KỸ THUẬT VÀ CÔNG NGHỆ DỮ LIỆU LỚN

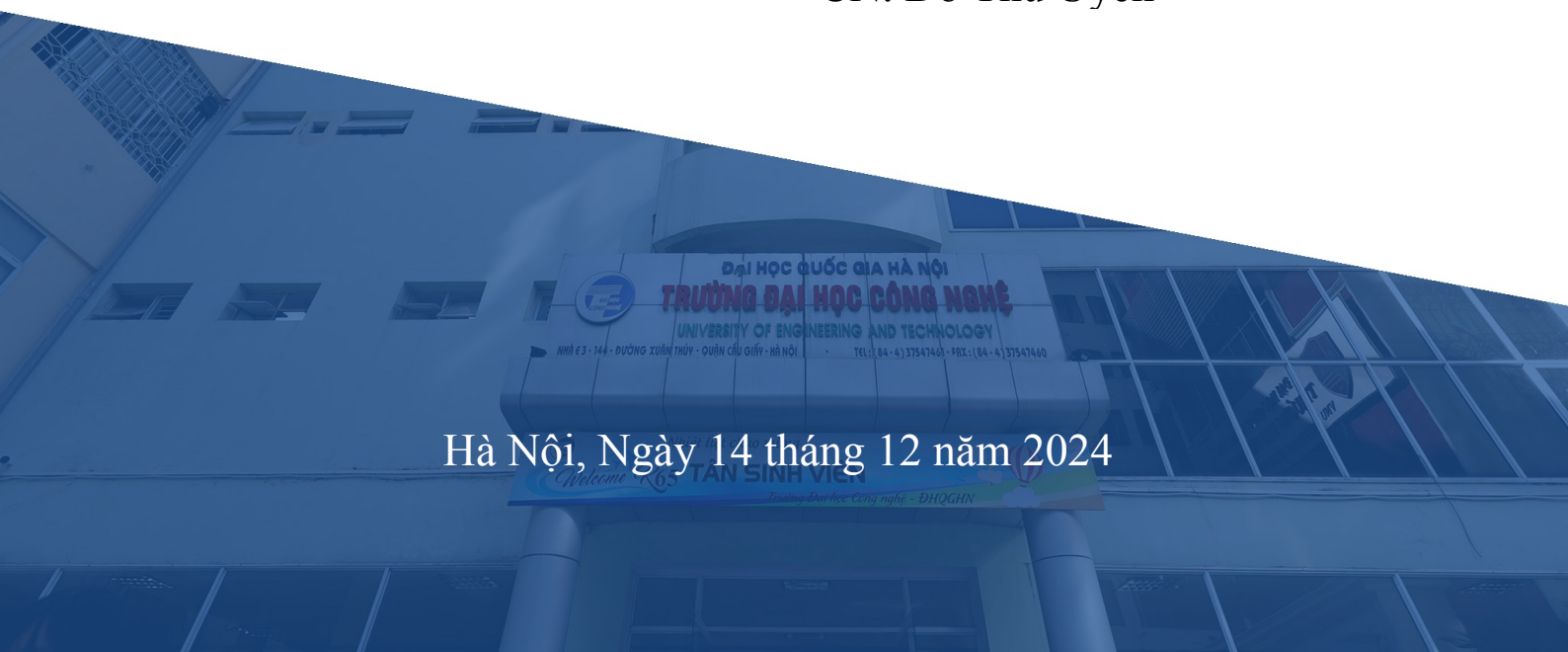
BÁO CÁO PROJECT

Text Sentiment Analysis in Hadoop and Spark

Nhóm 18

Họ và Tên	MSSV
Ngô Đức Hùng	22022652
Nguyễn Công Huynh	22022565
Nguyễn Văn Trường	22022571

Giảng viên hướng dẫn: TS. Trần Hồng Việt
CN. Đỗ Thu Uyên



Hà Nội, Ngày 14 tháng 12 năm 2024

Mục lục

1 | Giới thiệu

1.1 | Bối cảnh

Trong thời đại bùng nổ thông tin ngày nay, một lượng lớn dữ liệu văn bản được tạo ra hàng ngày trên internet, đặc biệt là trên các nền tảng mạng xã hội, diễn đàn, và các trang thương mại điện tử. Dữ liệu này chứa đựng những ý kiến, quan điểm, và cảm xúc phong phú của người dùng về vô số chủ đề, sản phẩm, và dịch vụ. Việc khai thác và phân tích hiệu quả nguồn dữ liệu tiềm năng này mang lại giá trị to lớn cho các doanh nghiệp, tổ chức trong việc thấu hiểu khách hàng, cải thiện sản phẩm, và đưa ra các quyết định kinh doanh chiến lược. Đây chính là lý do Phân tích cảm xúc (Sentiment Analysis) - một nhánh quan trọng của Xử lý ngôn ngữ tự nhiên - ngày càng thu hút được sự quan tâm rộng rãi.

Phân tích cảm xúc hướng đến việc xác định thái độ, quan điểm, hay cảm xúc (tích cực, tiêu cực, trung lập) ẩn chứa trong văn bản. Tuy nhiên, với khối lượng dữ liệu văn bản khổng lồ và không ngừng tăng lên, việc áp dụng các phương pháp phân tích cảm xúc truyền thống trở nên kém hiệu quả và tốn kém. Các thách thức về lưu trữ, xử lý, và phân tích dữ liệu lớn đòi hỏi những giải pháp mạnh mẽ và có khả năng mở rộng.

Chính vì vậy, dự án này hướng đến việc xây dựng một hệ thống phân tích cảm xúc hiệu quả, có khả năng xử lý dữ liệu văn bản lớn bằng cách tận dụng sức mạnh của các công nghệ dữ liệu lớn như Hadoop và Spark. Hadoop, với Hệ thống tệp phân tán Hadoop (HDFS) và mô hình lập trình MapReduce, cung cấp nền tảng lưu trữ và xử lý phân tán cho dữ liệu lớn. Spark, với khả năng xử lý in-memory và thư viện học máy MLlib, cho phép xây dựng và triển khai các mô hình học máy một cách nhanh chóng và hiệu quả trên dữ liệu lớn. Ngoài ra, dự án còn sử dụng Spark NLP, một thư viện NLP mạnh mẽ trên Spark, để nâng cao khả năng phân tích cảm xúc.

Toàn bộ mã nguồn của dự án được công khai tại: Github.

1.2 | Mục tiêu

Dự án này hướng đến các mục tiêu chính sau:

- Xây dựng một hệ thống phân tích cảm xúc có khả năng mở rộng sử dụng Hadoop và Spark.
- Tiền xử lý và làm sạch dữ liệu văn bản thô từ nguồn dữ liệu để chuẩn bị cho quá trình phân tích.
- Triển khai và đánh giá hiệu suất của bốn mô hình phân loại phổ biến: Logistic Regression, Naive Bayes, Random Forest và Support Vector Machine (SVM) sử dụng thư viện Spark MLlib.
- Áp dụng thư viện Spark NLP để tiền xử lý và trích xuất các đặc trưng ngôn ngữ cho việc phân tích cảm xúc.
- Phân tích và so sánh kết quả của các mô hình, từ đó rút ra những đánh giá và đề xuất cải tiến.
- Khám phá và làm rõ những lợi ích của việc áp dụng Hadoop và Spark (bao gồm cả MapReduce) trong việc lưu trữ và phân tích cảm xúc trên dữ liệu lớn.

1.3 | Phạm vi

Dự án này tập trung vào việc **phân tích cảm xúc** của khoảng 1,6 triệu tweets tiếng Anh đã được gán nhãn, với mục tiêu phân loại các tweets thành hai lớp: **tích cực** và **tiêu cực**. Để đạt được mục tiêu này, dự án sẽ sử dụng các kỹ thuật tiền xử lý văn bản cơ bản và triển khai bốn mô hình phân loại: **Logistic Regression**, **Naive Bayes**, **Random Forest** và **SVM**, tất cả đều thông qua thư viện **Spark MLlib**. Dữ liệu sẽ được lưu trữ và xử lý trên nền tảng **Hadoop** và **Spark**. Đặc biệt, dự án sẽ triển khai mô hình Naive Bayes bằng cả hai phương pháp, vừa sử dụng Spark MLlib vừa sử dụng **Hadoop MapReduce**, để hiểu rõ hơn về cách mô hình hoạt động trên môi trường phân tán. Ngoài ra, **Spark NLP** sẽ được sử dụng để tận dụng các kỹ thuật xử lý ngôn ngữ tự nhiên tiên tiến, hỗ trợ cho quá trình phân tích cảm xúc. Dự án không đi sâu vào việc so sánh hiệu năng với các công nghệ xử lý dữ liệu lớn khác ngoài Hadoop và Spark, cũng như không phát triển một ứng dụng người dùng cuối hoàn chỉnh.

2 | Tổng quan lý thuyết

2.1 | Phân tích cảm xúc

2.1.1 | Định nghĩa

Phân tích cảm xúc, còn được gọi là khai phá quan điểm (Opinion Mining), là một lĩnh vực nghiên cứu thuộc Xử lý ngôn ngữ tự nhiên (NLP), tập trung vào việc xác định và trích xuất thái độ, quan điểm, cảm xúc, hoặc đánh giá chủ quan của người viết/nói về một chủ thể, sự kiện, sản phẩm, hay dịch vụ nào đó từ dữ liệu văn bản. Mục tiêu chính là tự động phân loại cảm xúc trong văn bản thành các cực như tích cực, tiêu cực, hoặc trung lập.

2.1.2 | Phương pháp tiếp cận

Có nhiều phương pháp tiếp cận khác nhau trong phân tích cảm xúc, bao gồm:

- Phương pháp dựa trên từ điển cảm xúc: Phương pháp này dựa trên một từ điển cảm xúc được xây dựng sẵn, chứa các từ và cụm từ được gán nhãn với điểm số cảm xúc tương ứng. Văn bản đầu vào sẽ được phân tích và so khớp với từ điển để tính toán điểm số cảm xúc tổng thể.
- Phương pháp học máy: Phương pháp này sử dụng các thuật toán học máy để huấn luyện mô hình phân loại cảm xúc dựa trên dữ liệu đã được gán nhãn. Các mô hình phổ biến bao gồm Naive Bayes, Support Vector Machines (SVM), Logistic Regression, và gần đây là các mô hình Deep Learning.
- Phương pháp hỗn hợp: Kết hợp giữa phương pháp dựa trên từ điển và phương pháp học máy để tận dụng ưu điểm của cả hai phương pháp.

2.1.3 | Ứng dụng

Phân tích cảm xúc có rất nhiều ứng dụng thực tế trong nhiều lĩnh vực khác nhau, bao gồm:

- Quản lý danh tiếng thương hiệu: Theo dõi và phân tích đánh giá của khách hàng về sản phẩm, dịch vụ, và thương hiệu trên mạng xã hội, diễn đàn, và các trang web đánh giá.
- Nghiên cứu thị trường: Phân tích ý kiến khách hàng để hiểu rõ hơn về nhu cầu, sở thích, và xu hướng thị trường.
- Dịch vụ khách hàng: Tự động phân loại phản hồi của khách hàng để ưu tiên xử lý các vấn đề tiêu cực và cải thiện chất lượng dịch vụ.
- Phân tích chính trị: Phân tích ý kiến của cử tri về các ứng cử viên, đảng phái, và chính sách.
- Giám sát mạng xã hội: Theo dõi và phân tích các xu hướng, chủ đề nóng, và tâm lý cộng đồng trên mạng xã hội.

2.2 | Dữ liệu lớn

2.2.1 | Khái niệm

Dữ liệu lớn (Big Data) là thuật ngữ dùng để chỉ các tập dữ liệu có khối lượng lớn, tốc độ tăng trưởng nhanh, và đa dạng về định dạng, vượt quá khả năng xử lý của các hệ thống quản trị cơ sở dữ liệu truyền thống. Dữ liệu lớn thường được đặc trưng bởi 5V:

- Volume (Khối lượng)
- Velocity (Tốc độ)
- Variety (Đa dạng)
- Veracity (Tính xác thực)
- Value (Giá trị)

2.2.2 | Thách thức trong việc xử lý dữ liệu lớn

- Lưu trữ: Khó khăn trong việc lưu trữ lượng dữ liệu khổng lồ và không ngừng tăng lên.
- Xử lý: Khó khăn trong việc xử lý dữ liệu với tốc độ nhanh và hiệu quả.
- Phân tích: Khó khăn trong việc trích xuất thông tin hữu ích từ dữ liệu lớn và phức tạp.
- Bảo mật: Khó khăn trong việc đảm bảo an toàn và bảo mật cho dữ liệu lớn.

2.3 | Hadoop

2.3.1 | Giới thiệu

Hadoop là một framework mã nguồn mở được thiết kế để lưu trữ và xử lý dữ liệu lớn một cách phân tán trên các cụm máy tính thông thường. Hadoop bao gồm các thành phần chính sau:

- Hadoop Distributed File System (HDFS): Hệ thống tệp phân tán, cung cấp khả năng lưu trữ dữ liệu lớn một cách tin cậy và có khả năng chịu lỗi cao.
- MapReduce: Mô hình lập trình cho phép xử lý song song dữ liệu lớn trên các cụm máy tính.
- YARN (Yet Another Resource Negotiator): Quản lý tài nguyên và lập lịch cho các ứng dụng chạy trên Hadoop.
- Các dự án khác: HBase, Hive, Pig, ..., cung cấp các chức năng bổ sung cho việc lưu trữ, truy vấn, và phân tích dữ liệu.

2.3.2 | Ưu điểm và hạn chế

1. Ưu điểm
   - Khả năng mở rộng: Có thể mở rộng để xử lý lượng dữ liệu khổng lồ bằng cách thêm các nút vào cụm máy tính.
   - Khả năng chịu lỗi: Có khả năng tự động phục hồi khi một nút trong cụm bị lỗi.
   - Chi phí thấp: Sử dụng các máy tính thông thường thay vì các máy chủ đắt tiền.
2. Hạn chế
   - Độ trễ cao: MapReduce không phù hợp cho các ứng dụng yêu cầu độ trễ thấp.
   - Phức tạp: Việc lập trình MapReduce có thể phức tạp và khó khăn.

2.4 | Spark

2.4.1 | Giới thiệu

Apache Spark là một framework mã nguồn mở, cung cấp một nền tảng thống nhất cho việc xử lý dữ liệu lớn với tốc độ nhanh và dễ sử dụng. Spark khắc phục một số hạn chế của Hadoop MapReduce, đặc biệt là về độ trễ và tính phức tạp. Các thành phần chính của Spark bao gồm:

- Spark Core: Cung cấp các chức năng cơ bản của Spark, bao gồm lập lịch tác vụ, quản lý bộ nhớ, và phục hồi lỗi.
- Spark SQL: Cho phép truy vấn dữ liệu có cấu trúc sử dụng SQL hoặc DataFrame API.
- Spark Streaming: Cho phép xử lý dữ liệu luồng thời gian thực.
- Spark MLlib: Thư viện học máy cung cấp các thuật toán và công cụ cho các tác vụ học máy phổ biến.
- Spark NLP: Thư viện xử lý ngôn ngữ tự nhiên cung cấp các công cụ và mô hình để thực hiện các tác vụ NLP như tokenization, stemming, lemmatization, và named entity recognition.
- GraphX: Thư viện cho phép xử lý đồ thị.

2.4.2 | Ưu điểm và hạn chế so với Hadoop MapReduce

1. Ưu điểm
   - Tốc độ: Xử lý dữ liệu nhanh hơn MapReduce nhờ khả năng xử lý in-memory.
   - Dễ sử dụng: Cung cấp các API đơn giản và dễ sử dụng cho nhiều ngôn ngữ lập trình như Scala, Java, Python, R.
   - Tính linh hoạt: Hỗ trợ nhiều loại tác vụ xử lý dữ liệu khác nhau (batch, streaming, machine learning, graph processing) trên cùng một nền tảng.
2. Hạn chế
   - Chi phí bộ nhớ: Yêu cầu nhiều bộ nhớ hơn MapReduce do xử lý in-memory.
   - Phụ thuộc vào YARN: Thường được triển khai trên cụm Hadoop YARN để quản lý tài nguyên.

2.5 | Các mô hình phân loại trong Spark MLlib

Phần này giới thiệu bốn mô hình phân loại được sử dụng trong dự án, bao gồm Logistic Regression, Naive Bayes, Random Forest và Support Vector Machine (SVM). Lý do lựa chọn các mô hình này dựa trên các tiêu chí sau: tính phổ biến trong các bài toán phân tích cảm xúc, khả năng mở rộng tốt với dữ liệu lớn, và sự hỗ trợ mạnh mẽ từ thư viện Spark MLlib. Cả bốn mô hình đều đã được chứng minh là có hiệu quả trong nhiều nghiên cứu về phân tích cảm xúc và có thể cung cấp các kết quả baseline tốt để so sánh và đánh giá. Hơn nữa, việc lựa chọn các mô hình có nguyên lý hoạt động khác nhau (hồi quy, xác suất, ensemble learning, và phương pháp dựa trên margin) sẽ cho phép chúng ta có cái nhìn đa chiều hơn về hiệu suất phân loại trên tập dữ liệu tweet.

2.5.1 | Logistic Regression

Logistic Regression là một thuật toán học máy được sử dụng cho các bài toán phân loại nhị phân (binary classification). Mô hình này ước tính xác suất của một đối tượng thuộc về một lớp cụ thể dựa trên các biến đầu vào.

- Cách hoạt động: Sử dụng hàm sigmoid để ánh xạ đầu ra của một hàm tuyến tính vào khoảng (0, 1), biểu diễn xác suất.
- Ưu điểm: Đơn giản, dễ hiểu, và hiệu quả.
- Hạn chế: Giả định mối quan hệ tuyến tính giữa các biến đầu vào và đầu ra.
- Lý do lựa chọn: Logistic Regression được chọn vì tính đơn giản, dễ triển khai và diễn giải. Nó cung cấp một baseline model tốt để so sánh với các mô hình phức tạp hơn. Khả năng xử lý dữ liệu lớn của Logistic Regression trong Spark MLlib cũng là một yếu tố quan trọng.

2.5.2 | Naive Bayes

Naive Bayes là một thuật toán phân loại dựa trên định lý Bayes, với giả định ”ngây thơ” (naive) rằng các đặc trưng là độc lập với nhau.

- Cách hoạt động: Tính toán xác suất của một đối tượng thuộc về từng lớp dựa trên xác suất xuất hiện của các đặc trưng trong từng lớp.
- Ưu điểm: Nhanh, hiệu quả, và yêu cầu ít dữ liệu huấn luyện.
- Hạn chế: Giả định về tính độc lập giữa các đặc trưng thường không đúng trong thực tế.
- Lý do lựa chọn: Naive Bayes được chọn vì tốc độ huấn luyện nhanh và khả năng mở rộng tốt. Nó đặc biệt phù hợp với dữ liệu văn bản, nơi số lượng đặc trưng thường rất lớn. Việc giả định các từ trong tweet là độc lập với nhau, tuy ”ngây thơ”, nhưng thường mang lại hiệu quả đáng kể trong thực tế, đặc biệt là khi có ít thời gian và tài nguyên tính toán.

2.5.3 | Random Forest

Random Forest là một thuật toán học máy thuộc nhóm ensemble learning, xây dựng nhiều cây quyết định (Decision tree) và kết hợp kết quả dự đoán của các cây này để đưa ra kết quả cuối cùng.

- Cách hoạt động: Mỗi cây quyết định được huấn luyện trên một tập con ngẫu nhiên của dữ liệu huấn luyện và sử dụng một tập con ngẫu nhiên của các đặc trưng. Kết quả dự đoán của các cây được kết hợp bằng cách lấy trung bình (đối với bài toán hồi quy) hoặc bỏ phiếu (đối với bài toán phân loại).
- Ưu điểm: Độ chính xác cao, ít bị overfitting, và có khả năng xử lý dữ liệu nhiễu.
- Hạn chế: Khó diễn giải hơn so với các mô hình đơn giản như Logistic Regression hay Naive Bayes, đặc biệt là cần nhiều tài nguyên tính toán.
- Lý do lựa chọn: Random Forest được chọn vì khả năng đạt được độ chính xác cao và khả năng chống chịu với overfitting. Việc kết hợp nhiều cây quyết định giúp mô hình trở nên ổn định và đáng tin cậy hơn. Mặc dù Random Forest phức tạp hơn, nhưng nó thường mang lại hiệu suất tốt hơn so với Logistic Regression và Naive Bayes, đặc biệt là trên các tập dữ liệu lớn và phức tạp.

2.5.4 | Support Vector Machine (SVM)

Support Vector Machine (SVM) là một thuật toán học máy hiệu quả cho cả bài toán phân loại và hồi quy. Trong phân loại nhị phân, SVM tìm kiếm một siêu phẳng (hyperplane) tối ưu để phân tách hai lớp dữ liệu, sao cho khoảng cách từ siêu phẳng đó đến các điểm dữ liệu gần nhất (support vectors) của mỗi lớp là lớn nhất (maximum margin).

- Cách hoạt động: SVM tìm kiếm siêu phẳng tối ưu trong không gian đặc trưng để phân tách hai lớp dữ liệu. Trong trường hợp dữ liệu không phân tách tuyến tính, SVM sử dụng kỹ thuật kernel để ánh xạ dữ liệu sang không gian đặc trưng mới, nơi dữ liệu có thể được phân tách tuyến tính.
- Ưu điểm:
  - Hiệu quả trong không gian chiều cao.
  - Chỉ sử dụng một tập con của các điểm huấn luyện (support vectors) trong hàm quyết định, do đó tiết kiệm bộ nhớ.
  - Linh hoạt: Có thể sử dụng các kernel function khác nhau để phù hợp với các loại dữ liệu khác nhau.
- Hạn chế:
  - Nếu số lượng đặc trưng lớn hơn nhiều so với số lượng mẫu, cần cẩn thận trong việc lựa chọn kernel function và tham số regularization để tránh overfitting.
  - SVM không trực tiếp cung cấp ước tính xác suất, mà cần sử dụng phương pháp cross-validation tốn kém hơn.
- Lý do lựa chọn: SVM được chọn vì khả năng phân loại tốt, đặc biệt là trong các trường hợp dữ liệu không phân tách tuyến tính. SVM cũng là một trong những mô hình phổ biến trong phân tích cảm xúc và đã được chứng minh là có hiệu quả trong nhiều nghiên cứu. Việc bổ sung SVM vào danh sách các mô hình phân loại giúp tăng tính đa dạng và cho phép so sánh hiệu suất giữa các phương pháp tiếp cận khác nhau.

2.6 | Spark NLP

2.6.1 | Giới thiệu

Spark NLP là một thư viện xử lý ngôn ngữ tự nhiên (NLP) mã nguồn mở, được xây dựng trên nền tảng Apache Spark, cung cấp các API đơn giản, hiệu suất cao và có khả năng mở rộng cho các tác vụ NLP. Spark NLP hỗ trợ nhiều mô hình NLP hiện đại, bao gồm các mô hình Deep Learning, cho phép thực hiện các nhiệm vụ như phân tích cảm xúc, nhận dạng thực thể, phân loại văn bản, v.v. với độ chính xác cao. Một số ưu điểm nổi bật của Spark NLP bao gồm:

- Hiệu suất cao: Được xây dựng trên nền tảng Spark, Spark NLP có thể tận dụng khả năng tính toán phân tán để xử lý dữ liệu lớn một cách hiệu quả.
- Độ chính xác cao: Hỗ trợ nhiều mô hình Deep Learning tiên tiến, cho phép đạt được độ chính xác cao trong các tác vụ NLP.
- Khả năng mở rộng: Dễ dàng mở rộng để xử lý các tập dữ liệu lớn bằng cách thêm các nút vào cụm Spark.
- Hỗ trợ nhiều ngôn ngữ: Spark NLP hỗ trợ nhiều ngôn ngữ khác nhau, bao gồm cả tiếng Anh.
- Dễ sử dụng: Cung cấp các API đơn giản và trực quan, giúp cho việc triển khai các mô hình NLP trở nên dễ dàng hơn.

2.6.2 | Thành phần chính

Spark NLP cung cấp nhiều thành phần quan trọng cho xử lý ngôn ngữ tự nhiên. Trong dự án này, chúng ta đặc biệt quan tâm đến 'ClassifierDLApproach'

ClassifierDLApproach: Đây là một thành phần cho phép huấn luyện mô hình phân loại văn bản sử dụng mạng nơ-ron sâu (Deep Learning). Nó nhận đầu vào là văn bản đã được mã hóa thành các vector đặc trưng và nhãn tương ứng, sau đó huấn luyện một mô hình deep learning để phân loại văn bản vào các lớp khác nhau. ClassifierDLApproach cung cấp các tùy chọn để tinh chỉnh kiến trúc mạng nơ-ron, hàm mất mát, và thuật toán tối ưu để đạt được hiệu suất tốt nhất trên tập dữ liệu cụ thể.

2.7 | Naive Bayes trong Hadoop MapReduce

2.7.1 | Giới thiệu

Mặc dù Spark MLlib cung cấp một triển khai hiệu quả của thuật toán Naive Bayes, việc xây dựng và triển khai Naive Bayes bằng Hadoop MapReduce mang lại hiểu biết sâu sắc về cách mô hình này hoạt động trong một môi trường xử lý phân tán. Cách tiếp cận này tận dụng các nguyên tắc cơ bản của MapReduce để tính toán các xác suất cần thiết trong thuật toán Naive Bayes.

2.7.2 | Quá trình triển khai

Triển khai Naive Bayes trong Hadoop MapReduce thường bao gồm các bước sau:

1. Map (Giai đoạn ánh xạ):
   - Input: Mỗi mapper nhận một phần của tập dữ liệu huấn luyện, trong đó mỗi mẫu dữ liệu bao gồm các đặc trưng (ví dụ: các từ trong một tweet) và nhãn cảm xúc tương ứng.
   - Output: Các mapper sẽ xuất ra các cặp key-value. Key là tổ hợp của (nhãn, từ) và value là 1. Điều này cho phép thống kê số lần xuất hiện của mỗi từ trong từng lớp cảm xúc.
2. Shuffle and Sort (Giai đoạn xáo trộn và sắp xếp):
   - Hadoop tự động xáo trộn và sắp xếp các cặp key-value theo key, đưa các cặp có cùng key đến cùng một reducer.
3. Reduce (Giai đoạn rút gọn):
   - Input: Các reducer nhận các cặp key-value có cùng key.
   - Output: Mỗi reducer tính tổng số lần xuất hiện của mỗi từ trong mỗi lớp (tần suất xuất hiện của từ cho mỗi nhãn) và xuất ra một cặp key-value mới. Key là (nhãn, từ) và value là số lần xuất hiện của từ trong nhãn đó.
4. Tính toán xác suất (Sau giai đoạn MapReduce):
   - Từ output của các reducer, tính toán các xác suất cần thiết cho thuật toán Naive Bayes:
     - P(label): Xác suất của mỗi lớp cảm xúc.
     - P(word|label): Xác suất của mỗi từ xuất hiện trong mỗi lớp cảm xúc.
   - Các xác suất này có thể được lưu trữ để sử dụng cho việc phân loại.
5. Phân loại (Sử dụng các xác suất đã tính toán):
   - Cho một tweet mới, sử dụng các xác suất đã tính toán để dự đoán lớp cảm xúc của tweet đó.

2.7.3 | Ưu điểm và hạn chế

1. Ưu điểm
   - Hiểu rõ cơ chế hoạt động: Giúp hiểu rõ hơn về cơ chế hoạt động của Naive Bayes trên môi trường phân tán.
   - Khả năng tùy chỉnh: Cung cấp nhiều tùy chỉnh hơn so với việc sử dụng các thư viện có sẵn.
2. Hạn chế
   - Phức tạp: Cần nhiều công sức để triển khai so với việc sử dụng thư viện Spark MLlib.
   - Tốn thời gian: Quá trình xử lý có thể tốn thời gian hơn so với các triển khai được tối ưu hóa, đặc biệt là khi hoạt động trên máy ảo (dùng trong dự án) bị hạn chế về tài nguyên và cấu hình.

3 | Dữ liệu và tiền xử lý, EDA

3.1 | Mô tả dữ liệu

- Nguồn gốc dữ liệu: Dữ liệu sử dụng trong project này là tập dữ liệu Sentiment140 + Kaggle đã được tiền xử lý qua, bao gồm 1,6 triệu tweet được thu thập thông qua Twitter API và đã được gán nhãn cảm xúc 0 (tiêu cực) hoặc 1 (tích cực).
- Cấu trúc dữ liệu: Được lưu dưới định dạng csv. Mỗi bản ghi trong tập dữ liệu bao gồm 4 trường thông tin: **ItemID** (mã định danh tweet), **Sentiment** (nhãn cảm xúc), **SentimentSource** (nguồn lấy dữ liệu), **SentimentText** (nội dung tweet), ví dụ:

ItemID	Sentiment	SentimentSource	SentimentText
1	0	Sentiment140	is so sad for my APL friend.....
2	0	Sentiment140	I missed the New Moon trailer...
3	1	Sentiment140	omg its already 7:30 :O
...	...	...	...

Bảng 3.1: Tổng quan dữ liệu

- Kích thước dữ liệu: Tập dữ liệu gần 1,6 triệu tweets, với tổng dung lượng khoảng 150MB.

3.2 | Tiền xử lý dữ liệu

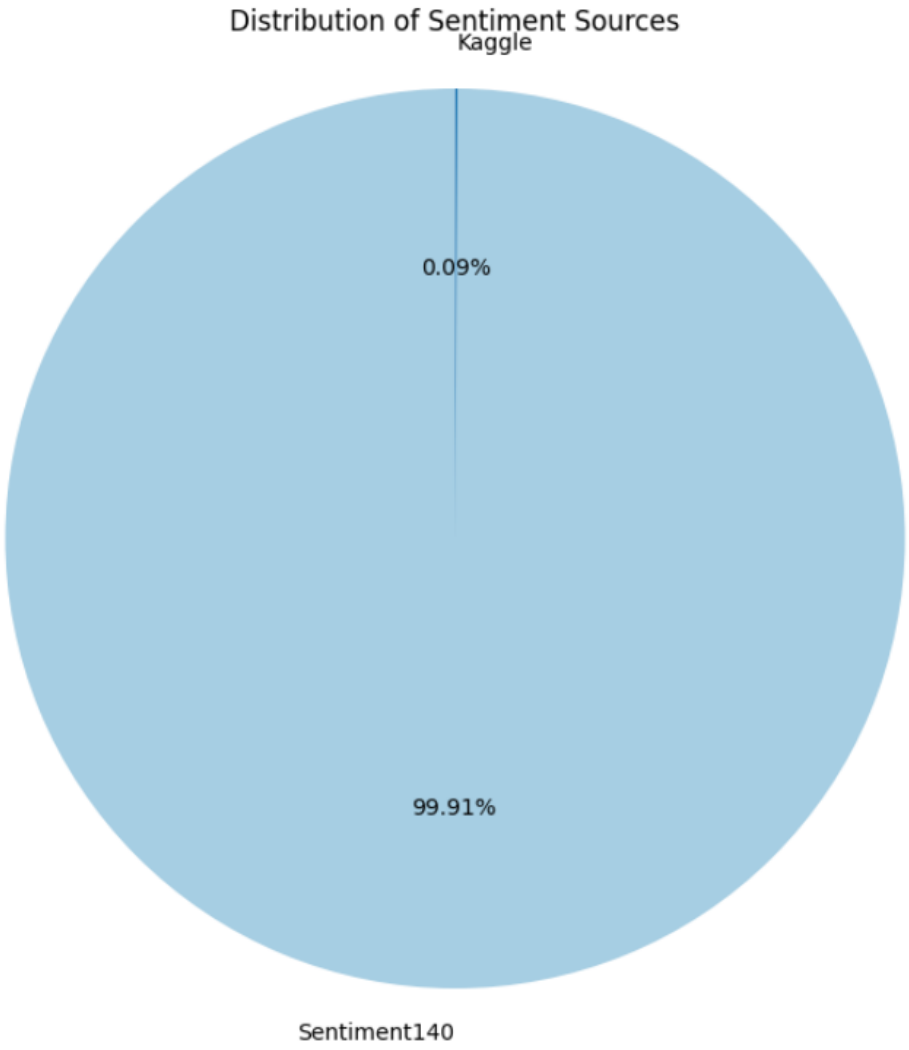
Quá trình tiền xử lý dữ liệu có thể xem tại đây: Preprocessing

- Làm sạch dữ liệu:
  - Kiểm tra giá trị thiếu và trùng lặp để xử lý.
  - Loại bỏ những tweet quá dài (>180 từ, được tính là ngoại lệ, không đáng kể số lượng).
  - Loại bỏ các kí tự đặc biệt, số, hashtag, username, URL, khoảng trắng ở đầu và cuối câu.
- Chuẩn hóa văn bản:
  - Chuyển văn bản từ chữ hoa về chữ thường.
  - Stemming/Lemmatization: Ví dụ stemming là chuyển ”running” thành ”run” hoặc Lemmatization là chuyển ”better” thành ”good”.
- Loại bỏ stop words: Loại bỏ các từ phổ biến nhưng ít mang ý nghĩa như các mạo từ ’a’, ’an’, ’the’, ...
- Tách văn bản thành các từ (tokens).

3.3 | EDA

- Phân phối về nguồn dữ liệu:

SentimentSource	Count
Sentiment140	1577278
Kaggle	1349

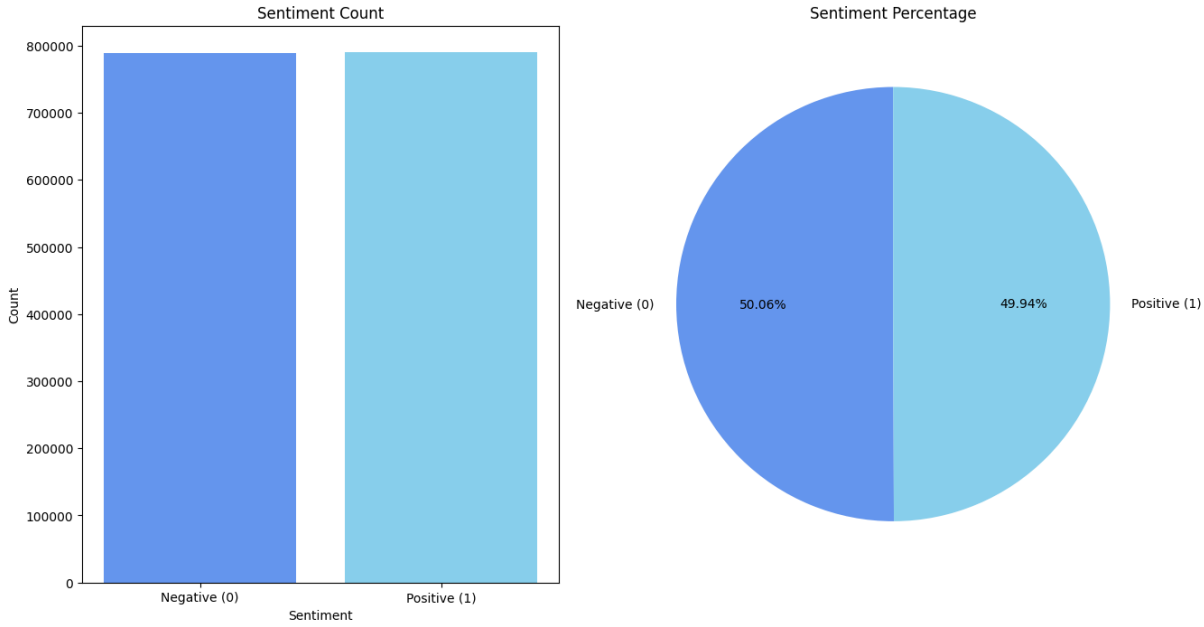


Hình 3.1: Tỉ trọng nguồn dữ liệu

Dữ liệu sử dụng trong dự án này chủ yếu dựa trên tập dữ liệu Sentiment140, bao gồm 1,577,278 tweets (chiếm 99.91% tổng số dữ liệu). Ngoài ra, tập dữ liệu còn bao gồm 1,349 tweets (chiếm 0.09%) từ nguồn Kaggle. Do tỷ lệ của nguồn Kaggle rất nhỏ nên sẽ không ảnh hưởng tới tính tổng quát của dữ liệu.

- Phân phối cảm xúc:

Sentiment	Count	Percentage
1	790185	50.05
0	788442	49.95



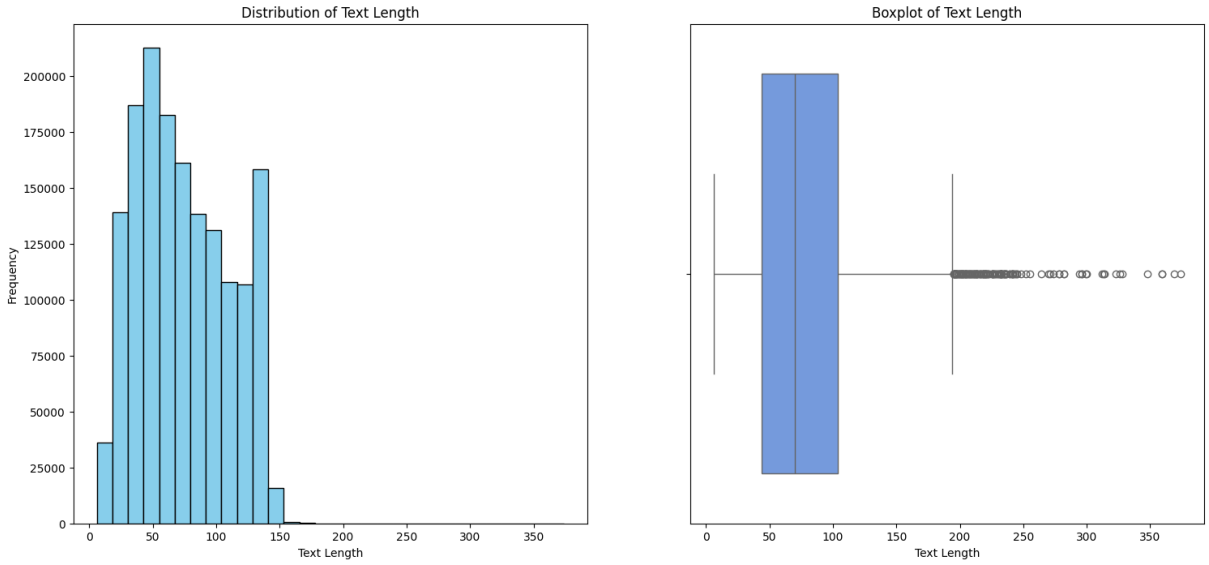
Hình 3.2: Phân phối và tỉ trọng cảm xúc

Tập dữ liệu có sự phân bố cảm xúc gần như cân bằng giữa hai nhãn Positive và Negative. Điều này là tốt và quan trọng cho việc huấn luyện mô hình học máy, vì nó giúp tránh việc mô hình bị thiên vị (biased) về một nhãn nào đó. Sự cân bằng này cho thấy tập dữ liệu Sentiment140 được gán nhãn khá tốt cho mục đích phân tích cảm xúc.

- Thống kê độ dài văn bản

Summary	Text Length
count	1578627
mean	74.6273
stddev	36.1355
min	6
max	374

  - Độ dài trung bình của các tweets trong tập dữ liệu là khoảng 74.6 ký tự, với phần lớn các tweets có độ dài từ 50 đến 100 ký tự, chỉ có một chút các tweets có độ dài lớn hơn 150 ký tự.
  - Phân phối độ dài văn bản lệch phải, với một số ít tweets có độ dài lớn bất thường, như các tweet khoảng 200 ký tự trở lên.



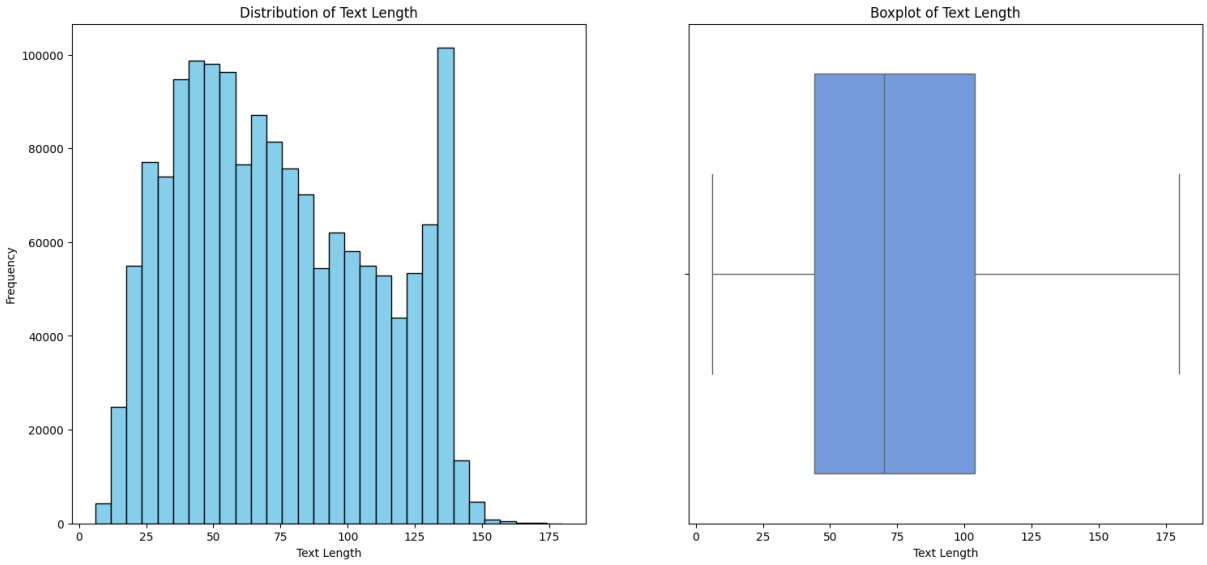
Hình 3.3: Phân phối và Boxplot của độ dài văn bản

- Ta sẽ tiến hành loại bỏ những tweet có lượng ký tự từ 180 trở lên để loại bỏ những giá trị ngoại lai (outliers) và tập trung vào phần lớn dữ liệu.

- Thống kê độ dài văn bản sau khi loại bỏ outliers

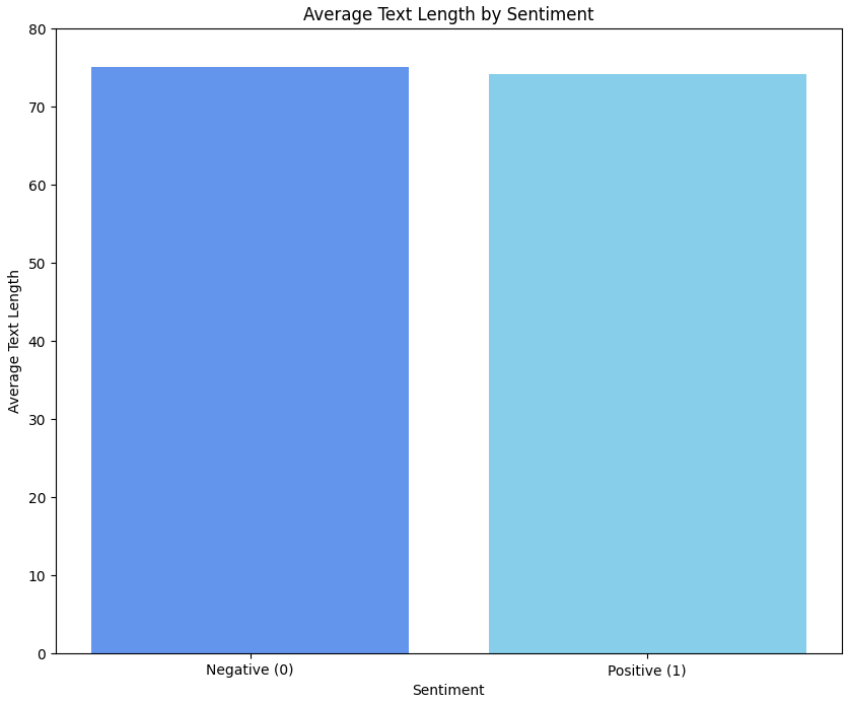
Summary	Text Length
count	1578405
mean	74.6073
stddev	36.0956
min	6
max	180

- Việc loại bỏ các tweets có độ dài trên 180 ký tự đã loại bỏ phần lớn outliers, làm cho phân phối độ dài tweets tập trung hơn
- Số lượng tweet chỉ giảm đi 222 tweets, không phải là một con số lớn đối với tập dữ liệu.
- Các đặc trưng chính của phân phối (lệch phải, giá trị trung bình, trung vị) vẫn được giữ nguyên sau khi loại bỏ outliers.
- Sự thay đổi đáng kể nhất là ở giá trị lớn nhất (max) và sự thu hẹp nhẹ của khoảng tứ phân vị (IQR).
- Nhìn chung, việc loại bỏ outliers đã làm cho tập dữ liệu ”sạch” hơn, tập trung hơn vào phần lớn dữ liệu, và có thể có lợi cho việc huấn luyện mô hình.



Hình 3.4: Phân phối và Boxplot của độ dài văn bản sau khi loại bỏ outliers

- Độ dài văn bản trung bình dựa theo cảm xúc



Hình 3.5: Độ dài văn bản trung bình dựa theo cảm xúc

- Hình 3.5 cho thấy không có sự khác biệt đáng kể về độ dài văn bản trung bình giữa các tweets có cảm xúc tiêu cực và tích cực.
- Dựa trên biểu đồ hình 3.5, có thể kết luận rằng độ dài trung bình của tweet không cung cấp thông tin hữu ích để phân biệt giữa cảm xúc tích cực và tiêu cực trong tập dữ liệu.

- Phân tích tần suất từ
  Để hiểu rõ hơn về nội dung của tập dữ liệu, ta tiến hành phân tích tần suất xuất hiện của các từ. Hai phương pháp trực quan hóa được sử dụng là Word Cloud và biểu đồ tần suất.

  - Word Cloud toàn bộ tập dữ liệu

Hình 3.6: Word Cloud của các từ phổ biến trong toàn bộ tập dữ liệu

Hình 3.6 hiển thị Word Cloud được tạo ra từ toàn bộ tập dữ liệu (sau khi đã loại bỏ stop words, dấu câu và thực hiện các bước tiền xử lý). Word Cloud cho thấy các từ như ”im”, ”get”, ”day”, ”good”, ”go”, ”like”, ”love”, ”today”, ”work” xuất hiện với tần suất cao, được thể hiện bằng kích thước lớn.
Điều này cho thấy một số chủ đề phổ biến trong các tweets, bao gồm các hoạt động hàng ngày (”day”, ”work”), cảm xúc (”good”, ”love”), và các từ thường được sử dụng trong giao tiếp (”get”, ”go”, ”like”). Cũng có thể nhận thấy các từ viết tắt, đặc trưng của ngôn ngữ mạng xã hội như ”u” (tức là ”you”).

  - Word Cloud cho Positive Sentiment

Hình 3.7: Word Cloud của các từ phổ biến trong các tweets tích cực

Hình 3.7 hiển thị Word Cloud được tạo ra từ các tweets được gán nhãn positive (sau khi đã loại bỏ stop words, dấu câu và thực hiện các bước tiền xử lý). Có thể nhận thấy các từ mang cảm xúc tích cực xuất hiện với tần suất cao và kích thước lớn như: ”good”, ”love”, ”great”, ”happy”, ”thanks”, ”like”, ”lol”. Ngoài ra, các từ liên quan đến hoạt động thường ngày vẫn xuất hiện như: ”day”, ”get”, ”go”, ”today”, ”work”.

- Word Cloud cho Negative Sentiment

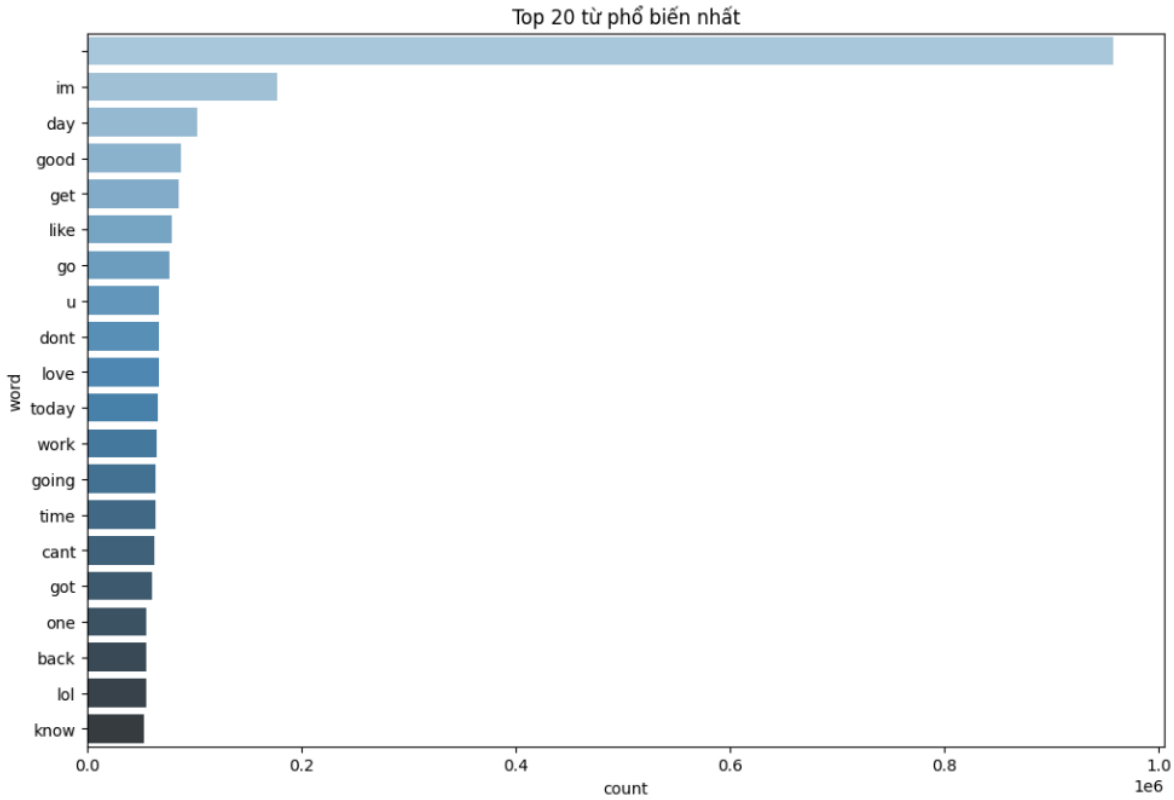
Hình 3.8: Word Cloud của các từ phổ biến trong các tweets tiêu cực

Hình 3.8 hiển thị Word Cloud được tạo ra từ các tweets được gán nhãn negative (sau khi đã loại bỏ stop words, dấu câu và thực hiện các bước tiền xử lý). Khác với Word Cloud của positive tweets, ở đây xuất hiện các từ mang cảm xúc tiêu cực với tần suất cao như: ”sad”, ”miss”, ”sorry”, ”bad”, ”hate”. Các từ như ”cant”, ”dont”, ”go”, ”get” cũng xuất hiện nhiều, có thể mang hàm ý tiêu cực trong một số ngữ cảnh.

- Top 20 từ phổ biến nhất

  Hình 3.9 bổ sung thêm thông tin chi tiết hơn bằng cách hiển thị 20 từ phổ biến nhất và tần suất cụ thể của chúng dưới dạng biểu đồ cột ngang. Dễ dàng nhận thấy rằng dấu khoảng cách (” ”) là ký tự (có thể coi là từ đặc biệt) xuất hiện nhiều nhất với tần suất gần 1 triệu, cao vượt trội so với các từ còn lại. Đứng thứ hai là từ ”im” (có thể là ”I’m”) với tần suất nhỏ hơn đáng kể. Các từ tiếp theo trong danh sách bao gồm ”day”, ”good”, ”get”, ”like”, ”go”, ”u”, ”dont”, ”love”, ”today”, ”work”, ”going”, ”time”, ”cant”, ”got”, ”one”, ”back”, ”lol”, ”know”.

- Nhận xét chung:
  - Word Cloud cho thấy sự khác biệt trong việc sử dụng từ ngữ giữa các tweets tích cực và tiêu cực.
  - Các từ tích cực như ”good”, ”love”, ”great” xuất hiện nhiều trong các tweets tích cực, trong khi các từ tiêu cực như ”sad”, ”miss”, ”sorry” xuất hiện nhiều trong các tweets tiêu cực.
  - Việc phân tích Word Cloud theo nhãn cảm xúc cung cấp cái nhìn sâu sắc hơn về đặc điểm ngôn ngữ của từng nhóm, hỗ trợ cho việc xây dựng mô hình phân loại cảm xúc.



Hình 3.9: Biểu đồ Top 20 từ phổ biến nhất

- Top 20 từ cung cấp thêm thông tin định lượng, tuy nhiên, do dấu ” ” bị loại bỏ khi tạo wordcloud nên dẫn đến thống kê top 20 từ có thể chưa phản ánh đúng ngữ cảnh.

3.4 | Lưu trữ dữ liệu trên HDFS

Sau khi tiền xử lý, dữ liệu được lưu trữ trên Hệ thống tệp phân tán Hadoop (HDFS) dưới định dạng **Parquet** bằng cách sử dụng Spark.

Lý do lựa chọn HDFS:

- Lưu trữ dữ liệu lớn, phân tán: HDFS được thiết kế để lưu trữ dữ liệu lớn, phân tán trên nhiều máy.
- Chịu lỗi tốt: HDFS có cơ chế sao lưu dữ liệu, đảm bảo an toàn dữ liệu khi có lỗi xảy ra.
- Tích hợp tốt với Spark: HDFS và Spark kết hợp tốt, cho phép xử lý dữ liệu hiệu quả.

Lý do lựa chọn định dạng Parquet:

- Truy vấn nhanh: Parquet lưu trữ dạng cột, giúp Spark chỉ đọc những cột cần thiết, tăng tốc độ truy vấn.
- Nén dữ liệu tốt: Parquet hỗ trợ nhiều thuật toán nén, giảm dung lượng lưu trữ.
- Tối ưu với Spark: Parquet được thiết kế tối ưu cho Spark, giúp tăng hiệu suất truy vấn.

4 | Triển khai mô hình bằng Spark MLlib

Phần này trình bày chi tiết các bước triển khai mô hình phân loại cảm xúc trên tập dữ liệu đã được tiền xử lý và lưu trữ trên HDFS. Bốn mô hình học máy được sử dụng bao gồm: Logistic Regression, Naive Bayes, Random Forest và SVM. Tất cả các mô hình đều được triển khai bằng thư viện Spark MLlib.

Chi tiết việc triển khai mô hình có thể xem tại đây: Classification.

4.1 | Biểu diễn đặc trưng

Để các mô hình học máy có thể làm việc với dữ liệu văn bản, chúng ta cần chuyển đổi văn bản thành dạng vector số học. Quá trình này được gọi là biểu diễn đặc trưng hay trích xuất đặc trưng. Trong nghiên cứu này, kỹ thuật **CountVectorizer** kết hợp với **IDF (Inverse Document Frequency)** được sử dụng để biểu diễn đặc trưng cho các tweets.

Các bước thực hiện:

1. CountVectorizer: Sử dụng ‘`CountVectorizer`‘ của Spark MLlib để đếm tần suất xuất hiện của mỗi từ (đã được lemmatized - cột ‘`lemmatized_words`‘) trong từng tweet. ‘`CountVectorizer`‘ tạo ra một từ điển (vocabulary) từ toàn bộ tập dữ liệu và biểu diễn mỗi tweet dưới dạng một vector, trong đó mỗi phần tử của vector tương ứng với số lần xuất hiện của một từ trong từ điển.

   Trong nghiên cứu này, ‘`CountVectorizer`‘ được cấu hình với các tham số ‘`maxDF`‘ và ‘`minDF`‘ để lọc ra các từ xuất hiện quá phổ biến hoặc quá hiếm trong tập dữ liệu:

   - ‘`maxDF=0.8`‘: Loại bỏ các từ xuất hiện trong hơn 80% số lượng tweets. Tham số này giúp loại bỏ các từ quá phổ biến, thường không mang nhiều ý nghĩa phân biệt.
   - ‘`minDF=5`‘: Loại bỏ các từ xuất hiện trong ít hơn 5 tweets. Tham số này giúp loại bỏ các từ quá hiếm, có thể là lỗi chính tả, từ không có nghĩa, hoặc các từ không mang tính đại diện cho tập dữ liệu.

   Kết quả của bước `CountVectorizer`‘ được lưu trong cột ‘`raw_features`‘.

2. IDF: Sử dụng ‘`IDF`‘ của Spark MLlib để tính toán trọng số IDF cho mỗi từ dựa trên cột ‘`raw_features`‘. IDF giúp giảm trọng số của các từ xuất hiện phổ biến trong toàn bộ tập dữ liệu và tăng trọng số của các từ hiếm, có khả năng phân biệt cao. Kết quả được lưu trong cột ‘`features`‘, là cột vector đặc trưng cuối cùng được sử dụng cho các mô hình phân loại.

Sau khi thu được cột vector đặc trưng ‘`features`‘ ta giữ lại cột này và cột ’`Sentiment`’ (cột label) và tiến hành loại bỏ các cột không cần thiết. Điều này giúp:

- Tối ưu bộ nhớ: Việc loại bỏ các cột không sử dụng giúp giảm dung lượng của DataFrame, giải phóng bộ nhớ cho các tác vụ khác.
- Tăng tốc độ xử lý: Thao tác trên DataFrame nhỏ hơn thường nhanh hơn, do đó có thể cải thiện hiệu suất tổng thể.
- Đơn giản hóa: Loại bỏ các cột không liên quan giúp DataFrame trở nên gọn gàng hơn, dễ hiểu hơn và tập trung vào các dữ liệu cần thiết cho mô hình.

4.2 | Chia dữ liệu

Tập dữ liệu được chia thành hai tập con: tập huấn luyện và tập kiểm tra theo tỷ lệ 9:1. Tập huấn luyện được sử dụng để huấn luyện các mô hình phân loại, trong khi tập kiểm tra được sử dụng để đánh giá hiệu suất của các mô hình trên dữ liệu chưa từng thấy.
Trong Spark, hàm ‘`randomSplit()`‘ được sử dụng để chia dữ liệu một cách ngẫu nhiên.

4.3 | Xây dựng mô hình với Spark MLlib

Để huấn luyện các mô hình phân loại, chúng ta sử dụng pipeline của Spark MLlib. ‘`Pipeline`‘ cho phép kết hợp các bước tiền xử lý dữ liệu, trích xuất đặc trưng và huấn luyện mô hình thành một quy trình thống nhất. Trong nghiên cứu này, pipeline bao gồm các bước: ‘`CountVectorizer`‘, ‘`IDF`‘, và mô hình phân loại (Logistic Regression, Naive Bayes, Random Forest, hoặc SVM). Đầu vào của pipeline là cột ‘`lemmatized_words`‘ và ‘`Sentiment`‘, và đầu ra là mô hình đã được huấn luyện.

Bốn mô hình phân loại được triển khai bằng thư viện Spark MLlib bao gồm:

- Logistic Regression
- Naive Bayes
- Random Forest
- Support Vector Machine

Các mô hình này đều yêu cầu đầu vào là một vector đặc trưng (cột ‘`features`‘ đã được tạo ở bước trước) và nhãn phân loại (cột ‘`Sentiment`‘).

4.3.1 | Logistic Regression

Logistic Regression là một mô hình phân loại tuyến tính, ước tính xác suất của một đầu vào thuộc về một lớp cụ thể. Trong Spark MLlib, ‘`LogisticRegression`‘ được sử dụng để huấn luyện mô hình Logistic Regression. Có tham số như sau:

- ‘`maxIter = 20`‘: Số lần lặp tối đa.
- ’`regParam = 0.3`’: Tham số regularization (chính quy hóa), giúp tránh overfitting bằng cách thêm một số hạng phạt vào hàm mất mát (loss function). Giá trị ‘regParam‘ càng lớn, mức độ chính quy hóa càng mạnh.
- ’`elasticNetParam = 0`’: Tham số Elastic Net, là sự kết hợp giữa hai phương pháp regularization L1 (Lasso) và L2 (Ridge). ‘`elasticNetParam`‘ kiểm soát tỷ lệ giữa L1 và L2. Giá trị 0 tương đương với L2 regularization, giá trị 1 tương đương với L1 regularization, và giá trị giữa 0 và 1 là sự kết hợp của cả hai.

4.3.2 | Naive Bayes

Naive Bayes là một mô hình phân loại dựa trên định lý Bayes, với giả định ngây thơ (naive) rằng các đặc trưng là độc lập với nhau. Trong Spark MLlib, ‘`NaiveBayes`‘ được sử dụng để huấn luyện mô hình Naive Bayes. Có tham số như sau:

- ‘`smoothing = 1.0`‘: Tham số làm mịn (smoothing parameter), tránh trường hợp xác suất bằng 0.
- ‘`modelType = "multinomial"`‘: Loại mô hình Naive Bayes.

4.3.3 | Random Forest

Random Forest là một mô hình học máy thuộc nhóm ensemble learning, xây dựng nhiều cây quyết định (decision tree) và kết hợp kết quả dự đoán của các cây này. Trong Spark MLlib, ‘`RandomForestClassifier`‘ được sử dụng để huấn luyện mô hình Random Forest. Có tham số như sau:

- ‘`numTrees = 40`‘: Số lượng cây quyết định).
- ‘`maxDepth = 5`‘: Độ sâu tối đa của cây.
- ‘`maxBins = 64`‘: Số lượng bins tối đa được sử dụng khi chia nhỏ các giá trị thuộc tính liên tục.

4.3.4 | Support Vector Machine (SVM)

Support Vector Machine (SVM) là một mô hình phân loại tìm kiếm một siêu phẳng (hyperplane) tối ưu để phân tách hai lớp dữ liệu trong không gian đặc trưng, sao cho margin (khoảng cách từ siêu phẳng tới các điểm dữ liệu gần nhất của mỗi lớp) là lớn nhất. Trong Spark MLlib, `LinearSVC` được sử dụng để huấn luyện mô hình SVM tuyến tính. Có tham số như sau:

- ‘`maxIter = 100`‘: Số lần lặp tối đa.
- ‘`regParam = 0.1`‘: Tham số regularization, kiểm soát độ lớn của margin. Giá trị `regParam` càng lớn, mô hình càng cố gắng phân loại đúng tất cả các điểm dữ liệu huấn luyện, dẫn đến margin nhỏ hơn và có thể gây ra overfitting.

4.4 | Đánh giá mô hình

Hiệu suất của các mô hình được đánh giá trên tập kiểm tra bằng các độ đo sau:

- **Accuracy (Độ chính xác):** Tỷ lệ phần trăm các dự đoán đúng trên tổng số các dự đoán.
- **F1-score:** Trung bình điều hòa của Precision và Recall, kết hợp cả hai độ đo.

Trong Spark MLlib, ‘`MulticlassClassificationEvaluator`‘ được sử dụng để tính toán các độ đo trên.

Ngoài ra, **Confusion Matrix** cũng được sử dụng để cung cấp cái nhìn chi tiết hơn về hiệu suất của từng mô hình. Ta sẽ biết biết số lượng các điểm dữ liệu được dự đoán đúng và sai cho từng lớp.

Kết quả đánh giá mô hình sẽ được trình bày và phân tích trong phần tiếp theo.

4.5 | Lưu trữ mô hình

Sau khi huấn luyện và đánh giá, các mô hình được lưu trữ trên HDFS để có thể tái sử dụng trong tương lai mà không cần phải huấn luyện lại. Việc này giúp tiết kiệm thời gian và tài nguyên tính toán, đặc biệt là đối với các mô hình lớn và phức tạp.

Mỗi mô hình (Naive Bayes, Logistic Regression, Random Forest) sau khi huấn luyện sẽ được lưu trữ vào một thư mục riêng biệt trên HDFS. Các thư mục này chứa thông tin về mô hình, bao gồm các tham số, trọng số, và các siêu dữ liệu cần thiết để tải và sử dụng lại mô hình sau này.

Việc lưu trữ mô hình được thực hiện bằng cách sử dụng các chức năng tương ứng của thư viện Spark MLlib. Sau khi lưu, các mô hình có thể được tải lại khi cần thiết để dự đoán trên dữ liệu mới.

5 | Kết quả và kết luận

Phần này trình bày kết quả đánh giá các mô hình phân loại cảm xúc Naive Bayes, Logistic Regression, Random Forest, Support Vector Machine trên tập kiểm tra, bao gồm các độ đo Accuracy, F1-score, và Confusion Matrix. Dựa trên các kết quả này, chúng ta sẽ tiến hành so sánh và thảo luận về hiệu suất của từng mô hình, cũng như phân tích các trường hợp lỗi.

5.1 | Đánh giá mô hình

Bảng 5.1 dưới đây tổng hợp kết quả đánh giá các mô hình trên tập kiểm tra, sử dụng hai độ đo chính là Accuracy và F1-score:

Mô hình	Accuracy	F1-score
Naive Bayes	0.8015	0.8015
Logistic Regression	0.8187	0.8187
Random Forest	0.6535	0.6461
Support Vector Machine	0.7919	0.7915

Bảng 5.1: Kết quả đánh giá các mô hình

Nhìn chung, **Logistic Regression** cho thấy hiệu suất vượt trội với Accuracy và F1-score đều đạt 0.8187, chứng tỏ khả năng dự đoán chính xác cao trên tập dữ liệu kiểm tra. **Naive Bayes** cũng thể hiện ấn tượng với Accuracy và F1-score đều đạt 0.8015, bám sát Logistic Regression. Dù dựa trên giả định ”ngây thơ” về tính độc lập giữa các đặc trưng, Naive Bayes vẫn có khả năng phân loại cảm xúc tốt trên tập dữ liệu này.

Mô hình **SVM** đạt kết quả khả quan với Accuracy 0.7919 và F1-score 0.7915, đứng thứ ba trong số bốn mô hình. Kết quả này cho thấy SVM cũng là một lựa chọn phù hợp cho bài toán phân tích cảm xúc trên tweets, mặc dù hiệu suất chưa bằng được Logistic Regression và Naive Bayes.

Trái ngược với ba mô hình trên, **Random Forest** cho thấy hiệu suất thấp nhất, với Accuracy chỉ đạt 0.6535 và F1-score là 0.6461. Kết quả này cho thấy mô hình Random Forest chưa thực sự phù hợp với tập dữ liệu hoặc tham số của mô hình chưa được tối ưu.

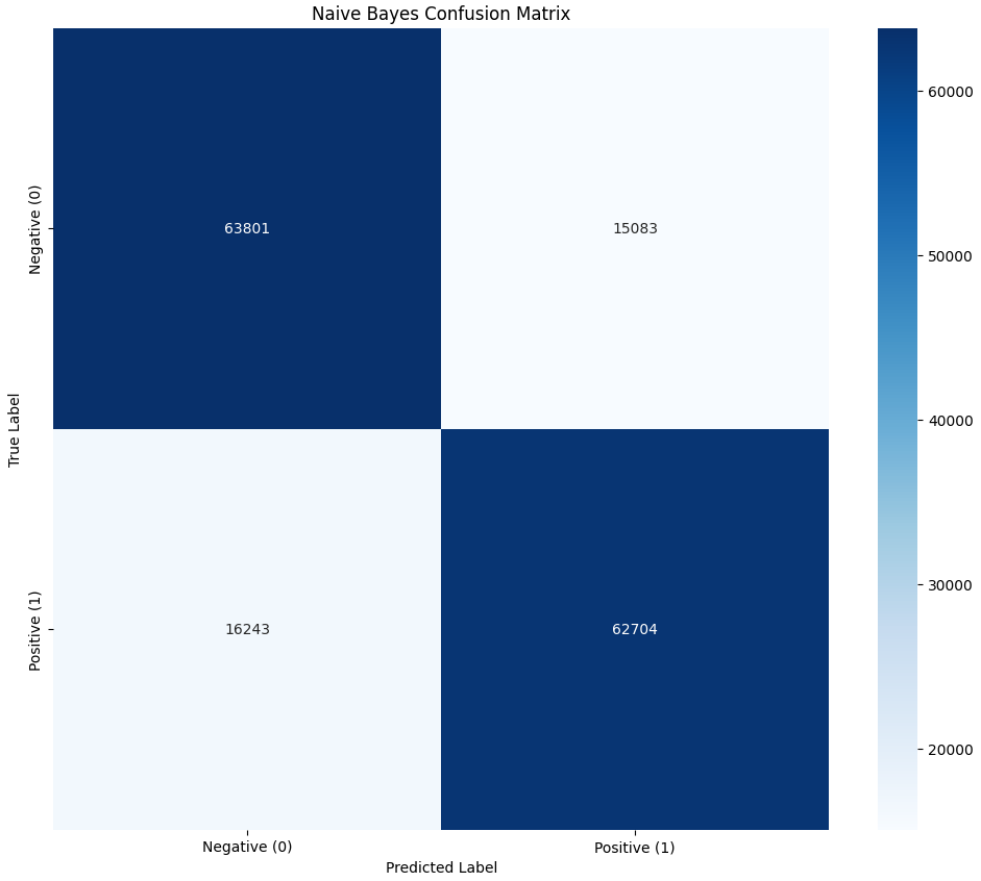
Bảng 5.2: Kết quả đánh giá các mô hình

Mô hình	Điểm Mạnh	Điểm Yếu
Logistic Regression	Đơn giản, dễ hiểu, hiệu quả, nhanh chóng	Giả định mối quan hệ tuyến tính, khó biểu diễn mối quan hệ phức tạp
Naive Bayes	Nhanh, hiệu quả, cần ít dữ liệu huấn luyện	Giả định các đặc trưng độc lập, có thể kém chính xác hơn
Random Forest	Độ chính xác cao, ít bị overfitting	Khó diễn giải, tốn nhiều tài nguyên tính toán, tham số chưa tối ưu
Support Vector Machine	Hiệu quả với dữ liệu chiều cao, linh hoạt với nhiều kernel	Cần chọn kernel và tham số cẩn thận, không có ước tính xác suất trực tiếp

5.2 | Phân tích Confusion Matrix

Để hiểu rõ hơn về hiệu suất của từng mô hình, chúng ta sẽ phân tích Confusion Matrix cho mỗi mô hình. Confusion Matrix cho biết số lượng các điểm dữ liệu được dự đoán đúng và sai cho từng lớp (Positive và Negative).

5.2.1 | Confusion Matrix cho mô hình Naive Bayes



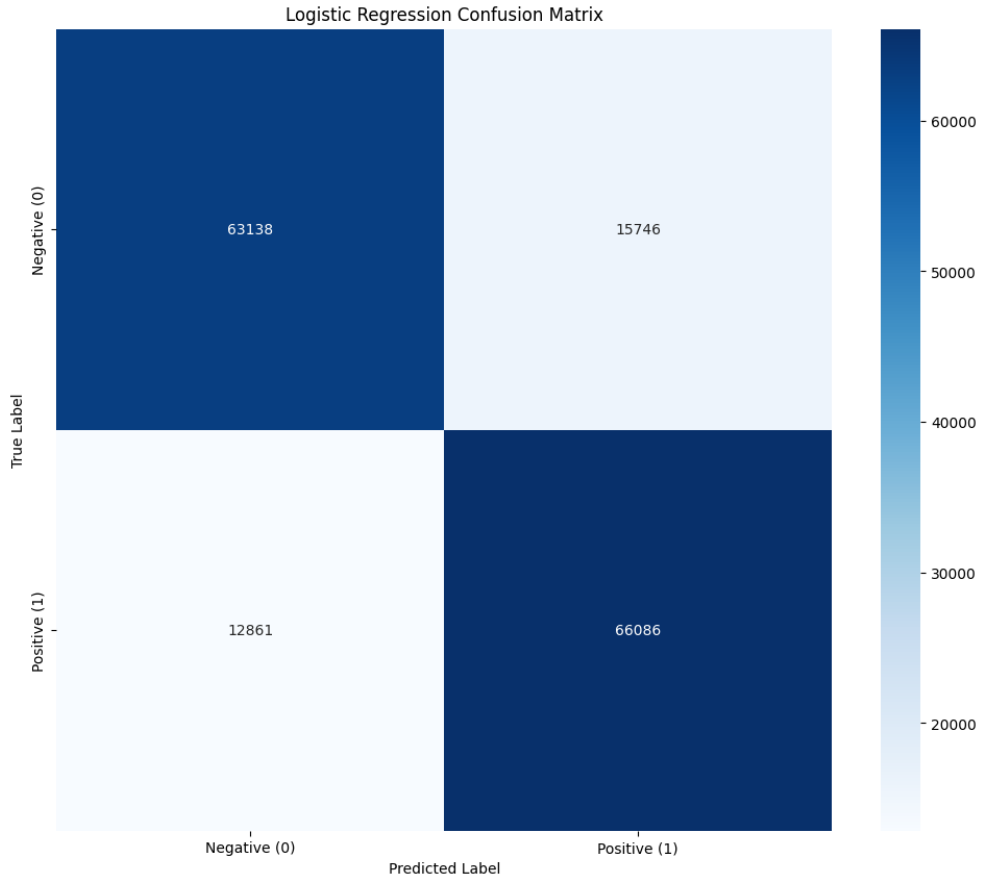
Hình 5.1: Confusion Matrix cho mô hình Naive Bayes

Hình 5.1 hiển thị Confusion Matrix của mô hình Naive Bayes. Các giá trị trên đường chéo chính (từ trái sang phải) thể hiện số lượng các dự đoán đúng (True Positives và True Negatives), trong khi các giá trị ngoài đường chéo chính thể hiện số lượng các dự đoán sai (False Positives và False Negatives).

Nhận xét:

- Mô hình dự đoán đúng 63,801 tweets Negative (TN) và 62.704 tweets Positive (TP).
- Mô hình dự đoán sai 16.243 tweets Positive thành Negative (FN) và 15.083 tweets Negative thành Positive (FP).
- Tỷ lệ True Positive Rate (TPR) hay Sensitivity của model đạt = 79.43%.
- Tỷ lệ True Negative Rate (TNR) hay Specificity của model đạt = 80.88%.

5.2.2 | Confusion Matrix cho mô hình Logistic Regression



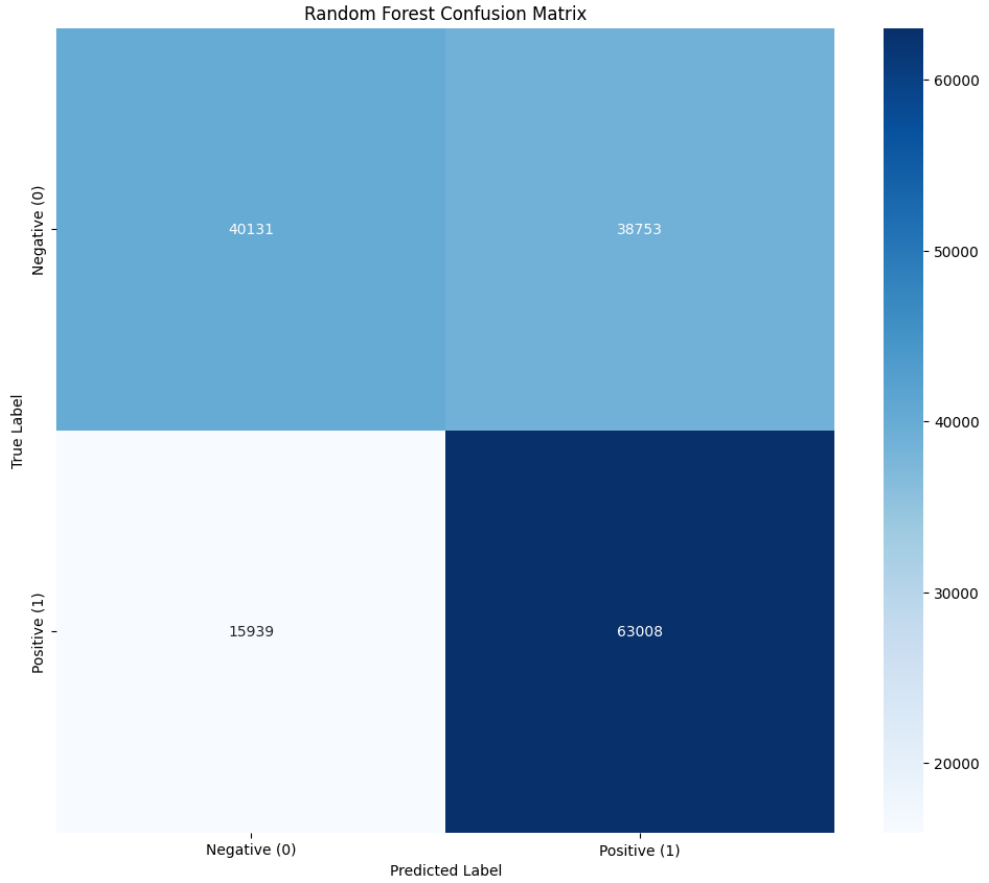
Hình 5.2: Confusion Matrix cho mô hình Logistic Regression

Hình 5.2 hiển thị Confusion Matrix của mô hình Logistic Regression.

Nhận xét:

- Mô hình dự đoán đúng 63,138 tweets Negative (TN) và 66,086 tweets Positive (TP).
- Mô hình dự đoán sai 12,861 tweets Positive thành Negative (FN) và 15,746 tweets Negative thành Positive (FP).
- Tỷ lệ True Positive Rate (TPR) hay Sensitivity của model đạt = 83.71%
- Tỷ lệ True Negative Rate (TNR) hay Specificity của model đạt = 80.04%.
- So với Naive Bayes, Logistic Regression dự đoán đúng ít trường hợp Negative hơn (TN thấp hơn) và đúng nhiều trường hợp Positive hơn (TP cao hơn). Tuy nhiên, tổng thể số lượng dự đoán đúng của Logistic Regression cao hơn Naive Bayes, dẫn đến Accuracy và F1-score cao hơn.

5.2.3 | Confusion Matrix cho mô hình Random Forest



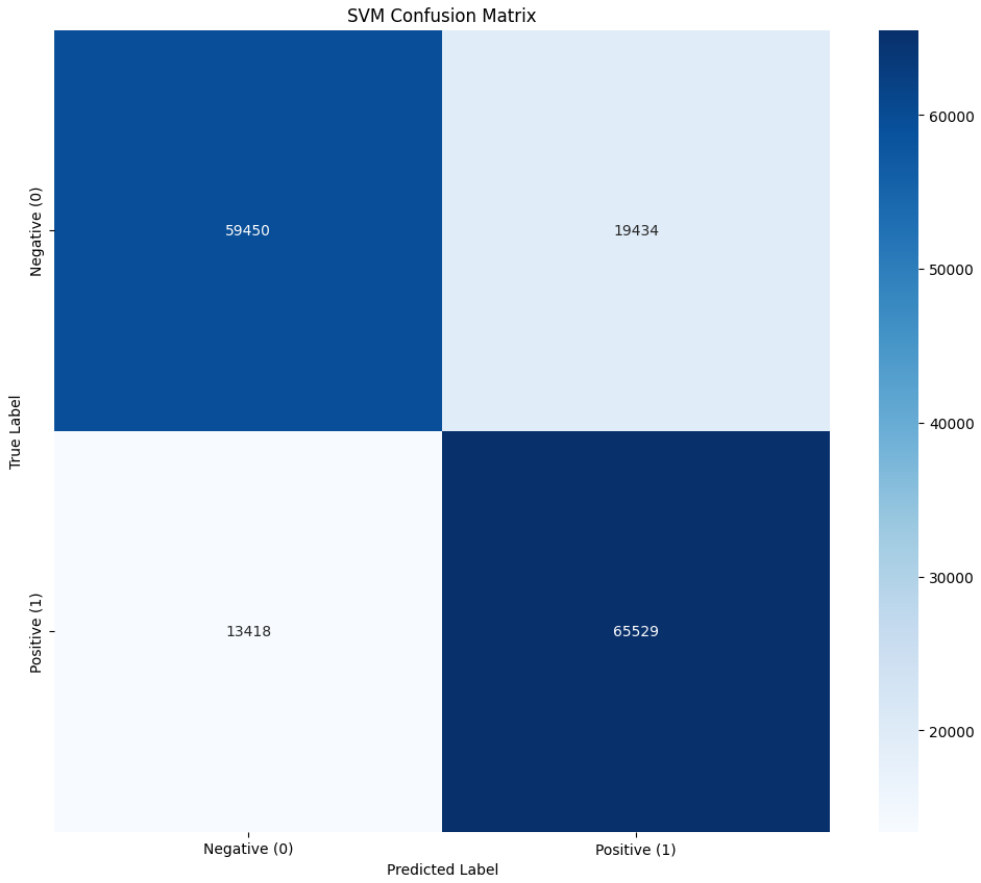
Hình 5.3: Confusion Matrix cho mô hình Random Forest

Hình 5.3 hiển thị Confusion Matrix của mô hình Random Forest.

Nhận xét:

- Mô hình dự đoán đúng 40,131 tweets Negative (TN) và 63,008 tweets Positive (TP).
- Mô hình dự đoán sai 15,939 tweets Positive thành Negative (FN) và 38,753 tweets Negative thành Positive (FP).
- Tỷ lệ True Positive Rate (TPR) hay Sensitivity của model đạt = 79.81%.
- Tỷ lệ True Negative Rate (TNR) hay Specificity của model đạt = 50.87%.
- Mô hình Random Forest có độ chính xác dự đoán các tweets Positive khá cao (TP cao), nhưng lại dự đoán các tweets Negative rất kém (TN thấp), dẫn đến hiệu suất tổng thể (Accuracy và F1-score) thấp hơn so với hai mô hình còn lại.

5.2.4 | Confusion Matrix cho mô hình Support Vector Machine



Hình 5.4: Confusion Matrix cho mô hình SVM

Hình 5.4 hiển thị Confusion Matrix của mô hình Support Vector Machine.

Nhận xét:

- Mô hình dự đoán đúng 59,450 tweets Negative (TN) và 65,529 tweets Positive (TP).
- Mô hình dự đoán sai 13,418 tweets Positive thành Negative (FN) và 19,434 tweets Negative thành Positive (FP).
- Tỷ lệ True Positive Rate (TPR) hay Sensitivity của model đạt = 83.00%.
- Tỷ lệ True Negative Rate (TNR) hay Specificity của model đạt = 75.36%.
- Mô hình SVM có độ chính xác dự đoán các tweets Positive khá cao (TP cao) và chỉ kém Logistic Regression một chút mà thôi, dự đoán các tweets Negative (TN) không quá cao, thấp hơn so với Naive Bayes và Logistic Regression nhưng cao hơn Random Forest, nhìn chung thì hiệu suất tổng thể (Accuracy và F1-score) chỉ đứng sau Naive Bayes một chút thôi

5.3 | So sánh hiệu suất các mô hình

Dựa trên các kết quả đánh giá ở bảng 5.1, 5.2 và phân tích Confusion Matrix ở các hình 5.1, 5.2, 5.3, và 5.4, ta có thể đưa ra một số nhận xét và so sánh về hiệu suất của bốn mô hình như sau:

- **Logistic Regression**: Cho thấy hiệu suất tốt nhất trong cả bốn mô hình với Accuracy và F1-score đạt 0.8187. Mô hình này đạt được sự cân bằng tốt giữa việc dự đoán đúng các trường hợp Positive và Negative, thể hiện qua Confusion Matrix với TPR đạt 83.71% và TNR đạt 80.04%.
- **Naive Bayes**: Có hiệu suất tốt thứ hai, với Accuracy và F1-score đều đạt 0.8015. Confusion Matrix cho thấy mô hình dự đoán khá tốt cả hai lớp Positive và Negative, với TPR đạt 79.43% và TNR đạt 80.88%. Tuy nhiên, mô hình có xu hướng dự đoán nhầm các tweets Negative thành Positive (FP = 15,083) hơn so với Logistic Regression.
- **SVM**: Xếp thứ ba với Accuracy 0.7919 và F1-score 0.7915. Mô hình cho thấy khả năng dự đoán các trường hợp Positive khá tốt (TPR đạt 83.00%, chỉ kém Logistic Regression một chút), nhưng dự đoán các trường hợp Negative kém chính xác hơn so với Naive Bayes và Logistic Regression (TNR đạt 75.36%).
- **Random Forest**: Có hiệu suất thấp nhất trong bốn mô hình với Accuracy chỉ đạt 0.6535 và F1-score là 0.6461. Confusion Matrix cho thấy mô hình dự đoán đúng 40,131 tweets Negative (TN) và 63,008 tweets Positive (TP), dự đoán sai 15,939 tweets Positive thành Negative (FN) và 38,753 tweets Negative thành Positive (FP). Tỷ lệ TPR hay Sensitivity của model đạt 79.81% và TNR hay Specificity đạt 50.87%. Như vậy, mô hình Random Forest có độ chính xác dự đoán các tweets Positive ở mức trung bình (TPR đạt gần 80%) nhưng lại dự đoán các tweets Negative rất kém (TNR chỉ hơn 50%), dẫn đến hiệu suất tổng thể (Accuracy và F1-score) thấp hơn so với ba mô hình còn lại.

Nhìn chung, Logistic Regression và Naive Bayes cho thấy sự vượt trội hơn hẳn so với Random Forest trong bài toán phân tích cảm xúc trên tập dữ liệu này. SVM cũng là một lựa chọn tốt, với hiệu suất gần bằng Naive Bayes.

5.4 | Phân tích lỗi

Phân tích Confusion Matrix ở các hình 5.1, 5.2, 5.3, và 5.4 cho thấy các mô hình, đặc biệt là Random Forest, gặp khó khăn trong việc dự đoán chính xác các tweets Negative (thể hiện qua tỷ lệ True Negative Rate - TNR thấp). Một số nguyên nhân có thể dẫn đến tình trạng này bao gồm:

- **Đặc trưng chưa đủ tốt:** Các đặc trưng được trích xuất bằng CountVectorizer và IDF có thể chưa đủ để nắm bắt đầy đủ sắc thái cảm xúc trong các tweets. Một số từ ngữ biểu thị cảm xúc Negative có thể không xuất hiện trong từ điển của CountVectorizer hoặc có trọng số IDF thấp.
- **Mẫu Negative khó phân loại hơn:** Có thể tồn tại sự mất cân bằng trong cách sử dụng từ ngữ hoặc mức độ phức tạp trong việc thể hiện cảm xúc giữa hai lớp. Các tweets Negative có thể sử dụng ngôn ngữ phức tạp hơn, nhiều từ đồng nghĩa/trái nghĩa, hoặc chứa các yếu tố mỉa mai, châm biếm, khiến cho việc phân loại trở nên khó khăn hơn.

- **Tham số mô hình chưa tối ưu:** Các tham số của các mô hình, đặc biệt là Random Forest, có thể chưa được tối ưu cho tập dữ liệu này. Việc tinh chỉnh tham số (parameter tuning) có thể giúp cải thiện hiệu suất của các mô hình.
- **Độ phức tạp của ngôn ngữ tự nhiên:** Phân tích cảm xúc trên mạng xã hội là một nhiệm vụ khó, do sự đa dạng trong cách diễn đạt, sử dụng từ lóng, viết tắt, sai chính tả, và các yếu tố cảm xúc tinh tế, khó nắm bắt.
- **Thiếu ngữ cảnh**: Các mô hình hiện tại chỉ dựa vào nội dung của từng tweet riêng lẻ mà không xét đến ngữ cảnh rộng hơn (ví dụ: các tweets trước đó trong cùng một cuộc hội thoại, thông tin về người dùng, v.v.). Việc bổ sung ngữ cảnh có thể giúp cải thiện độ chính xác của mô hình.

Để cải thiện hiệu suất của các mô hình, cần tiến hành phân tích lỗi chi tiết hơn, xem xét các trường hợp dự đoán sai để tìm ra nguyên nhân cụ thể và đưa ra các giải pháp phù hợp.

6 | Phân tích cảm xúc sử dụng Spark NLP và Deep Learning

Phần này trình bày chi tiết về việc triển khai mô hình phân tích cảm xúc sử dụng thư viện Spark NLP, với trọng tâm là áp dụng mô hình Deep Learning mạnh mẽ '**ClassifierDLApproach**' kết hợp với '**BertEmbeddings**' để biểu diễn từ.

6.1 | Triển khai mô hình

Để tận dụng sức mạnh của Deep Learning trong bài toán phân tích cảm xúc, chúng ta sử dụng 'ClassifierDLApproach' của Spark NLP, kết hợp với 'BertEmbeddings' để biểu diễn từ.

- 'ClassifierDLApproach': cho phép huấn luyện các mô hình phân loại văn bản dựa trên các kiến trúc Deep Learning. Mô hình này nhận đầu vào là các vector biểu diễn câu (sentence embeddings) và đầu ra là nhãn dự đoán cho từng câu.
- 'BertEmbeddings': cung cấp các vector biểu diễn từ (word embeddings) được huấn luyện trước dựa trên mô hình BERT (Bidirectional Encoder Representations from Transformers). BERT là một mô hình ngôn ngữ mạnh mẽ, có khả năng nắm bắt ngữ nghĩa và ngữ cảnh của từ một cách hiệu quả. Việc sử dụng 'BertEmbeddings' giúp cải thiện đáng kể độ chính xác của mô hình phân loại cảm xúc.

6.1.1 | Chuẩn bị dữ liệu

Do hạn chế về tài nguyên tính toán, chúng ta tiến hành lấy mẫu từ tập dữ liệu gốc để thu được tập dữ liệu cân bằng hơn, bao gồm 200,000 tweets với tỷ lệ số lượng mẫu positive/negative là 1/1 (mỗi loại 100,000 tweets). Các bước thực hiện như sau:

1. Lấy mẫu dữ liệu: Từ tập dữ liệu đã qua tiền xử lý, lấy mẫu 100,000 tweets tích cực ('`Sentiment`' = 1) và 100,000 tweets tiêu cực ('`Sentiment`' = 0).
2. Kết hợp dữ liệu: Kết hợp hai tập dữ liệu positive và negative thành một tập dữ liệu cân bằng duy nhất.
3. Chia dữ liệu: Chia tập dữ liệu cân bằng thành hai tập: tập huấn luyện (80%) và tập kiểm tra (20%) bằng cách sử dụng phương thức '`randomSplit()`'.

6.1.2 | Xây dựng Pipeline

Mô hình 'ClassifierDLApproach' được triển khai trong pipeline của Spark MLlib để đảm bảo tính nhất quán trong các bước tiền xử lý và biểu diễn đặc trưng. Pipeline bao gồm các bước chính sau:

1. 'DocumentAssembler': Chuyển đổi cột '`SentimentText`' (chứa nội dung tweet) thành định dạng '`document`' - định dạng chuẩn cho các annotator đầu vào của Spark NLP.
2. 'Tokenizer': Tách các câu trong '`document`' thành các token (từ).
3. 'BertEmbeddings': Sử dụng mô hình '`small_bert_L4_256`' đã được huấn luyện trước ('pretrained') để tạo vector biểu diễn cho từng từ (token) trong câu. Mô hình này là một phiên bản thu nhỏ của mô hình BERT, phù hợp với các trường hợp tài nguyên hạn chế.

4. ‘SentenceEmbeddings‘: Kết hợp các vector biểu diễn từ (word embeddings) thành một vector biểu diễn cho toàn bộ câu (sentence embedding) bằng cách sử dụng phương pháp trung bình cộng (‘AVERAGE‘).
5. ‘ClassifierDLApproach‘: Huấn luyện mô hình phân loại cảm xúc dựa trên các vector biểu diễn câu. Các tham số quan trọng bao gồm:
   - ‘maxEpochs‘: Số lượng epochs tối đa.
   - ‘lr‘: learning rate.
   - ‘batchSize‘: Kích thước batch.
   - ‘validationSplit‘: Tỷ lệ dữ liệu validation.
   - ‘dropout‘: Tỷ lệ dropout, giúp tránh overfitting.
   - ‘setEnableOutputLogs(True)‘: Bật in ra thông tin training log.
   - ‘setOutputLogsPath('logs')‘: Thiết lập đường dẫn lưu log, thuận tiện cho việc theo dõi quá trình train.

Pipeline sẽ thực hiện tuần tự các bước trên, từ tiền xử lý dữ liệu, trích xuất đặc trưng, đến huấn luyện hoặc sử dụng mô hình phân loại cảm xúc.

6.1.3 | Huấn luyện và đánh giá mô hình

Mô hình ‘ClassifierDLApproach‘ được huấn luyện trên tập ‘train_data‘ thông qua ‘pipeline‘. Sau khi huấn luyện, mô hình được sử dụng để dự đoán trên tập train_data‘. Để đánh giá hiệu suất, ta sử dụng các độ đo ‘Accuracy‘ và ‘F1-score‘, thông qua ‘MulticlassClassificationEvaluator‘ của Spark MLlib. Ngoài ra, ‘classification_report‘ của thư viện ‘scikit-learn‘ cũng được sử dụng để cung cấp báo cáo phân loại chi tiết hơn.

6.1.4 | Theo dõi quá trình huấn luyện

Để theo dõi quá trình huấn luyện, ta bật ‘setEnableOutputLogs(True)‘ và đặt ‘setOutputLogsPath(’logs’)‘ để lưu log vào thư mục ‘logs‘. Việc theo dõi log giúp ta nắm bắt được các thông số như ‘loss‘, ‘accuracy‘ sau mỗi epoch, từ đó có thể điều chỉnh các tham số của mô hình cho phù hợp.

6.2 | Kết quả và So sánh

Mô hình	Accuracy	F1-score
Naive Bayes	0.8015	0.8015
Logistic Regression	0.8187	0.8187
Random Forest	0.6535	0.6461
Support Vector Machine	0.7919	0.7915
ClassifierDL (Bert Embeddings)	0.78	0.78

Bảng 6.1: Kết quả đánh giá các mô hình

Bảng 6.1 tổng hợp kết quả đánh giá các mô hình, bao gồm cả 4 mô hình đã triển khai ở phần trước (Naive Bayes, Logistic Regression, Random Forest, SVM) và mô hình ‘ClassifierDLApproach‘ với ‘BertEmbeddings‘ mới được huấn luyện.

Mô hình 'ClassifierDLApproach' đạt được Accuracy 0.78 và F1-score 0.78 trên tập kiểm tra. Kết quả này cho thấy mô hình Deep Learning với 'BertEmbeddings' có hiệu suất tương đương với mô hình SVM và kém hơn một chút so với Naive Bayes và Logistic Regression, nhưng vượt trội hơn so với Random Forest.
Dưới đây là báo cáo phân loại chi tiết từ 'classification_report' cho mô hình 'ClassifierDLApproach':

	Precision	Recall	F1-score	Support
0	0.73	0.80	0.76	18171
1	0.82	0.75	0.79	22046
Accuracy			0.78	40217
Macro avg	0.78	0.78	0.78	40217
Weighted avg	0.78	0.78	0.78	40217

Bảng 6.2: Classification Report

6.2.1 | Nhận xét

- Mô hình đạt precision 0.73 cho lớp 0 (Negative) và 0.82 cho lớp 1 (Positive), recall 0.80 cho lớp 0 và 0.75 cho lớp 1.
- F1-score cho lớp 0 là 0.76 và cho lớp 1 là 0.79.
- Kết quả cho thấy mô hình dự đoán lớp 1 (Positive) tốt hơn một chút so với lớp 0 (Negative).

6.2.2 | So sánh với các mô hình khác

- So với các mô hình truyền thống (Logistic Regression, Naive Bayes, SVM, Random Forest), mô hình `ClassifierDLApproach` với `BertEmbeddings` cho kết quả ở mức **khá tốt**, gần bằng SVM.
- Mặc dù chưa đạt được hiệu suất cao nhất, nhưng việc sử dụng `BertEmbeddings` đã cho thấy tiềm năng trong việc cải thiện độ chính xác của mô hình phân tích cảm xúc. `BertEmbeddings` có khả năng nắm bắt ngữ nghĩa và ngữ cảnh tốt hơn so với `CountVectorizer` và `IDF`, dẫn đến các vector biểu diễn câu có chất lượng cao hơn.

6.2.3 | Phân tích nguyên nhân mô hình Deep Learning chưa đạt hiệu suất cao nhất

Mặc dù 'BertEmbeddings' và Deep Learning thường được kỳ vọng mang lại hiệu suất vượt trội, trong trường hợp này, mô hình 'ClassifierDLApproach' chưa đạt được kết quả cao nhất. Một số nguyên nhân có thể lý giải cho điều này bao gồm:

- Kích thước tập dữ liệu huấn luyện: Mô hình Deep Learning thường yêu cầu lượng dữ liệu huấn luyện lớn để đạt hiệu suất tối ưu. Trong nghiên cứu này, do hạn chế về tài nguyên, chúng ta chỉ sử dụng tập dữ liệu mẫu với 200,000 tweets. Tập dữ liệu này có thể chưa đủ lớn để mô hình Deep Learning phát huy hết tiềm năng.

- Tham số mô hình chưa tối ưu: Việc tinh chỉnh tham số (hyperparameter tuning) cho các mô hình Deep Learning rất quan trọng và thường tốn nhiều thời gian. Trong phạm vi nghiên cứu này, các tham số của ‘ClassifierDLApproach‘ có thể chưa được tối ưu hoàn toàn cho tập dữ liệu.
- Kiến trúc mô hình: Mô hình ‘ClassifierDLApproach‘ với ‘BertEmbeddings‘ sử dụng kiến trúc mạng đơn giản (chỉ gồm các lớp ‘WordEmbeddings‘, ‘SentenceEmbeddings‘, ‘ClassifierDL‘). Việc thử nghiệm với các kiến trúc mạng phức tạp hơn (như CNN, RNN) hoặc sử dụng các pre-trained models BERT lớn hơn có thể mang lại hiệu suất cao hơn.
- Đặc thù của dữ liệu: Có thể đặc thù của dữ liệu Twitter (nhiều từ viết tắt, tiếng lóng, ký tự đặc biệt, v.v.) làm cho việc áp dụng các mô hình Deep Learning trở nên khó khăn hơn. Cần có các bước tiền xử lý và kỹ thuật đặc biệt để xử lý hiệu quả loại dữ liệu này.
- Sự đơn giản của bài toán: Trong một số trường hợp, các mô hình truyền thống như Logistic Regression và Naive Bayes vẫn có thể hoạt động tốt với các bài toán đơn giản hoặc với các đặc trưng được trích xuất tốt. Có thể trong bài toán này, sự kết hợp CountVectorizer + IDF với Logistic Regression/Naive Bayes đã đủ hiệu quả, và sự phức tạp của mô hình Deep Learning chưa mang lại lợi thế đáng kể.

6.2.4 | Kết luận

- Mô hình Deep Learning `ClassifierDLApproach` với `BertEmbeddings` cho kết quả khả quan trong bài toán phân tích cảm xúc trên tập dữ liệu tweets, đạt hiệu suất gần bằng với SVM.
- Việc sử dụng pre-trained `BertEmbeddings` giúp cải thiện khả năng biểu diễn đặc trưng cho văn bản, dẫn đến kết quả phân loại tốt hơn.
- Cần thử nghiệm thêm với việc tinh chỉnh tham số, thay đổi kiến trúc mạng, và sử dụng các pre-trained embeddings khác để có thể cải thiện hơn nữa hiệu suất của mô hình.
- Việc in ra các thông số `loss` và `accuracy` trong quá trình training giúp ích cho việc điều chỉnh các tham số.

7 | Phân tích cảm xúc trên Hadoop MapReduce

Phần này trình bày chi tiết về việc triển khai mô hình Naive Bayes để phân tích cảm xúc trên Hadoop MapReduce, tập trung vào các bước cài đặt, cấu hình, và kết quả thực nghiệm.

7.1 | Triển khai và Cài đặt

Mô hình Naive Bayes được triển khai trên Hadoop MapReduce, bao gồm hai giai đoạn chính: **Training** (huấn luyện) và **Testing** (kiểm thử).

7.1.1 | Giai đoạn Training

Giai đoạn này sử dụng một job MapReduce để tính toán tần suất xuất hiện của các từ trong từng loại cảm xúc (Positive và Negative).

- Mapper (Map_Training):
  - Đầu vào: Các dòng trong file dữ liệu, mỗi dòng chứa thông tin về một tweet (ID, nhãn cảm xúc, nội dung tweet).
  - Tiền xử lý:
    - Loại bỏ các URL.
    - Loại bỏ các mentions, hashtags, và các ký tự đặc biệt.
    - Loại bỏ các số.
    - Loại bỏ dấu câu.
    - Chuyển đổi tất cả các ký tự thành chữ thường.
    - Loại bỏ khoảng trắng thừa.
  - Xử lý:
    - Xác định nhãn cảm xúc của tweet (Positive (1) hoặc Negative (0)).
    - Tách các từ trong nội dung tweet.
    - Phát ra cặp key-value <`word`, `sentiment`> với mỗi từ.
    - Theo dõi số lượng tweets, số lượng tweets Positive/Negative, số lượng từ trong các tweets Positive/Negative qua các Counters: `TWEETS_SIZE`, `POS_TWEETS_SIZE`, `NEG_TWEETS_SIZE`, `POS_WORDS_SIZE`, `NEG_WORDS_SIZE`.
  - Đầu ra: Các cặp key-value <`word`, `sentiment`>.
- Reducer (Reduce_Training):
  - Đầu vào**:** Các cặp key-value <`word`, `sentiment`> được nhóm theo `word`.
  - Xử lý:
    - Với mỗi từ, đếm số lần xuất hiện trong các tweets Positive (`pos_count`) và Negative (`neg_count`).
    - Theo dõi tổng số từ duy nhất qua Counter `FEATURES_SIZE`.
  - Đầu ra: Các cặp key-value <`word`, `pos_count@neg_count`> được lưu trong thư mục `training` trên HDFS.

7.1.2 | Giai đoạn Testing

Giai đoạn này sử dụng một job MapReduce để dự đoán cảm xúc của các tweets trong tập kiểm tra.

- Mapper (Map_Testing):
  - Thiết lập (Setup):
    - Đọc các giá trị counters từ job training.
    - Tính xác suất cho mỗi loại (Positive/Negative).
    - Đọc dữ liệu từ thư mục `training`, lưu trữ số lần xuất hiện của mỗi từ trong từng loại vào hai HashMap: `pos_words` và `neg_words`.
    - Tính xác suất của từng từ trong mỗi loại (dùng Laplace smoothing) và lưu vào `pos_words_probabilities` và `neg_words_probabilities`.
  - Đầu vào: Các dòng trong file dữ liệu kiểm tra, mỗi dòng chứa thông tin về một tweet.
  - Tiền xử lý: Tương tự như trong giai đoạn Training.
  - Xử lý:
    - Với mỗi tweet, tính xác suất để tweet đó thuộc về mỗi loại (Positive/Negative) dựa trên các từ trong tweet và xác suất của từng từ trong mỗi loại.
    - Gán nhãn cho tweet dựa trên xác suất lớn hơn.
    - Theo dõi các giá trị TP, TN, FP, FN qua Counters: `TRUE_POSITIVE`, `TRUE_NEGATIVE`, `FALSE_POSITIVE`, `FALSE_NEGATIVE`.
  - Đầu ra: Các cặp key-value <`tweet_id@tweet_text`, `predicted_sentiment`>.
- Reducer: Không sử dụng Reducer. Kết quả của Mapper được ghi trực tiếp ra HDFS.

7.2 | Kết quả thực nghiệm

Kết quả từ hai lần chạy thử nghiệm với 100,000 tweets và 200,000 tweets (75% train và 25% test) như sau:

1. **Lần chạy 1:**
   - Confusion Matrix:

$$\begin{matrix} 11793 & 3005 \\ 2973 & 7229 \end{matrix}$$

   - Accuracy: 0.76088
   - Thời gian thực thi: 258.7057 giây
2. **Lần chạy 2:**
   - Confusion Matrix:

$$\begin{matrix} 23780 & 6102 \\ 5765 & 14353 \end{matrix}$$

   - Accuracy: 0.76266

- Thời gian thực thi: 768.74725 giây

Nhận xét:

- Mô hình Naive Bayes triển khai trên Hadoop MapReduce đạt được độ chính xác 0.76088 trong lần thử nghiệm đầu tiên, cho thấy tiềm năng trong việc xử lý dữ liệu lớn.
- Hadoop MapReduce cho phép xử lý tập dữ liệu lớn hiệu quả, tận dụng khả năng tính toán song song và phân tán.
- Thời gian thực thi cho thấy MapReduce chạy khá tốn thời gian, đặc biệt là với tập dữ liệu lớn (Với tập dữ liệu gồm 1,000,000 tweets có thể tốn thời gian gấp khoảng 80 lần tập dữ liệu 100,000 tweets)

8 | Kết luận và Hướng phát triển

8.1 | Kết luận

Dự án này đã trình bày quá trình xây dựng và đánh giá các mô hình học máy để phân tích cảm xúc trên tập dữ liệu gồm 1.6 triệu tweets từ nguồn Sentiment140. Hai nền tảng chính được sử dụng là Spark (với thư viện Spark MLlib và Spark NLP) và Hadoop MapReduce. Các mô hình được triển khai bao gồm: Logistic Regression, Naive Bayes, Random Forest, Support Vector Machine (SVM) trên Spark MLlib, mô hình ‘ClassifierDLApproach‘ với ‘BertEmbeddings‘ trên Spark NLP, và Naive Bayes trên Hadoop MapReduce.

Kết quả thực nghiệm cho thấy mô hình Logistic Regression trên Spark MLlib đạt hiệu suất cao nhất với Accuracy và F1-score đều đạt 0.8187. Mô hình Naive Bayes trên cả Spark MLlib và Hadoop MapReduce đều cho kết quả tốt, với Accuracy và F1-score xấp xỉ 0.80. Mô hình SVM cũng cho thấy hiệu suất khả quan, đạt Accuracy 0.7919 và F1-score 0.7915. Mô hình ‘ClassifierDLApproach‘ với ‘BertEmbeddings‘ đạt kết quả hứa hẹn, với Accuracy và F1-score cùng đạt 0.78, cho thấy tiềm năng của Deep Learning trong phân tích cảm xúc. Mô hình Random Forest đạt kết quả thấp nhất, cho thấy sự chưa phù hợp của mô hình với tập dữ liệu hoặc cần tinh chỉnh tham số tốt hơn.

Phân tích Confusion Matrix đã cung cấp cái nhìn chi tiết hơn về hiệu suất của từng mô hình, đặc biệt là trong việc dự đoán các trường hợp Positive và Negative.

Việc lưu trữ dữ liệu đã qua tiền xử lý trên HDFS dưới định dạng Parquet, cũng như lưu trữ các mô hình đã huấn luyện (đối với Spark), đã cho thấy tính hiệu quả và khả năng ứng dụng của dự án trong việc xử lý dữ liệu lớn và triển khai mô hình học máy.

Việc triển khai Naive Bayes trên Hadoop MapReduce tuy có phần thủ công và tốn thời gian hơn, nhưng cũng đã cho thấy khả năng xử lý dữ liệu lớn và phân tán của nền tảng này, đạt độ chính xác 0.76088 ở lần thử nghiệm đầu tiên.

Nhìn chung, dự án đã đạt được mục tiêu đề ra là xây dựng được các mô hình phân tích cảm xúc trên dữ liệu tweets sử dụng Spark MLlib, Spark NLP và Hadoop MapReduce, đồng thời đánh giá và so sánh hiệu suất của các mô hình này. Kết quả dự án cung cấp cái nhìn sâu sắc về đặc điểm của tập dữ liệu Sentiment140, cũng như tiềm năng và hạn chế của các mô hình học máy khác nhau trong việc phân tích cảm xúc trên dữ liệu văn bản từ mạng xã hội.

8.2 | Hạn chế

Mặc dù đạt được những kết quả nhất định, dự án vẫn còn tồn tại một số hạn chế:

- **Tập dữ liệu:** Dự án chỉ tập trung vào tập dữ liệu Sentiment140, do đó, kết quả có thể không khái quát hóa được cho các tập dữ liệu khác hoặc các nguồn dữ liệu khác (ví dụ: Facebook, diễn đàn, v.v.). Hơn nữa, việc gán nhãn cảm xúc trong Sentiment140 dựa trên biểu tượng cảm xúc có thể chưa phản ánh đầy đủ sắc thái cảm xúc trong ngôn ngữ.
- **Biểu diễn đặc trưng:** Việc sử dụng CountVectorizer và IDF để biểu diễn đặc trưng trong các mô hình truyền thống có thể chưa nắm bắt được đầy đủ ngữ nghĩa và ngữ cảnh của các

từ trong câu, cũng như mối quan hệ giữa các từ. Mặc dù 'BertEmbeddings' đã cải thiện được phần nào, nhưng vẫn cần nghiên cứu thêm để tối ưu hóa việc biểu diễn đặc trưng cho dữ liệu ngôn ngữ tự nhiên, đặc biệt là trên mạng xã hội.

- **Mất cân bằng dữ liệu:** Mặc dù tập dữ liệu có số lượng positive/negative labels cân bằng, nhưng vẫn tồn tại sự mất cân bằng trong tần số xuất hiện của các từ/cụm từ mang cảm xúc, điều này có thể ảnh hưởng đến hiệu suất của mô hình.
- **Tham số mô hình:** Các tham số của các mô hình, đặc biệt là Random Forest và mô hình Deep Learning, có thể chưa được tối ưu hoàn toàn. Việc tinh chỉnh tham số (tuning) tốn nhiều thời gian, công sức, do đó trong phạm vi dự án này, các tham số được chọn dựa trên kinh nghiệm và thử nghiệm ban đầu.
- **Hạn chế về tài nguyên và triển khai**: Việc chạy thử nghiệm trên máy ảo đơn lẻ cũng phần nào giới hạn khả năng tính toán và đánh giá hiệu suất của các mô hình trên dữ liệu lớn hơn. Việc triển khai mô hình Naive Bayes trên Hadoop MapReduce còn thủ công và chưa tối ưu.
- **Phạm vi nghiên cứu:** Dự án này mới chỉ tập trung vào việc phân loại cảm xúc thành hai lớp Positive và Negative. Việc phân loại cảm xúc thành nhiều lớp hơn (ví dụ: Neutral, Angry, Happy, Sad, v.v.) hoặc phân tích các khía cạnh cảm xúc cụ thể hơn chưa được thực hiện.

8.3 | Hướng phát triển

Để cải thiện và mở rộng dự án, các hướng phát triển trong tương lai có thể bao gồm:

- **Thử nghiệm trên các tập dữ liệu khác:** Đánh giá hiệu suất của các mô hình trên các tập dữ liệu khác nhau, với các đặc điểm và nguồn gốc dữ liệu khác nhau, để kiểm tra tính khái quát hóa của mô hình.
- **Cải tiến biểu diễn đặc trưng:**
  - Sử dụng các kỹ thuật biểu diễn đặc trưng tiên tiến hơn như Word Embeddings (Word2Vec, GloVe, FastText) cho các mô hình truyền thống, TF-IDF kết hợp n-grams để nắm bắt tốt hơn ngữ nghĩa và ngữ cảnh của từ.
  - Tinh chỉnh và tối ưu hóa việc sử dụng 'BertEmbeddings' trong mô hình Deep Learning, thử nghiệm với các pre-trained models khác.
  - Kết hợp thêm các đặc trưng ngôn ngữ khác như Part-of-Speech tags, Named Entities, v.v., để cung cấp thêm thông tin cho mô hình.
- **Xử lý mất cân bằng dữ liệu:** Áp dụng các kỹ thuật xử lý mất cân bằng dữ liệu như oversampling, undersampling, SMOTE để cải thiện hiệu suất của mô hình, đặc biệt là trên các lớp thiểu số.
- **Tối ưu hóa tham số:** Sử dụng các kỹ thuật tối ưu hóa tham số như Grid Search, Random Search, Bayesian Optimization để tìm ra bộ tham số tối ưu cho từng mô hình, từ đó nâng cao hiệu suất.

- **Thử nghiệm với các mô hình khác:** Tiếp tục nghiên cứu, triển khai và đánh giá các mô hình học sâu (Deep Learning) như Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), và các mô hình dựa trên Transformers (BERT, RoBERTa) để so sánh hiệu suất với các mô hình truyền thống và mô hình 'ClassifierDLApproach' hiện tại.
- **Phân tích cảm xúc đa lớp và theo khía cạnh:** Mở rộng dự án để phân loại cảm xúc thành nhiều lớp hơn và phân tích cảm xúc theo từng khía cạnh cụ thể của sản phẩm/dịch vụ, giúp cung cấp thông tin chi tiết và hữu ích hơn cho người dùng.
- **Triển khai trên cụm máy:** Chạy thử nghiệm trên cụm máy Hadoop/Spark thực tế để đánh giá khả năng mở rộng và hiệu suất của các mô hình trên dữ liệu lớn hơn, từ đó tiến gần hơn đến việc triển khai thực tế.
- **Xây dựng hệ thống thời gian thực:** Phát triển hệ thống phân tích cảm xúc theo thời gian thực, có khả năng thu thập, xử lý và phân tích dữ liệu từ các nguồn trực tuyến (ví dụ: Twitter API), cung cấp thông tin cập nhật liên tục về cảm xúc của người dùng.
- **Tối ưu hóa triển khai mô hình trên Hadoop MapReduce**: Tự động hóa và tối ưu hóa quá trình triển khai mô hình Naive Bayes trên Hadoop MapReduce, ví dụ như sử dụng các công cụ hỗ trợ (Oozie, Azkaban) để quản lý workflow.

9 | Nhiệm vụ các thành viên trong dự án

Để đảm bảo tiến độ và chất lượng của dự án, các thành viên đã được phân công nhiệm vụ cụ thể dựa trên thế mạnh và sở trường của từng người. Bảng 9.1 mô tả chi tiết nhiệm vụ và đóng góp của từng thành viên trong nhóm.

Tên thành viên	MSSV	Nhiệm vụ cụ thể
Ngô Đức Hùng	22022652	1. Thực hiện tiền xử lý dữ liệu.
		2. Thực hiện EDA và trực quan hóa, phân tích.
		3. Triển khai và đánh giá các mô hình bằng Spark MLlib.
		4. Tổng hợp kết quả và viết báo cáo về Spark MLlib
		5. Xây dựng khung sườn báo cáo.
		6. Đóng góp ý kiến, hỗ trợ các phần khác
Nguyễn Công Huynh	22022565	1. Triển khai phân tích cảm xúc bằng Spark NLP.
		2. Tổng hợp kết quả và viết báo cáo về Spark NLP
		3. Kiểm thử, đánh giá, đảm bảo tính tin cậy của kết quả.
		4. Đảm bảo chất lượng mã nguồn.
		5. Đóng góp ý kiến, hỗ trợ các phần khác
Nguyễn Văn Trường	22022571	1. Triển khai phân tích cảm xúc bằng Hadoop MapReduce.
		2. Viết báo cáo các phần liên quan đến Hadoop MapReduce.
		3. Kiểm thử, đánh giá, đảm bảo tính tin cậy của kết quả.
		4. Thiết kế và chuẩn bị slide.
		5. Đóng góp ý kiến, hỗ trợ các phần khác.

Bảng 9.1: Nhiệm vụ đảm nhận

Tất cả các thành viên đều tham gia tích cực vào quá trình thảo luận, trao đổi ý tưởng, và cùng nhau giải quyết các vấn đề phát sinh trong quá trình thực hiện dự án. Nhóm đã sử dụng các công cụ quản lý phiên bản (Git) và nền tảng cộng tác trực tuyến (GitHub) để quản lý mã nguồn, theo dõi tiến độ, và chia sẻ tài liệu một cách hiệu quả.