) = \frac{count(t_i \mid c)}{\Sigma_{t \in V} count(t, c)} \qquad (2.8)$$

Trong đó:

- $Count(t_i \mid c)$: Số lần token t_i xuất hiện trong văn bản thuộc lớp c.
- V: Tập hợp tất cả các token trong từ điển của lớp y.

Xử lý vấn đề dữ liệu chưa xuất hiện (Làm mịn Laplace)

Nếu token t_i *không xuất hiện*: Xác suất $P(t_i \mid c)$ bằng 0. Ta áp dụng làm mịn Laplace:

$$P(t_i \mid c) = \frac{count(t_i \mid c) + 1}{\Sigma_{t \in V} count(t, c) + |V|} \qquad (2.9)$$

Sử dụng log để tránh các giá trị nhỏ: Do $P(t_i \mid c)$ thường rất nhỏ, ta dùng log để tránh underflow:

$$c = argmax_c \left(log(P(c) + \sum_{i=1}^{n} log(P(t_i \mid c)\right) \qquad (2.10)$$

Khi đó, thuật toán sẽ tính toán $logP(c)$ và $logP(t_i \mid c)$ để chọn lớp có xác

suất lớn nhất cho văn bản T.

Chương 3: MAPREDUCE HÓA THUẬT TOÁN NAIVE BAYES CHO PHÂN LỚP VĂN BẢN.

3.1 CHUẨN BỊ DỮ LIỆU

Việc MapReduce hóa thuật toán Naive Bayes cho phân lớp văn bản bao gồm chia quá trình huấn luyện và phân lớp thành các bước mà có thể được xử lý song song trên nhiều nút trong hệ thống phân tán.

Bộ dữ liệu sử dụng : lưu dưới dạng file CSV gồm 2 cột :

Cột 1 (Loại tin nhắn):

- Lưu trữ giá trị "ham" hoặc "spam" để phân loại tin nhắn.
- Dữ liệu dạng text.

Cột 2 (Nội dung tin nhắn):

- Lưu trữ nội dung của từng tin nhắn, dạng text.

3.2 QUÁ TRÌNH TÍNH TOÁN VÀ SONG SONG HÓA VỚI MAPREDUCE

3.2.1 Thực hiện đếm số lần mỗi token xuất hiện ứng với mỗi Class

Mapper nhận đầu vào từ file và thực hiện các bước sau:

Đọc dữ liệu:

- Input: Một dòng dữ liệu dạng Class, Text.

Tách dữ liệu:

- Tách dòng val thành 2 phần: Text (nội dung) và Class (loại).

- Tiền xử lý dữ liệu:
  - Áp dụng mô hình xử lý ngôn ngữ tự nhiên Stanford CoreNLP để loại bỏ stopwords, punctuations và tokenize dữ liệu Text.

Tạo key-value mới:

- Với mỗi token, kết hợp với class để tạo thành key mới.
- Key-value đầu ra của hàm map:
  $(newkey, newval) = ((token, class), [0/1])$

Reducer tổng hợp các cặp key-value từ hàm map:

Đầu vào:

- Một key là cặp (token, class).
- Một danh sách các giá trị (value list) tương ứng với key.

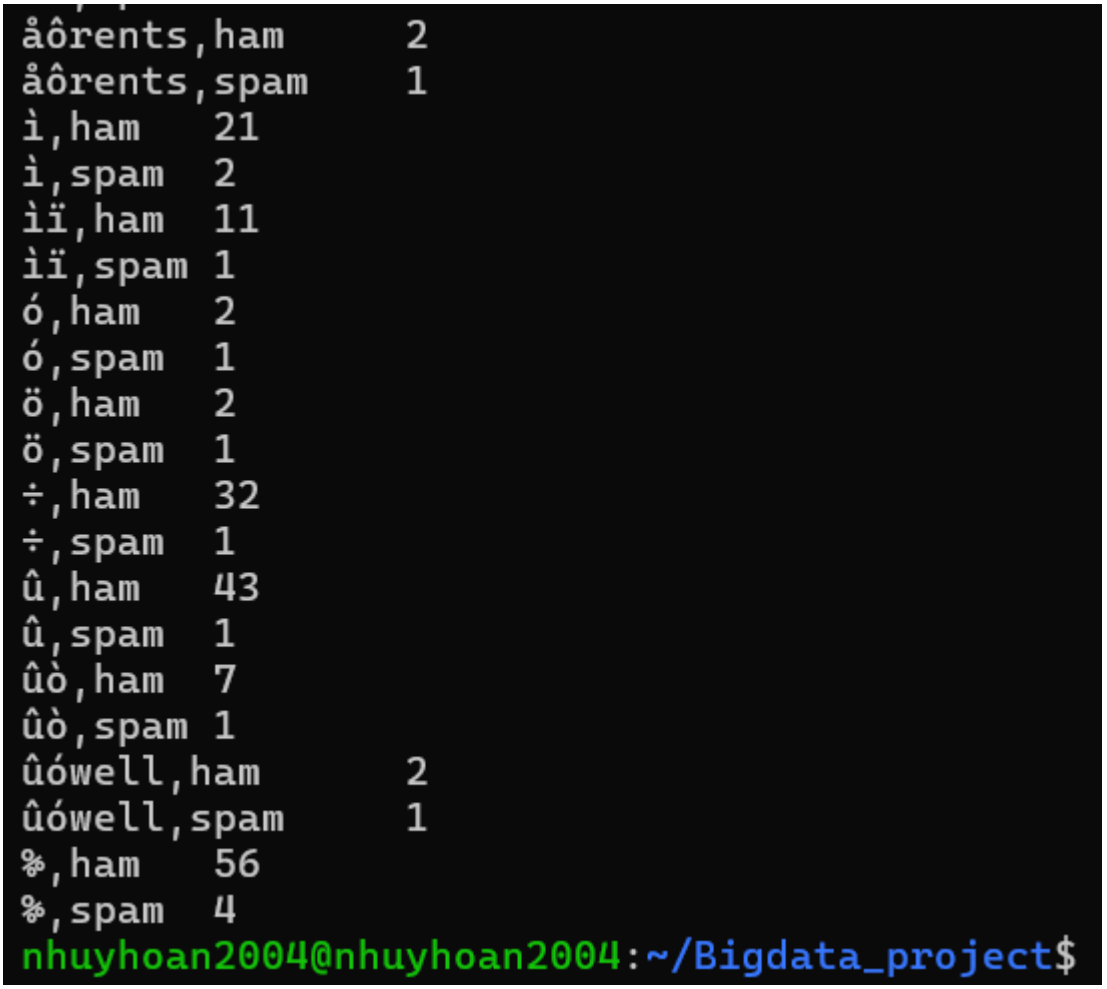
Thực hiện tính tổng:

- Tính tổng tất cả các giá trị trong danh sách (value list).
- Đầu ra là tổng số lần xuất hiện của từ t_i trong class c_i.

Output cuối:

- Kết quả được lưu vào file đầu ra (countWordClass):
  (newkey,newval)=((token,class), sum+1)

Sau khi hoàn thành job, thu được file đầu ra countWordClass chứa số lần mỗi token t_i xuất hiện trong từng class c_i.



3.2.2 Tính tổng số lần các token xuất hiện ứng với mỗi Class

Đầu vào:

- Sử dụng file đầu ra từ job trước đó (countWordClass) làm dữ liệu đầu vào.

File này chứa số lần xuất hiện của từ t_i trong mỗi class c.

Quy trình MapReduce:

- Mapper:
  - Input: $(key, value) = ((token, class), count)$.

- Output: $(class, count)$.

- Reducer:
  - Tổng hợp tất cả các giá trị count theo từng class.
  - Output: $(class, \sum_{t \in V} (count(t, c))$

Kết quả: File đầu ra tên countAnyClass.

```
nhuyhoan2004@nhuyhoan2004:~/Bigdata_project$ hadoop fs -cat /user/nhuyhoan2004/countAnyClass/part-r-00000
ham     40689
spam    18086
nhuyhoan2004@nhuyhoan2004:~/Bigdata_project$
```

3.2.3 Tính toán xác suất hậu nghiệm

Tính xác suất của từng từ t_i xuất hiện trong class c bằng cách chia giá trị từ file countWordClass cho tổng trong file countAnyClass. Dùng công thức (2.9):

$$P(t_i \mid c) = \frac{count(t_i \mid c) + 1}{\Sigma_{t \in V} count(t, c) + |V|}$$

Đầu vào:

- File countWordClass: $((token, class), count)$.
- File countAnyClass: (class, $\sum_{t \in V}$ (count(t,c))

Quy trình MapReduce:

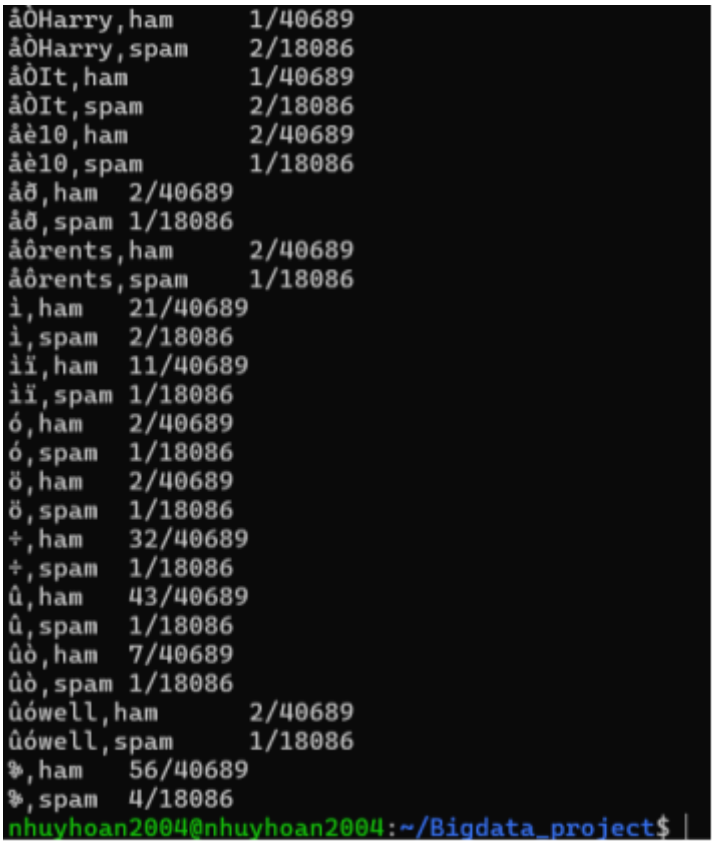
- Mapper:
  - Đọc dữ liệu từ cả hai file, ánh xạ class từ file countWordClass

và file countAnyClass.

- Output: $((token, class),\ probability)$

- Reducer: Không cần thực hiện.

Kết quả: File đầu ra là likelihood, chứa xác suất $P(t_i \mid c)$ cho mỗi token t_i trong từng class.



Hình 3.2: Xác suất P(ti|c)

3.3.4 Tính toán xác suất tiên nghiệm

Xác định số lượng tài liệu thuộc mỗi class c và tổng toàn bộ tài liệu N.

Đầu vào

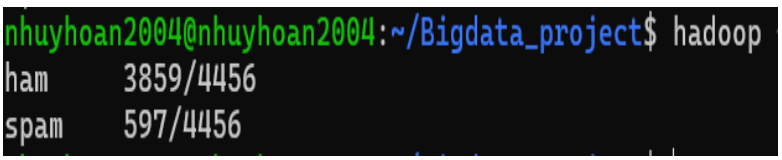
- Sử dụng dữ liệu gốc (file train), mỗi dòng chứa nội dung và class tương ứng.

Quy trình MapReduce

- Mapper:
  - Input: Dòng dữ liệu dạng (Class,Text)
  - Output: (class,1)
- Reducer:
  - Tính tổng số lần xuất hiện của từng class c, tạo output: $(class, Nc)$.

Kết quả

- Tổng số tài liệu N
- Tính xác suất prior :$P(c) = \frac{N_c}{N}$



3.3.5 Quá trình kiểm định và đánh giá

Kiểm định: Áp dụng mô hình để dự đoán class của tài liệu mới.

- Đầu vào:

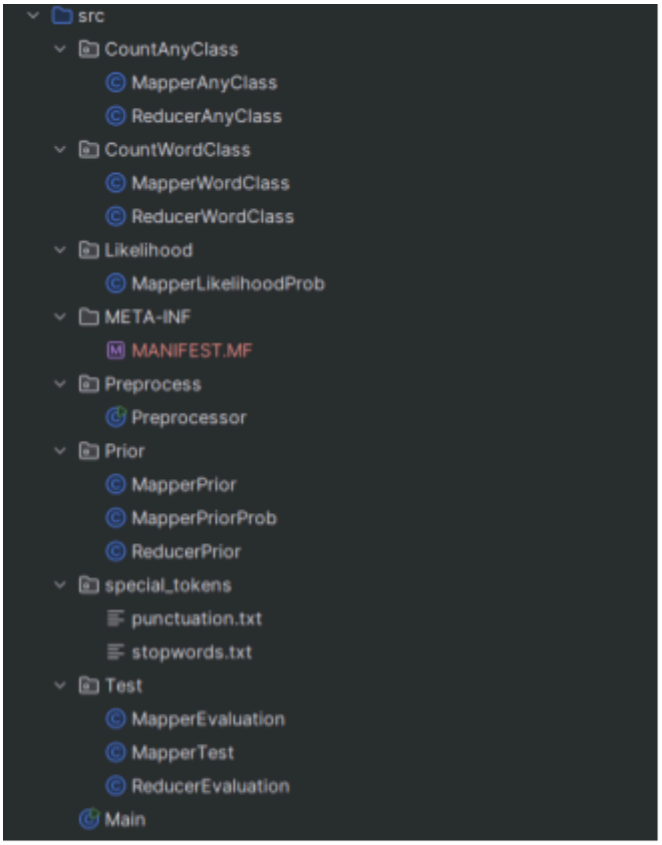
    - File test chứa mỗi dòng là một đoạn text và class thực tế.
- Quy trình MapReduce:
    - Mapper:
        - Tách nội dung text và class.
        - Thực hiện tiền xử lý và tokenize text.
        - Tính xác suất: $log(P(c) + \sum_{i=1}^{n} log(P(t_i \mid c))$
        - Output: (Text,actual class,predict class,probabilities)
    - Reducer: Không sử dụng.

Đánh giá: Sử dụng các chỉ số:

- Accuracy: Tỷ lệ dự đoán đúng.
- Precision: Độ chính xác trong số các dự đoán positive.
- Recall: Khả năng tìm ra tất cả các positive.
- F1 Score: Trung bình hài hòa của precision và recall.

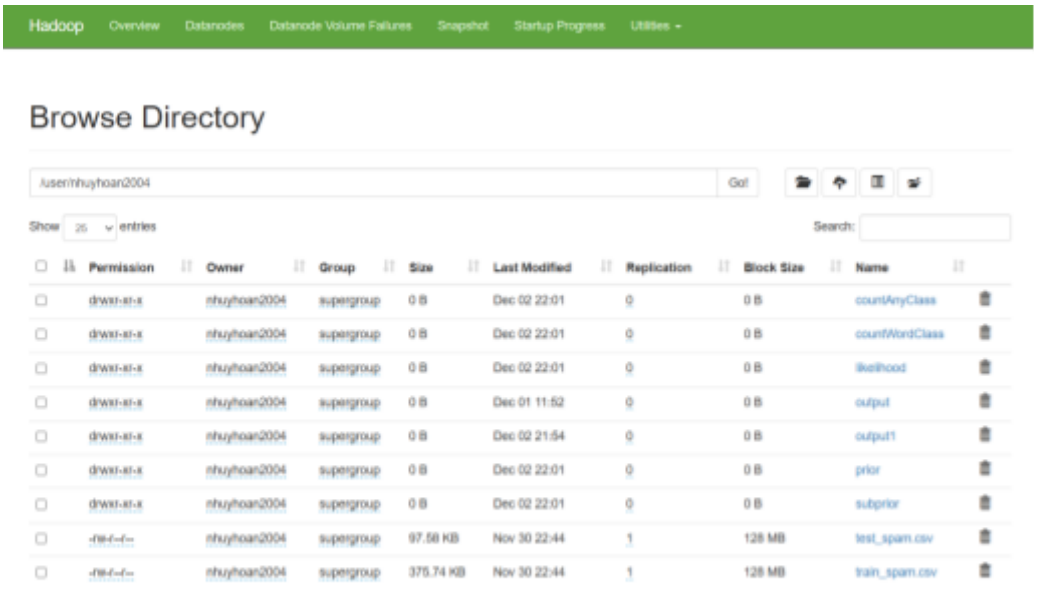
3.3 DEMO CHƯƠNG TRÌNH CÀI ĐẶT

Cấu trúc thư mục project

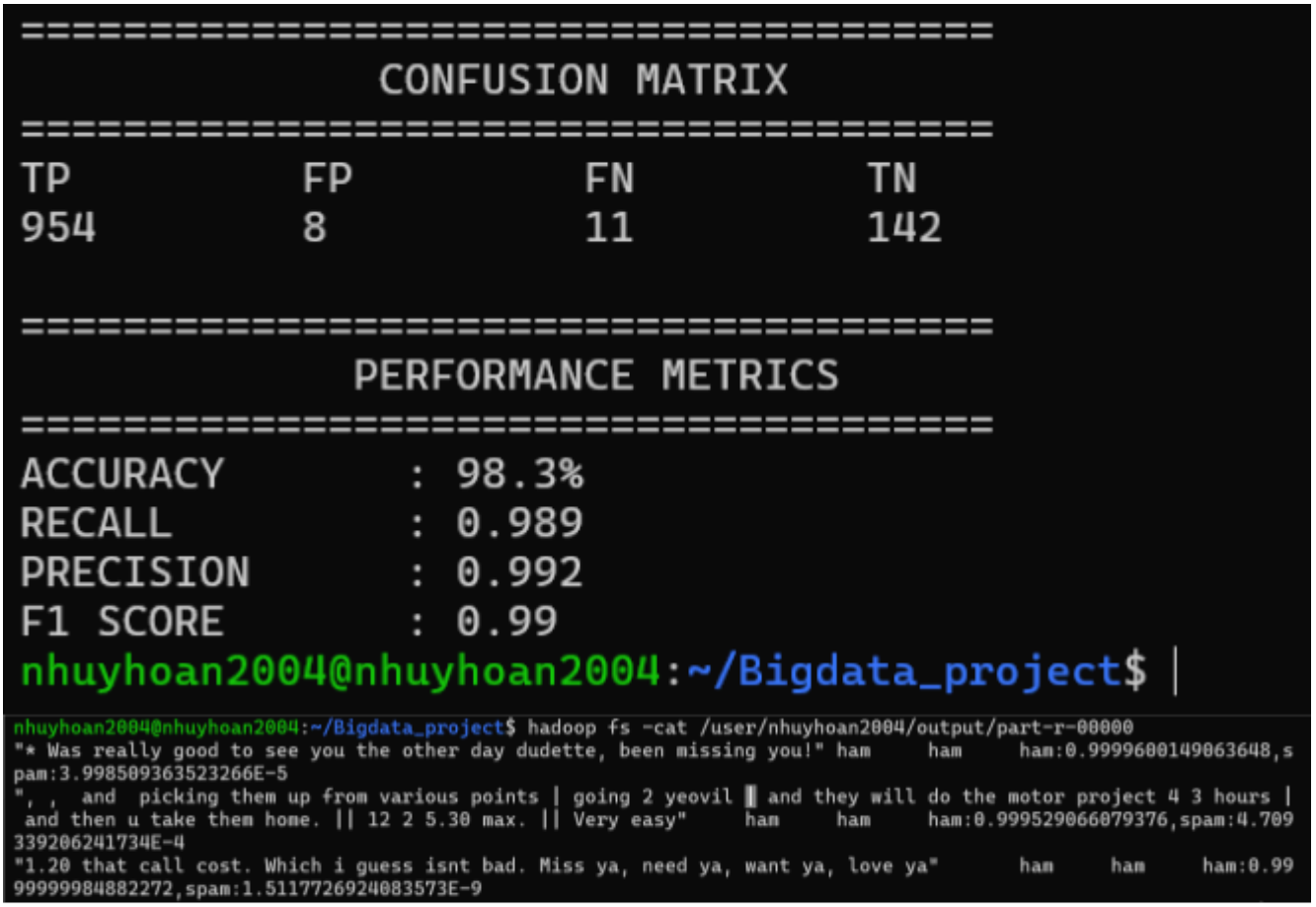


Hình 3.4: Cấu hình project

Kết quả chạy chương trình



Hình 3.5: Kết quả trên Hadoop UI



Hình 3.6: Confusion Matrix và Metrics trong phân loại văn bản với MapReduce

CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

4.1 KẾT LUẬN

Với sự bùng nổ của dữ liệu lớn ngày nay, các công cụ như Hadoop MapReduce trở nên vô cùng quan trọng để xử lý khối lượng dữ liệu khổng lồ một cách hiệu quả. Dự án của nhóm em đã thành công trong việc áp dụng kỹ thuật này để thực hiện thuật toán Naïve Bayes và xây dựng chương trình MapReduce cho bài toán phân loại văn bản.

Nhóm em đã đạt được một số thành quả đáng kể, bao gồm:

- Hiểu biết tổng quan về Big Data, Hadoop, MapReduce.
- Nắm vững chi tiết về thuật toán Naïve Bayes.
- Hiểu rõ về bài toán phân loại văn bản.
- Xây dựng và triển khai thành công chương trình.
- Đánh giá và phân tích kết quả chương trình.

Tuy nhiên, dự án vẫn còn một số hạn chế:

- Dữ liệu sử dụng chưa đủ lớn để đảm bảo tính chính xác cao.
- Một số mô hình trong Naïve Bayes (ví dụ Gaussian) chưa được triển khai.

4.2 HƯỚNG PHÁT TRIỂN

- Mở rộng thử nghiệm với bộ dữ liệu lớn, có thể lên đến hàng Gigabyte.
- Áp dụng vào các bài toán thực tế, ví dụ như hệ thống đề xuất sản phẩm được đánh giá cao cho khách hàng.

Tài liệu tham khảo

[1] https://www.ibm.com/topics/naivebayes#:~:text=Na%C3%AFve%20Bayes%20is%20part%20of,important%20to%20differentiate%20between%20classes.

[2] https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

[3] https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[4] https://hadoop.apache.org/

[5] https://en.wikipedia.org/wiki/MapReduce

[6] https://www.ibm.com/topics/mapreduce

[7] https://monkeylearn.com/text-classification/

[8] https://www.researchgate.net/publication/228057571_Tackling_the_Poor_Assumptions_of_Naive_Bayes_Text_Classifiers

[9] https://drive.google.com/file/d/1xrX398WskZYWGQZU_-EB5nfK7v0hfejs/view

Phân công nhiệm vụ

Ngô Huy Hoàn	Tìm hiểu về Hadoop, Mapreduce Tìm và phân tích data Cài đặt chương trình Chạy chương trình demo và đánh giá Viết báo cáo + làm slide (phần 2 +3)
Nguyễn Hải Nam	Tìm hiểu về Hadoop, Mapreduce Triển khai thuật toán Naive Bayes Cài đặt chương trình Chạy chương trình demo và đánh giá Viết báo cáo + làm slide (phần 1 +2)
Trần Phạm Hoàng	Tìm hiểu về Hadoop, Mapreduce Vẽ biểu đồ cho Mapreduce hóa Naive Bayes Cài đặt chương trình Chạy chương trình demo và đánh giá Viết báo cáo + làm slide (phần 2+3)
Tạ Nguyên Dũng	Tìm hiểu về Hadoop, Mapreduce Đưa ra ý tưởng phát triển Cài đặt chương trình Chạy chương trình demo và đánh giá Viết báo cáo + làm slide (phần 1+3)