UNIVERSITÀ DELLA CALABRIA
DIPARTIMENTO DI
INGEGNERIA INFORMATICA,
MODELLISTICA, ELETTRONICA
E SISTEMISTICA
DIMES

Relazione per il progetto di Modelli e Tecniche per Big Data:

“ **Analisi delle catastrofi Sandy e Joplin attraverso il framework SPARK** ”

Studenti
Mattia Gatto 216649
Francesco Maria Granata 216648

Professori
Prof. Paolo Trunfio
Prof. Fabrizio Marozzo

Anno Accademico 2020/2021

Indice

1. Introduzione

Come si può notare nell'ultimo periodo le piattaforme di microblogging sono divenute uno strumento importante per condividere informazioni sul Web, specialmente durante eventi critici come i disastri naturali o altri eventi causati dall'uomo.
Negli ultimi anni nello specifico, Twitter è stato utilizzato per diffondere notizie su vittime, danni, donazioni, avvisi, e anche informazioni multimediali come video e foto.
Durante eventi catastrofici gli utenti, online, generano una quantità significativa di dati, alcuni dei quali sono estremamente preziosi per innumerevoli fini.
In questo progetto, viene effettuata un'analisi preliminare sul contenuto di un discreto numero di Tweet generati durante due differenti disastri naturali.
Nello specifico utilizziamo due Dataset di dati relativi alle seguenti emergenze:

- **Joplin 2011**: 206.764 tweet raccolti durante il tornado che ha colpito Joplin, Missouri (USA) il 22 maggio 2011. I ricercatori dell'Università del Colorado a Boulder hanno raccolto il set di dati tramite l'API di Twitter utilizzando l'hashtag "#joplin".
- **Sandy 2012**: 140.000 tweet raccolti durante l'uragano Sandy, che ha colpito il nord-est degli Stati Uniti il 29 ottobre 2012. Il set di dati è stato raccolto utilizzando gli hashtag "#sandy", "#nyc."

I Dataset forniti sono stati creati con un metodo in due fasi al fine di estrarre informazioni relative ai 2 disastri:

- Classificazione dei tweet;
- Estrazione dai tweet;

Poiché i messaggi generati durante un disastro sono estremamente vari, un sistema automatico deve iniziare filtrando i messaggi che non contribuiscono a informazioni preziose.
Questi includono quelli che sono interamente di natura personale e quelli non rilevanti per la crisi in corso. In particolare, iniziamo dividendo i messaggi in queste classi principali:

- **Personale**: se un messaggio interessa solo il suo autore e la sua cerchia immediata di familiari/amici e non trasmette alcuna informazione utile alle persone che non conoscono l'autore del Tweet.
- **Informativo**: se il messaggio è di interesse ad altre persone al di fuori

della cerchia immediata dell'autore. All’interno dei Tweet di tipo informativo, si distinguono tre tipi di messaggi:

- **diretti**, cioè scritti da una persona che è testimone oculare di ciò che sta accadendo;
- **indiretti**, quando il messaggio ripete informazioni riportate da altre fonti.
- **Altro**: se il messaggio non è correlato al disastro.

Ovviamente i Tweet più interessanti sono quelli di tipo informativo, quindi questo progetto si concentra maggiormente su di essi.

2. Preelaborazione dei dati

Inizialmente, prima di poter effettuare particolari query sui dati forniti, abbiamo visualizzato i Dataset in formato tabellare, sfruttando la libreria “pandas” del linguaggio Python, eliminando poi le colonne all’interno delle quali non vi erano informazioni utili. In particolare, diverse colonne erano relative alle tecniche di crowdsourcing utilizzate per etichettare i Tweet.
Tutto questo procedimento è stato notevolmente utile al fine di avere una visione più chiara delle potenzialità dei Dataset.

3. Framework e servizi utilizzati

Nel progetto sono stati utilizzati:

- Apache Spark: un **framework** open-source, per il processamento di dati su larga scala.
- Apache Livy: un **servizio** che consente un facile invio di processi Spark o frammenti di codice Spark, nonché gestione del contesto Spark, il tutto tramite una semplice interfaccia REST. Apache Livy semplifica anche l'interazione tra Spark e i server delle applicazioni, consentendo così l'uso di Spark sia per applicazioni locali che web.

3.1. Apache Spark

Per quanto riguarda l'elaborazione dei dataset sfruttiamo Apache Spark, attraverso il linguaggio di programmazione Scala, gestendo le dipendenze e la fase di compilazione con il build tool "sbt".

Nella porzione del progetto sviluppata in Scala, abbiamo creato differenti oggetti, ciascuno dei quali opera su un dataset specifico tra i file csv modificati durante la fase di preelaborazione dei Dati.

In ogni oggetto troviamo due differenti famiglie di query:

- Le query della prima famiglia restituiscono statistiche di tipo count sulle classi di Tweet presenti nel Dataset considerato.
- Le query della seconda famiglia restituiscono invece i Tweet di una certa classe oppure gli Utenti autori di tali messaggi.

In seguito, abbiamo definito un oggetto "Main" che prende in input un argomento che corrisponde al nome di una determinata query, e va a salvare in una directory "results" il file ".csv" ottenuto in base al risultato della query.

Infine, abbiamo generato attraverso il comando "sbt package" il file ".jar" relativo a questa parte di progetto, che verrà poi utilizzato dal servizio di Apache Livy.

3.2 Apache Livy

Attraverso tale servizio, creiamo un'interfaccia tra il cluster Spark e l'applicazione client che abbiamo scritto in Python, nello specifico Livy ci

permetterà di creare un'API REST che il client potrà utilizzare con semplici richieste "http".
In particolare, abbiamo inserito due cartelle all'interno della directory bin di apache Livy:

- "results": una cartella nella quale abbiamo salvato i risultati delle query in formato csv.
- "dataset": una cartella nella quale abbiamo inserito i dataset preelaborati che verranno poi letti durante l'esecuzione delle query.

Il server Livy è in grado di sottoporre al cluster Spark i "job" che il client indica, insieme ad eventuali parametri, all'interno di richieste POST a lui dirette.
Il nostro "job" Spark rappresentato dal binario ottenuto attraverso sbt, verrà eseguito dal cluster e in base agli argomenti definiti dall'utente, l'oggetto "Main" eseguirà l'opportuna query e scriverà il risultato dentro la cartella "results".
Il client per inviare richieste al server Livy, quindi per richiedere l'esecuzione di una determinata query e informarsi sulla disponibilità dei corrispondenti risultati utilizza il modulo "livy.py".
Il dialogo con il server Livy avviene attraverso due particolari metodi:

- Con il metodo "livy_submit_job" che prende come parametro un argomento, andiamo a:
  - Indicare il path assoluto per accedere all'eseguibile ".jar" che rappresenta il "job" Spark;
  - Indicare l'url del server Livy;
  - Costruire la richiesta di tipo POST che contiene (in formato json) tutti i dati di cui Livy necessita per fare la submit del "job". Ci viene restituita una risposta in formato json.

  Alla fine, andremo a restituire l' "id" relativo alla risposta del server Livy.
- Con il metodo "livy_job_success" che prende come parametro l'"id" del job eseguito successivamente alla richiesta POST, andiamo a interrogare il server Livy attraverso richieste GET ogni 5 secondi a proposito dello stato del job. Se lo stato del job è "dead" il metodo ritorna False, altrimenti se è "success" ritorna True.

4.Grafica

Per quello che riguarda la componente grafica dell'applicazione abbiamo deciso di utilizzare un particolare modulo Python, chiamato TKinter.
Attraverso questo modulo, abbiamo ideato un'interfaccia grafica che gestisce tre tipi di Query:

- Esclusivamente rivolte alla catastrofe Sandy;
- Esclusivamente rivolte alla catastrofe Joplin;
- Rivolte ad effettuare contronti tra le due catastrofi;

Attraverso il modulo "results.py" abbiamo definito due metodi che hanno lo scopo di prelevare in modo opportuno i risultati:

- Con il metodo "tweet_list" andiamo a selezionare l'opportuno risultato (in base al nome di query passato come parametro) tra i risultati della famiglia di query che si occupa di restituire Tweet di una determinata classe o gli utenti che hanno generato tali Tweet. Alla fine del processo andremo a restuire un risultato sottoforma di lista.
- Con il metodo "statistics_csv" " andiamo a selezionare invece, l'opportuno risultato (in base al nome di query passato come parametro) però questa volta tra i risultati della famiglia di query che si occupa di restituire Statistiche di tipo count sulle classi di Tweet presenti nel Dataset considerato. Alla fine del processo andremo a restuire un risultato sottoforma di DataFrame pandas.

La visualizzazione dei dati ottenuti è stata effettuata grazie all'utilizzo di due metodi di cui :

- il primo, "stampaTweets" va a rappresentare in un campo di testo i risultati che si manifestano sotto forma di liste di Tweet o di Utenti.
- Il secondo, "plotData" va a rappresentare in un oggetto plot appartente al modulo Python matplotlib, i risultati che si manifestano sotto forma di DataFrame pandas.

Per concludere ecco la rappresentazione grafica dell'interfaccia all'avvio dell'applicazione.