Code file streaming.py  
Xử lý dữ liệu streaming sử dụng Apache Spark để phân tích cảm xúc của các tweet trong thời gian thực. Cụ thể, mã sẽ đọc các bài tweet từ một thư mục chứa các file CSV, áp dụng các bước tiền xử lý để làm sạch văn bản (bao gồm việc loại bỏ URL, ký tự đặc biệt, và chuyển đổi văn bản thành chữ thường), sau đó sử dụng một mô hình học máy đã được huấn luyện sẵn để phân loại cảm xúc của từng tweet vào các nhóm: Positive (Tích cực), Negative (Tiêu cực), Neutral (Trung lập), và Irrelevant (Không liên quan). Quá trình này được thực hiện theo dạng streaming (xử lý theo từng dòng dữ liệu) và mỗi batch dữ liệu được xử lý một cách độc lập, với kết quả dự đoán được in ra cho từng tweet, bao gồm cả nội dung gốc và nội dung đã làm sạch. Kết quả phân loại sẽ giúp người sử dụng hiểu được cảm xúc chung của các tweet theo thời gian thực, phục vụ cho các ứng dụng như phân tích xu hướng trên mạng xã hội, giám sát cảm xúc người dùng hoặc phân tích dữ liệu truyền thông. Toàn bộ quá trình chạy liên tục cho đến khi người dùng dừng lại, giúp hệ thống có thể xử lý và phân tích dữ liệu liên tục mà không bị gián đoạn.

|  |
| --- |
| import findspark  # Thư viện findspark giúp tìm kiếm và cấu hình đường dẫn Spark khi sử dụng Spark trong Python  findspark.init('C:\Spark\spark-3.5.3-bin-hadoop3')  # Khởi tạo Spark với đường dẫn tới thư mục cài đặt Spark (ở đây là C:\Spark\spark-3.5.3-bin-hadoop3)  import pyspark  # Nhập khẩu thư viện pyspark để sử dụng SparkContext và các tính năng Spark khác  sc = pyspark.SparkContext()  # Tạo một SparkContext, điểm bắt đầu cho mọi chương trình Spark  import re  # Nhập khẩu thư viện regex (regular expression) để xử lý các chuỗi văn bản  from pyspark.sql import SparkSession  # Import SparkSession từ pyspark.sql, giúp tạo phiên làm việc với Spark SQL  from pyspark.sql.functions import col, udf  # Import col và udf (user-defined functions) từ pyspark.sql.functions để thao tác với cột và tạo hàm tự định nghĩa  from pyspark.sql.types import StringType, StructType, StructField  # Import các kiểu dữ liệu để khai báo schema cho DataFrame  from pyspark.ml import PipelineModel  # Import PipelineModel để tải mô hình đã huấn luyện trong Spark MLlib  # Định nghĩa schema cho file CSV  schema = StructType([  # Định nghĩa cấu trúc của các trường trong file CSV      StructField("id", StringType(), True),  # Trường "id" là kiểu String và có thể là null      StructField("company", StringType(), True),  # Trường "company" là kiểu String và có thể là null      StructField("label", StringType(), True),  # Trường "label" là kiểu String và có thể là null      StructField("tweet", StringType(), True),  # Trường "tweet" là kiểu String và có thể là null  ])  # Tạo SparkSession và cấu hình cho master cluster  spark = SparkSession.builder.appName("Spark Streaming").master("spark://192.168.1.66:7077").config("spark.executor.memory", "4g").config("spark.executor.cores", "4").getOrCreate()  # Đọc dữ liệu từ file CSV  csv\_dir = "C:\Spark\spark\_streaming\data"  # Đường dẫn tới thư mục chứa dữ liệu CSV  df = spark \      .readStream \      .schema(schema) \      .option("header", "false") \      .csv(csv\_dir)      # Hàm để làm sạch văn bản (tiền xử lý văn bản)  def clean\_text(*text*):  # Hàm này nhận vào một chuỗi văn bản và trả về văn bản đã được làm sạch      if *text* is not None:  # Nếu văn bản không phải là None  *text* = re.sub(r'https?://*\S*+|www\.*\S*+|\.com*\S*+|youtu\.be/*\S*+', '', *text*)  # Loại bỏ các URL  *text* = re.sub(r'*(*@|#*)\w*+', '', *text*)  # Loại bỏ các ký hiệu @ và # trong văn bản  *text* = *text*.lower()  # Chuyển toàn bộ văn bản thành chữ thường  *text* = re.sub(r'[^a-zA-Z*\s*]', '', *text*)  # Loại bỏ các ký tự không phải là chữ cái hoặc khoảng trắng  *text* = re.sub(r'*\s*+', ' ', *text*).strip()  # Loại bỏ khoảng trắng thừa          return *text*  # Trả về văn bản đã làm sạch      else:          return ''  # Nếu văn bản là None, trả về chuỗi rỗng  # Định nghĩa UDF (User Defined Function) để áp dụng hàm làm sạch văn bản  clean\_text\_udf = udf(clean\_text, StringType())  # UDF sẽ trả về kiểu String sau khi làm sạch  # Tải mô hình đã huấn luyện từ file  pipeline = PipelineModel.load('C:\Spark\model\_spark\logistic\_regression\_model.pkl')  # Tải mô hình Logistic Regression đã huấn luyện từ file  # Định nghĩa hàm xử lý từng batch dữ liệu  class\_index\_mapping = {0: 'Negative', 1: 'Positive', 2: 'Neutral', 3: 'Irrelevant'}  # Mã hóa chỉ số thành nhãn cảm xúc  def process\_batch(*batch\_df*, *batch\_id*):  # Hàm này xử lý dữ liệu của từng batch trong dòng stream  *batch\_df* = *batch\_df*.withColumn("text", clean\_text\_udf(col("tweet")))  # Áp dụng UDF để làm sạch văn bản trong cột "tweet"      processed\_validation = pipeline.transform(*batch\_df*)  # Sử dụng mô hình đã huấn luyện để dự đoán      predictions = processed\_validation.select("tweet", "text", "prediction").collect()  # Lấy các kết quả dự đoán từ DataFrame      for row in predictions:  # Duyệt qua từng dòng dự đoán          tweet = row['tweet']  # Lấy tweet gốc          preprocessed\_tweet = row['text']  # Lấy tweet đã được làm sạch          prediction = row['prediction']  # Lấy chỉ số dự đoán của mô hình          prediction\_classname = class\_index\_mapping[int(prediction)]  # Lấy tên lớp dự đoán từ chỉ số          # In ra kết quả dự đoán          print("-> Tweet:", tweet)          print("-> preprocessed\_tweet:", preprocessed\_tweet)          print("-> Predicted Sentiment:", prediction)          print("-> Predicted Sentiment classname:", prediction\_classname)    # Khởi động streaming query  query = df.writeStream \      .foreachBatch(process\_batch) \      .outputMode('append') \      .start()  query.awaitTermination()  # Dừng SparkSession khi kết thúc  spark.stop() |