Università degli Studi della Calabria

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

---

Corso di Laurea Magistrale in Ingegneria Informatica

Relazione Progetto Modelli e Tecniche Per Big Data

Harvey Tracker

Daniele Francesca Mafalda Matr. 247108

Gugliotta Carmelo Matr. 247032

Anno Accademico 2022-23

Indice

1 Introduzione

L'Uragano Harvey del 2017 rappresentò un evento catastrofico con conseguenze disastrose che hanno profondamente colpito la vita di milioni di individui. In tale contesto, i social media hanno svolto un ruolo di primaria importanza nella condivisione di informazioni, esperienze e richieste di aiuto. Nel presente elaborato, viene presentato un progetto volto all'analisi dei tweet correlati all'Uragano Harvey al fine di estrarre informazioni rilevanti e offrire una rappresentazione facilmente comprensibile delle stesse.

Al fine di gestire l'analisi dei dati provenienti dai tweet, si è fatto uso delle funzionalità offerte da Apache Spark. Tale framework ha consentito di effettuare l'estrapolazione delle informazioni in modo efficiente, sfruttando la scalabilità e la velocità di elaborazione caratteristiche di Spark.

Parallelamente, si è sviluppato un algoritmo di machine learning per la classificazione dei tweet. Attraverso l'utilizzo dell'apprendimento automatico, è stato possibile identificare e categorizzare i tweet in base a specifici contenuti, quali richieste di aiuto, informazioni di soccorso e altre tematiche connesse all'emergenza. Ciò ha permesso di ottenere una comprensione più approfondita del contesto e di fornire informazioni utili per la gestione dell'emergenza.

Per garantire un'interfaccia intuitiva e facilmente fruibile, si è sviluppata una web app mediante il framework Angular. Ciò ha consentito di creare un'interfaccia interattiva che permette agli utenti di esplorare i dati in modo agevole. In particolare, si sono utilizzate mappe interattive per la geolocalizzazione dei post e diversi grafici per rappresentare le distribuzioni delle parole, delle menzioni e degli hashtag più utilizzati, offrendo così una visione chiara e immediata.

In sintesi, il progetto in oggetto integra l'analisi dei tweet mediante Apache Spark, un interfaccia intuitiva sviluppata mediante Angular e l'impiego di algoritmi di classificazione al fine di fornire un'analisi esaustiva dei dati relativi all'Uragano Harvey del 2017.

2 Analisi Dei Dati

Il dataset in esame, proveniente dallo studio di ricerca *"A Twitter Tale of Three Hurricanes: Harvey, Irma, and Maria"* è costituito da due componenti principali: una raccolta di tweet provenienti da Twitter durante i tre uragani - Harvey, Irma e Maria e un archivio contenente l'associazione tra l'identificativo univoco di ciascun tweet e una specifica categoria che indica il suo contenuto o argomento. Il periodo temporale è compreso tra il 25 agosto e il 3 ottobre del 2017. Questo intervallo temporale coincide principalmente con l'impatto dell'uragano Harvey. I tweet sono stati principalmente raccolti nella zona colpita dalla catastrofe, fornendo una preziosa fonte di informazioni sulla situazione e le reazioni delle persone coinvolte.

2.0.1 Raccolta di Tweet

La raccolta di tweet è organizzata in due dataset distinti, denominati "001" e "002". Il dataset "001" contiene un totale di 1.348.157 record (circa 9,74 GB), mentre il dataset "002" contiene 950.906 record (circa 7,14 GB). Tuttavia, è importante notare che durante l'analisi è stato individuato un record corrotto nel dataset "001", il quale è stato escluso dalle analisi successive. Di conseguenza, il dataset "001" risulta composto da 1.348.156 record validi.

Sulla base di queste considerazioni, l'attenzione è stata concentrata sul dataset "001" poichè l'utilizzo di un dataset più ampio contribuisce a fornire una visione più approfondita delle dinamiche dei tweet, consentendo di ottenere risultati più significativi e accurati nel contesto delle analisi effettuate.

É importante notare che, sebbene abbiano dimensioni significative, esiste anche la possibilità di unire i due dataset ("001" e "002") per ottenere una visione più completa e integrata dei dati. Tuttavia, l'unione dei dataset richiede una pianificazione e una gestione attenta delle risorse, poiché l'aumento delle dimensioni dello stesso comporta un maggiore consumo di memoria e risorse di elaborazione. Pertanto, per eseguire questa operazione, è necessario assicurarsi di avere a disposizione sufficienti risorse hardware per evitare problemi di prestazioni o limitazioni dovute alla capacità di elaborazione.

Entrambi i dataset sono costituiti da 37 colonne, tuttavia solo 21 di queste sono state considerate per la realizzazione dell'applicazione. In particolare, tra quelle considerate, ne esistono alcune utilizzate più frequentemente, quali:

- **Text** (`text`): Il testo del tweet.
- **User** (`user`): Contiene informazioni sull'utente che ha pubblicato il tweet. È una struttura (struct) con diversi campi, come ad esempio *created_at* che rappresenta la data e l'ora di creazione dell'account utente, *description* che contiene la descrizione fornita dall'utente nel suo profilo e *location* che rappresenta la posizione geografica indicata dall'utente nel suo profilo
- **Entities** (`entities`): Le entità forniscono metadati e informazioni contestuali aggiuntive sui contenuti pubblicati nel tweet. È una struttura ("struct") ed alcuni dei campi presenti sono i seguenti:
  - Hashtags (`hashtags`): Lista di stringhe che contiene gli hashtag contenuti nel tweet.
  - User Mentions (`user_mentions`): Lista di oggetti contenenti informazioni sugli utenti menzionati nel tweet, come l'ID e il nome utente.

– Media (`media`): Lista di oggetti contenenti informazioni sui media presenti nel tweet, come l'URL e il tipo di media.

In particolare questi array sono comodi nel momento in cui si analizzano i tweet, poiché Twitter ha essenzialmente elaborato o analizzato in precedenza il corpo del testo. Invece di dover cercare esplicitamente queste entità nel corpo del tweet (campo text), risultano immediatamente disponibili in questa sezione.

- **Quoted Status** (`quoted_status`): Tale colonna è una struttura ("struct") e rappresenta il tweet originale che è stato citato o incluso come citazione nel tweet. Le informazioni associate riguardano il testo, l'utente che ha pubblicato il tweet, la data di creazione e altre proprietà del tweet originale.

  Si osservi dunque che presenza della colonna *"quoted_status"* indica che il tweet corrente è una citazione di un altro tweet e fornisce un riferimento al tweet originale. Questo campo è utile per comprendere il contesto e le interazioni tra i tweet all'interno di una discussione su Twitter.

2.0.2 Archivio di Categorizzazione dei Tweet

Per quanto concerne il secondo componente del dataset, ossia l'archivio "harvey data 2017 aidr classification.txt" che come anticipato precedentemente consente di identificare e classificare i messaggi in base ai loro contenuti tematici e di analizzarli in modo più accurato e strutturato, esso risulta costituito da 4 colonne, di seguito elencate:

- **Tweet ID** (`TweetID`): Rappresenta l'identificativo univoco di ciascun tweet nel formato di stringa. Questo ID viene assegnato a ogni tweet per distinguerlo da tutti gli altri messaggi pubblicati su Twitter.

- **Date** (`Date`): Indica la data e l'ora in cui è stato creato il tweet. La colonna Date memorizza informazioni sulla data e sull'orario nel formato *"EEE MMM dd HH:mm:ss Z yyyy"* consentendo di eseguire filtri temporali sul dataset.

- **AIDR Label** (`AIDRLabel`): Rappresenta la categoria o l'etichetta assegnata a ciascun tweet attraverso l'utilizzo del sistema di classificazione AIDR (Artificial Intelligence for Disaster Response). Le categorie possibili sono elencate nella seguente tabella:

Tag	**Significato**
`injured_or_dead_people`	Indica informazioni o segnalazioni relative a persone ferite o decedute.
`relevant_information`	Riguarda informazioni rilevanti correlate all'evento o al disastro in questione.
`caution_and_advice`	Comprende avvisi, precauzioni o consigli utili per affrontare la situazione.
`displaced_and_evacuations`	Indica situazioni in cui le persone sono state sfollate o evacuate.
`sympathy_and_support`	Riguarda messaggi di sostegno, solidarietà o condoglianze.

Continua nella pagina successiva

Tabella 1 – Continuazione dalla pagina precedente

Tag	Significato
response_efforts	Comprende informazioni sulle operazioni di risposta o sulle iniziative intraprese.
infrastructure_and_utilities_damage	Indica danni alle infrastrutture e ai servizi pubblici.
personal	Riguarda tweet di natura personale o esperienze individuali.
affected_individual	Comprende tweet che descrivono come un individuo sia stato colpito o influenzato dalla situazione.
not_related_or_irrelevant	Indica tweet che non sono correlati all'evento o che sono considerati irrilevanti.
missing_and_found_people	Riguarda persone scomparse o ritrovate in relazione all'evento o al disastro.
donation_and_volunteering	Comprende informazioni o richieste relative a donazioni o volontariato.

- **AIDR Confidence** (`AIDRConfidence`): Indica il grado di affidabilità o la misura di confidenza associata alla classificazione del tweet da parte del sistema AIDR. Questo valore rappresenta una stima della precisione o dell'accuratezza della classificazione, espresso come un numero in virgola mobile (double), che può variare da 0 a 1. Un valore più vicino a 1 indica una maggiore fiducia nella classificazione del tweet, mentre un valore più vicino a 0 indica una minore fiducia o incertezza associata alla classificazione.

2.0.3 Problema di categorizzazione e Soluzione

Durante l'analisi dei dati, si è riscontrato che una parte dei record in entrambi i dataset non era stata correttamente categorizzata. Nel dataset 001, sono stati individuati 404 record che non erano stati assegnati a nessuna categoria, mentre nel dataset 002 sono stati trovati 365 record con la stessa problematica. Questo ha rappresentato una sfida significativa per l'analisi, poiché la categorizzazione corretta dei dati è essenziale per ottenere risultati accurati e significativi.

Per affrontare il problema della categorizzazione mancante dei record, abbiamo implementato l'algoritmo di classificazione Naïve Bayes. Questo algoritmo è stato scelto per la sua efficacia nella classificazione testuale e la sua capacità di gestire dataset di grandi dimensioni.

Innanzitutto, abbiamo eseguito una serie di passaggi di pre-elaborazione dei dati utilizzando diverse trasformazioni, come la pulizia del testo dei tweet, la rimozione delle stop words, l'applicazione di una rappresentazione vettoriale (HashingTF), il calcolo dell'IDF (Inverse Document Frequency) e l'indicizzazione delle etichette di categorizzazione.

Il dataset è stato quindi diviso in due parti: set di addestramento, e set di test. La suddivisione è stata effettuata in modo casuale, assegnando a ciascun subset una percentuale specifica dei dati totali.

Successivamente, abbiamo utilizzato il Cross-Validator di Apache Spark ML sul set di addestramento per selezionare il modello ottimale e i relativi parametri.

Infine, abbiamo applicato il modello al set di test e calcolato diverse metriche come accuracy, precision, recall e F1-score per valutare le prestazioni del modello su dati non visti in precedenza.

Il modello addestrato e testato sui dati categorizzati è stato poi utilizzato per assegnare correttamente le categorie ai record non categorizzati nel dataset.

Si osservi che il modello è stato presentato in modo semplicistico fino a questo punto, fornendo solo una panoramica generale delle sue caratteristiche e del processo di addestramento. Tuttavia, nelle prossime sezioni del progetto, esploreremo in modo più dettagliato il suo funzionamento.

2.0.4 Conclusioni

In questa sezione, abbiamo esaminato il dataset raccolto dai tweet durante gli uragani Harvey, Irma e Maria, nonché l'archivio di categorizzazione dei tweet associato. Abbiamo identificato le principali caratteristiche dei dati, compreso il numero di record, le colonne rilevanti e le categorie di classificazione disponibili.

Durante l'analisi dei dati, abbiamo affrontato la sfida della categorizzazione mancante dei record. Abbiamo adottato un approccio basato sull'algoritmo di classificazione Naïve Bayes e abbiamo eseguito diverse operazioni di pre-elaborazione dei dati per migliorare la qualità delle predizioni.

Nelle prossime sezioni del progetto, esploreremo nel dettaglio l'architettura dell'applicazione, le query, la rappresentazione grafica dei dati e il funzionamento del modello per fornire una comprensione più profonda dei risultati analizzati.

3 Architettura dell'applicazione

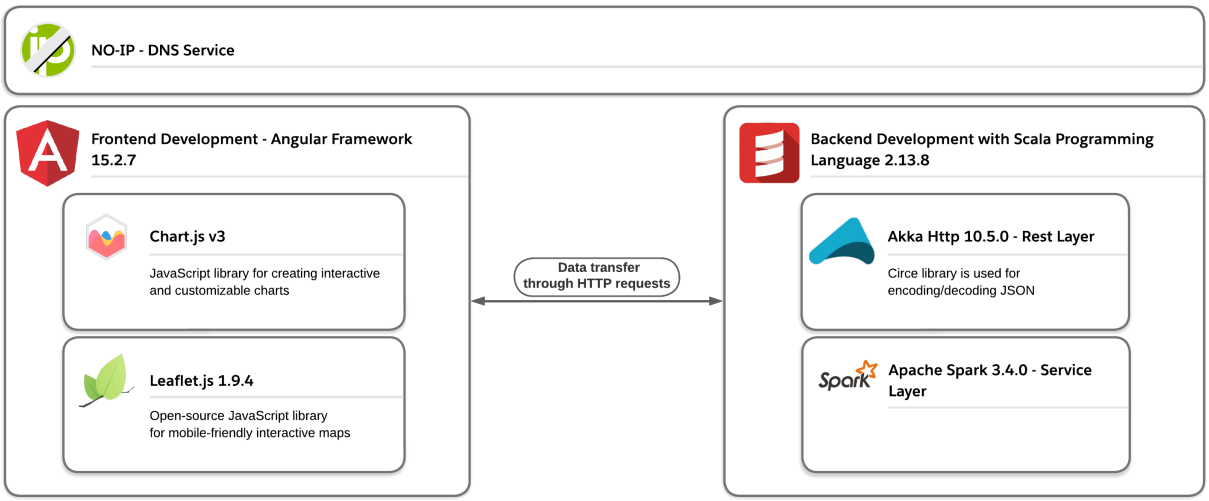


Figura 1: Architettura Applicazione Harvey Tracker

La Figura 1 illustra l'architettura complessiva dell'applicazione Harvey Tracker, che è costituita da diversi componenti che collaborano per fornire funzionalità avanzate.

Il front-end dell'applicazione è stato sviluppato utilizzando il framework Angular, insieme alle librerie Leaflet e Chart.js. Angular ha permesso di creare una web app che può essere utilizzata tramite browser e ciò significa che gli utenti possono accedere e utilizzare Harvey Tracker in modo comodo e intuitivo da un qualsiasi dispositivo connesso a Internet.

La libreria Leaflet è stata impiegata per la rappresentazione geografica dei dati del dataset. Grazie a Leaflet, è stato possibile inserire la visualizzazione di mappe interattive in grado di evidenziare le aree colpite dalla catastrofe di Harvey.

Inoltre, per la visualizzazione di grafici al fine di comprendere ulteriormente gli eventi legati alla catastrofe di Harvey, è stata utilizzata la libreria Chart.js che offre un'ampia varietà di tipi di grafici (come linee, barre, torte) e permette quindi di rappresentare i dati in modo chiaro e intuitivo.

Il back-end dell'applicazione è stato realizzato utilizzando il linguaggio di programmazione Scala che riesce a coniugare semplicità di stesura a leggibilità del codice. Il back-end risulta essere costituiro da due componenti principali: Lo strato REST e uno strato Service.

Lo strato REST è stato implementato tramite l'ausilio della libreria Akka Http che fornisce un'infrastruttura per la gestione delle richieste e delle risposte HTTP. Tramite i suoi meccanismi di routing e gestione degli endpoint, consente al front-end di comunicare con il back-end in modo efficiente.

Lo strato Service fa uso del framework Apache Spark per l'elaborazione e la manipolazione di grandi volumi di dati in modo efficiente e scalabile. Difatti uno dei vantaggi di Spark, rispetto ad altri framework, è l'elaborazione dei dati in memoria centrale: il dataset viene caricato in RAM in strutture dati resilienti (RDD, Dataset o Dataframe) e su di esse si effettuano le interrogazioni. Questo rappresenta un grosso vantaggio perché, a patto di avere RAM a sufficienza, si può

effettuare una sola lettura dei dati e successivamente effettuare qualsiasi tipo di elaborazione in memoria centrale. Differentemente, in Hadoop ad ogni iterazione del programma i dati vengono letti da HDFS, elaborati e il risultato prodotto viene scritto su HDFS. Nel caso di applicazioni con tante iterazioni, questo meccanismo si traduce in un costo abbastanza importante. Al contrario, però, se la memoria RAM non ha spazio a sufficienza, per ospitare l'intero dataset, Spark farà ricorso alla memorizzazione su disco di una parte dei dati con conseguente crollo delle prestazioni. Nel caso in questione, il dataset ha una dimensione che si aggira intorno ai 9 Gb, per cui per gestire tale quantità di dati è stato configurato Apache Spark per l'utilizzo della memoria e del disco. Per garantire prestazioni ottimali, è stato fondamentale utilizzare unità SSD (Solid State Drive) invece dei tradizionali dischi rigidi (HDD). Grazie all'uso degli SSD, si è evitato un crollo delle prestazioni e si è potuto sfruttare al massimo la potenza di calcolo di Apache Spark.

Per rendere accessibile l'applicazione Harvey Tracker su Internet, viene utilizzato il servizio di No-IP. No-IP permette di associare un nome di dominio personalizzato a un indirizzo IP dinamico, consentendo così di raggiungere l'applicazione attraverso un URL semplice e memorizzabile. Senza il servizio No-IP, sarebbe necessario utilizzare l'indirizzo IP dinamico, che può variare nel tempo, rendendo difficile l'accesso costante all'applicazione.

In sintesi l'architettura complessiva di Harvey Tracker sfrutta la potenza di Angular per creare un'interfaccia utente intuitiva e coinvolgente. Il back-end basato su Scala utilizza Akka HTTP per gestire le richieste del front-end e si avvale di Apache Spark per l'elaborazione dei dati. Questa combinazione di tecnologie offre un'applicazione robusta, scalabile e ad alte prestazioni.

4 Analisi Dettagliata del Back-End e dei Meccanismi delle Query

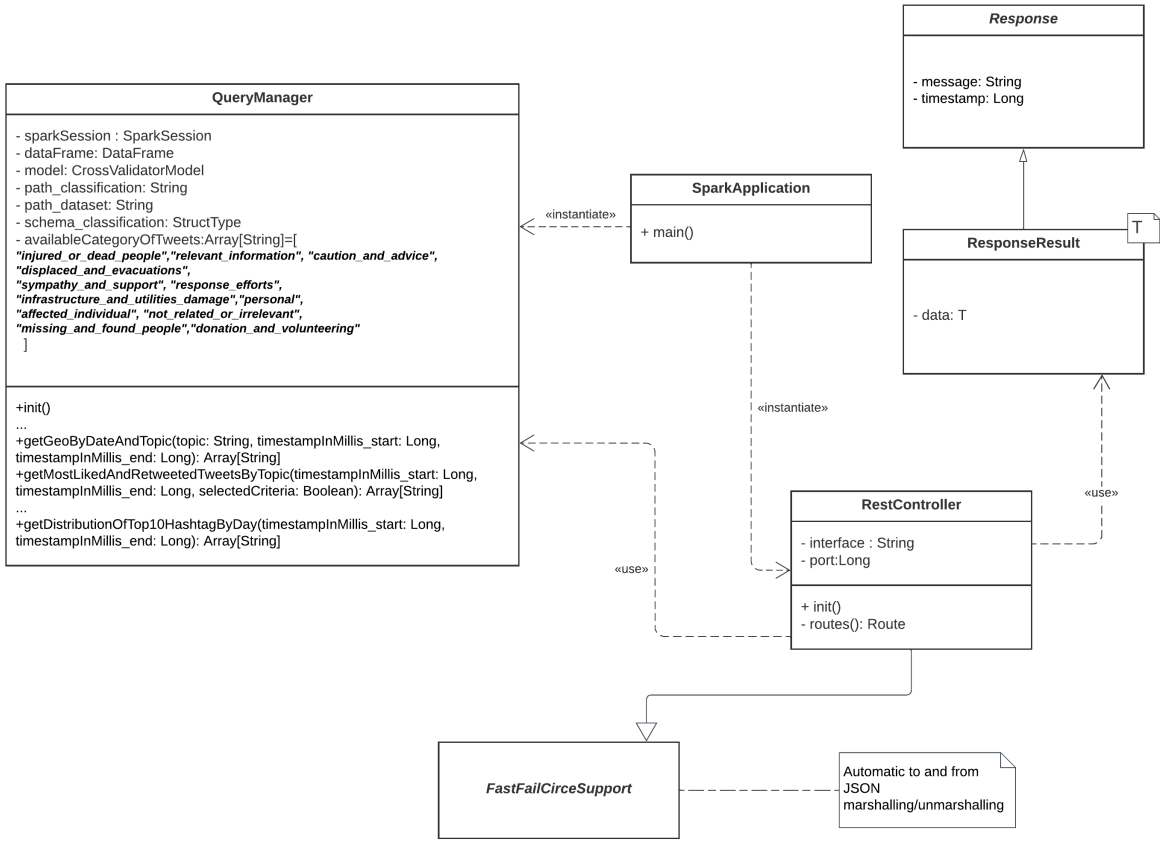


Figura 2: Class Diagram

La figura 2 svela alcuni aspetti che erano rimasti celati nella sezione precedente. Al centro di questi dettagli si trova la classe "**SparkApplication**", dichiarata come un oggetto singleton in Scala. Questo significa che viene creata un'unica istanza di questa classe all'interno dell'applicazione.

La classe "SparkApplication" riveste un ruolo fondamentale nell'ambito dell'applicazione Spark in questione. Essa assume il compito di essere il punto di ingresso principale, responsabile di avviare l'applicazione e coordinare una serie di operazioni.

Quando l'applicazione viene avviata, il metodo "main" di "SparkApplication" viene eseguito, il quale invoca il metodo init() sugli oggetti GestoreSpark e RestController, dichiarati anche essi come Oggetti Singleton.

La classe "**RestController**" è responsabile della gestione delle richieste HTTP in arrivo e dell'elaborazione delle relative risposte. Inoltre, questa classe implementa funzionalità di supporto per la serializzazione e la deserializzazione dei dati estendendo la classe FastFailCirceSupport.

All'interno della classe, vengono definiti diversi percorsi (routes) corrispondenti a diverse operazioni o query che l'applicazione può gestire. Ogni percorso (route) rappresenta un'unità logica di elaborazione per una specifica richiesta HTTP e definisce come l'applicazione deve rispondere a quella richiesta. Ad esempio, ci sono percorsi per ottenere informazioni sulle coordinate geografiche e gli ID dei tweet, per ottenere tweet in base a una determinata data e/o topic, e così via.

Ogni percorso è definito come una funzione all'interno della quale vengono specificati i parametri della richiesta e viene utilizzato il metodo "complete()" per restituire la risposta corrispondente. La risposta è incapsulata in un oggetto "ResponseResults" che contiene i dati richiesti, insieme a un messaggio descrittivo e un timestamp.

Infine, il metodo "init()" avvia il server HTTP utilizzando l'oggetto "Http()" e specifica l'indirizzo IP e la porta su cui il server sarà in ascolto.

La classe "**QueryManager**" svolge un ruolo fondamentale nel processo di configurazione dell'ambiente Spark e nell'esecuzione delle query. Il suo metodo init() è responsabile di tutte le operazioni necessarie per inizializzare correttamente l'ambiente di lavoro.

Innanzitutto, il metodo init() carica un'istanza di SparkSession, che rappresenta l'interfaccia principale per interagire con il framework Spark. Durante il caricamento, vengono impostati il nome dell'applicazione e alcune opzioni specifiche, tra cui la politica di analisi temporale da utilizzare in Spark SQL. Nel nostro caso, abbiamo scelto la politica "LEGACY", che si riferisce a un approccio più consolidato per l'elaborazione dei dati temporali all'interno di Spark SQL. Questa scelta è stata determinata dalla necessità di affrontare un problema specifico legato alla formattazione delle date all'interno del nostro dataset. In particolare, il problema riguardava la corretta lettura e interpretazione delle date presenti nei dati che stavamo analizzando.

In aggiunta, il metodo init() configura anche il percorso della directory locale in cui Spark dovrà archiviare i dati temporanei e i file di lavoro durante l'esecuzione delle query. Questo è utile per gestire in modo efficiente la memorizzazione temporanea dei dati e garantire prestazioni ottimali.

Successivamente, viene caricato il modello di classificazione. Questo modello è stato addestrato in precedenza utilizzando l'algoritmo Naive Bayes e permette di assegnare una categoria ai tweet in base al loro contenuto. Una volta caricato, otteniamo un oggetto CrossValidatorModel, che rappresenta il modello addestrato e pronto per essere utilizzato nelle successive fasi di classificazione dei tweet.

Una volta completata la configurazione iniziale e il caricamento del modello di classificazione, passiamo alla lettura dei dati dei tweet. Questi dati sono immagazzinati in file JSON e vengono rappresentati in memoria attraverso la struttura dati "Dataframe". Durante la lettura, i tweet vengono arricchiti con le rispettive categorie, che vengono estratte dall' *Archivio di Categorizzazione dei Tweet*.

Inoltre, nel processo di lettura, viene effettuata la sostituzione del testo troncato dei tweet con il testo completo, qualora presente nei record. Questo ci consente di avere una visione completa del contenuto dei tweet e facilita ulteriori analisi.

Infine, utilizzando il modello di classificazione precedentemente caricato, vengono classificati i tweet che non erano stati categorizzati in precedenza, al fine di ottenere un'analisi completa dei

dati.

Una volta che il metodo init() è completato con successo, tutti i componenti necessari sono pronti per iniziare a interrogare e analizzare i dati dei tweet utilizzando gli altri metodi disponibili nella classe "QueryManager".

4.1 Query 1 - Geo-localizzazione dei tweet (getGeoByDateAndTopic)

La prima query è progettata per ottenere la geo-localizzazione dei tweet in base a determinati criteri di filtraggio. Questa query riceve in input un topic d'interesse e un intervallo di tempo.

All'interno della query, vengono selezionate le colonne "TweetID", "geo", "Date" e "AIDR-Label" dal DataFrame. Successivamente, vengono applicati dei filtri per ottenere solo i tweet correlati al topic specificato, che rientrano nell'intervallo di tempo desiderato e che presentano informazioni di geo-localizzazione disponibili.

La geolocalizzazione dei tweet è stata rappresentata graficamente attraverso l'utilizzo di una mappa interattiva. Ogni tweet è stato posizionato sulla mappa in base alle sue coordinate geografiche e per differenziare visivamente i tweet associati a diversi topic di interesse, è stata adottata un'icona distintiva per ciascun topic.

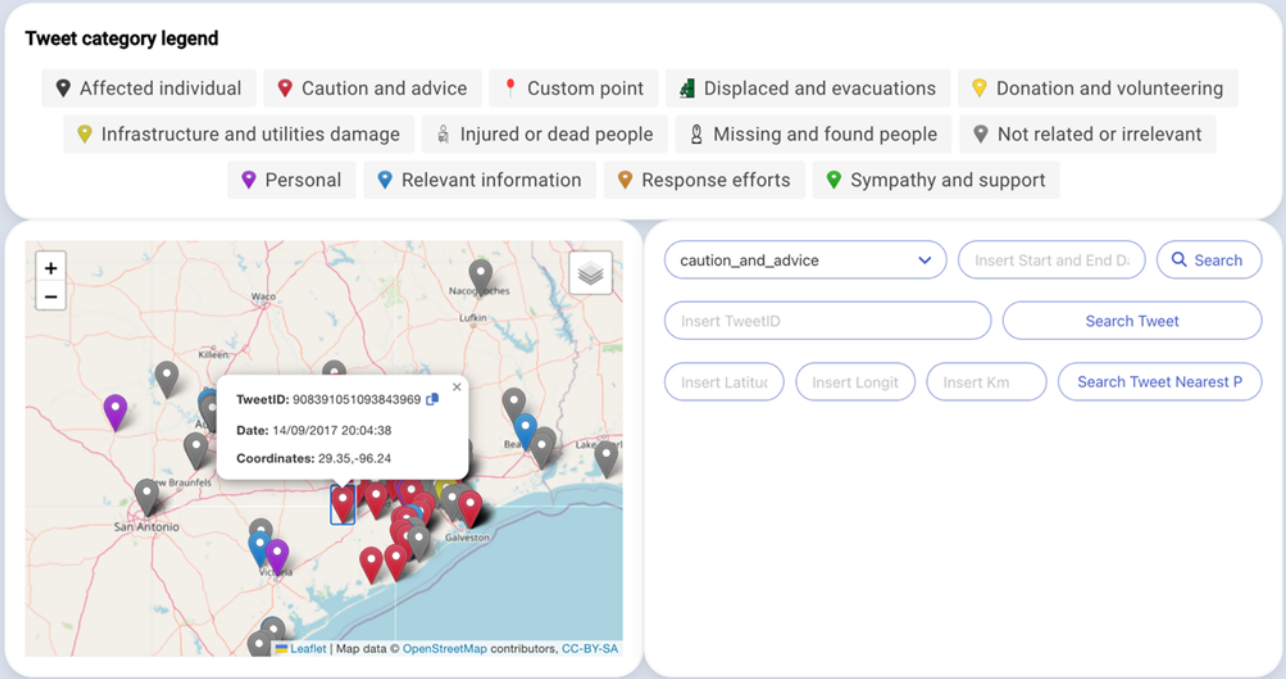


Figura 3: Query 1

I risultati possono essere utilizzati per identificare i luoghi in cui sono concentrate le attività correlate al topic d'interesse, evidenziare eventuali differenze territoriali o per comprendere le dinamiche spaziali associate al topic in esame.

4.2 Query 2 - I 10 Tweet più discussi (top10MostCommentedTweets)

La query restituisce i primi 10 tweet più commentati in base alla categoria selezionata o, se viene selezionata l'opzione "All", in base a tutte le categorie.

Inizialmente si effettua il filtraggio del DataFrame in base alla categoria selezionata. Successivamente, vengono estratte le colonne `is_quote_status`, `quoted_status_id` e `quoted_status` dal DataFrame selezionando solo i tweet che sono commenti di un altro tweet (`is_quote_status = true`). Si osservi che la colonna `quoted_status` detiene le informazioni riguardo il tweet originale.

Successivamente, i tweet vengono raggruppati in base all'ID del tweet originale (`quoted_status_id`), e si effettua il conteggio dei commenti per ciascun tweet al fine di selezionare i primi 10 tweet più commentati.

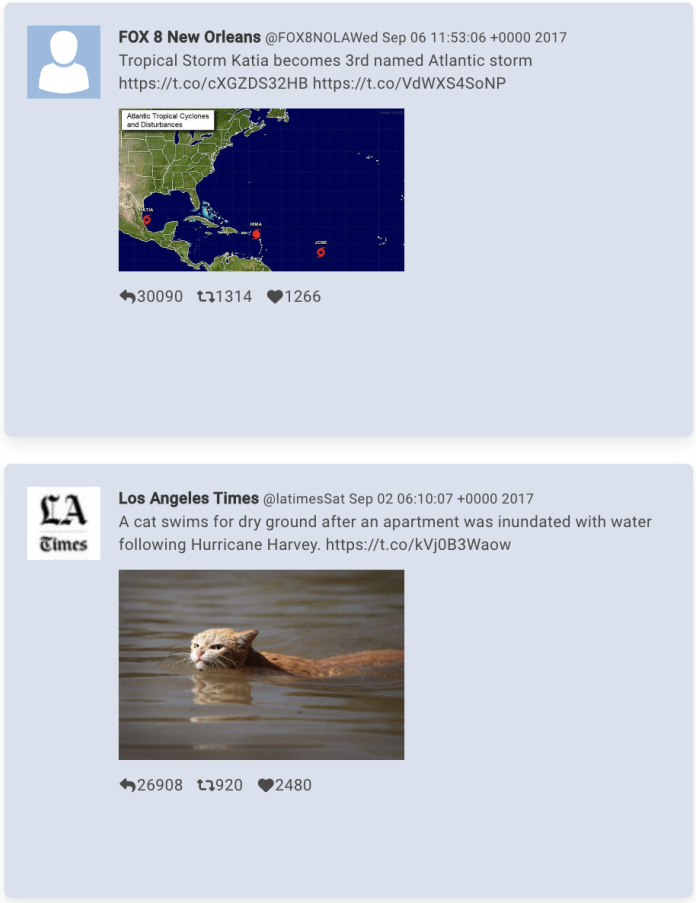


Figura 4: Query 2 - getGeoByDateAndTopic

Come illustrato dalla figura 4, per ogni tweet viene mostrato il testo, l'autore, il numero di like, commenti e retweet, oltre a eventuali allegati come immagini o video. I tweet vengono presentati in gruppi di due per pagina, consentendo agli utenti di scorrere in modo ordinato e semplice attraverso i risultati.

La query consente di identificare i tweet che hanno generato un elevato livello di discussione e interazione da parte degli utenti. Queste informazioni possono essere utilizzate per valutare la risonanza di determinati argomenti o per identificare tweet influenti all'interno di una specifica categoria.

4.3 Query 3 - Distribuzione dei tweet per giorno (getDistributionOfTweetsByDay)

La query prende in input due parametri, per definire l'intervallo temporale di interesse e restituisce il numero dei tweet pubblicati per ogni giorno, insieme alle informazioni sulla percentuale delle diverse categorie.

Listing 1: 3. getDistributionOfTweetsByDay

```
val categoryColumns = categoryNames.map(name => count(when
    (col("AIDRLabel") === name, 1)).as(name))
val res = dataFrame
  .select(date_trunc("day", col("Date")).as("Day"),
    col("AIDRLabel"))
  .filter(col("Day") >= timestamp_start &&
        col("Day") <= timestamp_end)
  .groupBy("Day")
  .agg(count(col("AIDRLabel")).as("Total"),
        categoryColumns.toList: _*)
  .withColumnRenamed("count", "Percentage")
  .withColumnRenamed("AIDRLabel", "Category")
  .orderBy("Day")
  .toJSON.collect()
res
```

In particolare vengono selezionate le colonne di interesse, tra cui la data (arrotondata al giorno) e l'etichetta della categoria di ciascun tweet. Viene applicato un filtro per includere solo i tweet che rientrano nel periodo specificato. Successivamente, i dati vengono raggruppati per giorno e vengono calcolati il numero totale di tweet e il conteggio per ciascuna categoria. I nomi delle colonne vengono rinominati per una migliore leggibilità.

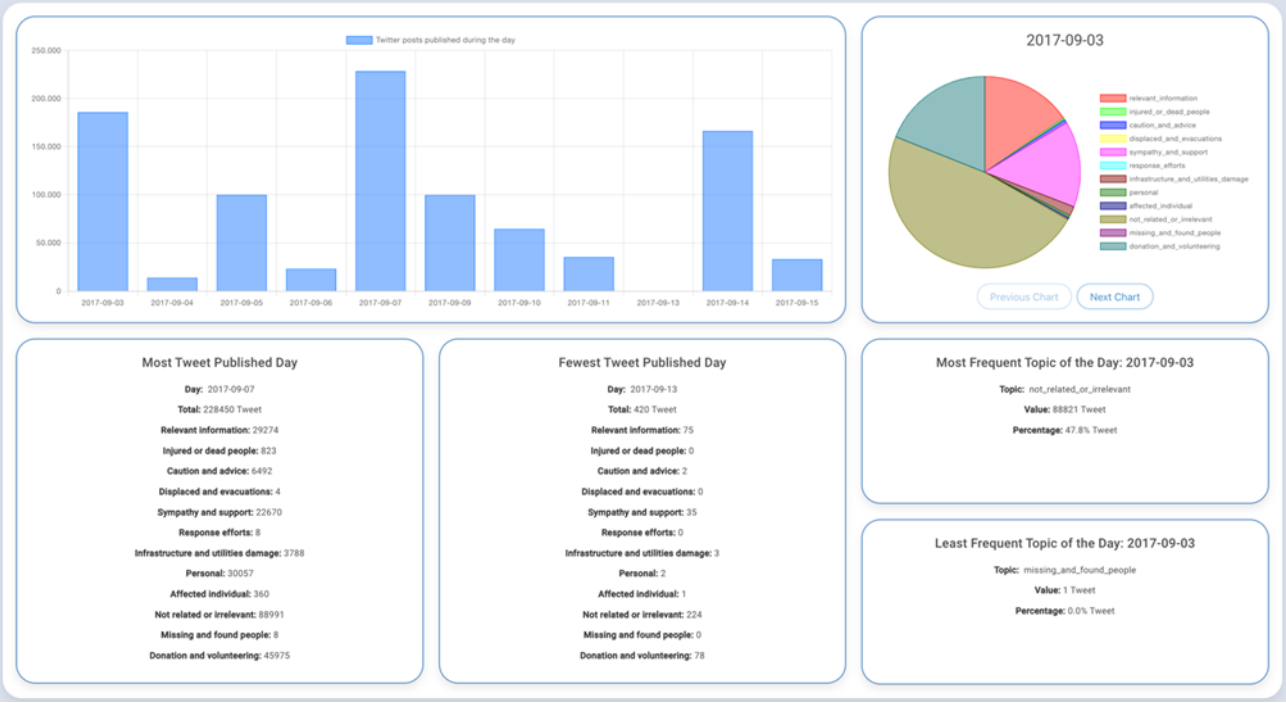


Figura 5: Query 3 - Distribuzione dei tweet per giorno

Sul front-end viene visualizzato un istogramma mostra le occorrenze dei tweet pubblicati giorno per giorno per l'intervallo temporale considerato e per ciascun giorno, un grafico a torta che rappresenta la percentuale dei topic d'interesse rispetto al totale dei tweet pubblicati. Questa visualizzazione consente di identificare rapidamente le variazioni di interesse nel corso del tempo e di individuare le categorie più rilevanti per ogni singolo giorno.

4.4 Query 4 - Utenti con il maggior numero di tweet (getNUserWithMostOfTweet)

La query "getNUserWithMostOfTweet" accetta due parametri in input: "N", che rappresenta il numero desiderato di utenti, e "selectedCategoriesOfTweets", un array di stringhe che contiene le categorie di tweet da considerare. La query restituisce gli "N" utenti con il maggior numero di tweet, fornendo per ogni categoria il conteggio dei tweet pubblicati e la confidenza media dell'AIDR (Artificial Intelligence for Disaster Response) nella classificazione di tali tweet nella rispettiva categoria.

Listing 2: 3. getNUserWithMostOfTweet

```
val filteredDF = dataFrame
.filter(col("AIDRLabel").isin(selectedCateogoriesOfTweets: _*))

val TopUserIDTweetsCount = ... //utenti che hanno pubblicato di piu'

val users = ... /*prelevo le informazioni (nome utente ecc)
piu' aggiornate riguardo i TopUser*/

val res = filteredDF
    .select("user", "AIDRLabel", "AIDRConfidence")
    .groupBy("user.id")
    .agg(
        count(col("AIDRLabel")).as("Total"),
        selectedCateogoriesOfTweets.map(name =>
        array(
            count(when(col("AIDRLabel") === name, 1)),
            avg(when(col("AIDRLabel") === name,
            col("AIDRConfidence")))
        ).as(name)): _*
    )
  .withColumn("max_topic_value", greatest(categoryColumns: _*))
  .withColumn("max_topic_name", max_topic_nameExpr)
  .join(users, col("id") === users("user.id"), "inner")
  .orderBy(desc("Total"))
  .toJSON.collect()
res
```

Inizialmente, vengono identificati gli utenti con il maggior numero di tweet nelle categorie selezionate. Successivamente, per ogni utente vengono calcolati il conteggio totale dei tweet di ogni utente e le statistiche delle categorie selezionate, come il numero di tweet pubblicati in quella categoria e la media della fiducia associata alla stessa, determinando anche la categoria in cui sono stati pubblicati più tweet.

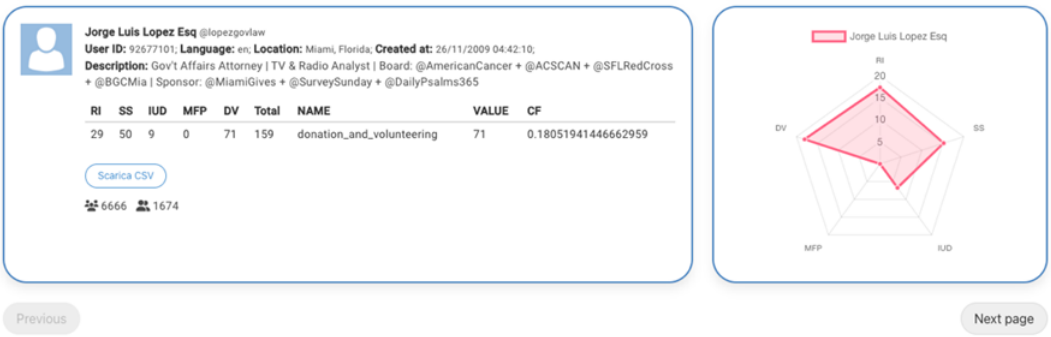


Figura 6: Query 4 - getNUserWithMostOfTweet

Sul front-end viene mostrato il profilo degli N utenti desiderati riportando una tabella scaricabile sotto forma di un file CSV che fornisce il numero totale di tweet pubblicati dall'utente, nonché il conteggio per ogni categoria selezionata, evidenziando la categoria con il maggior numero di tweet la confidenza media dell'AIDR associata alla stessa. Inoltre, viene mostrata una rappresentazione grafica della confidenza media dell'AIDR relativa a ciascuna categoria.

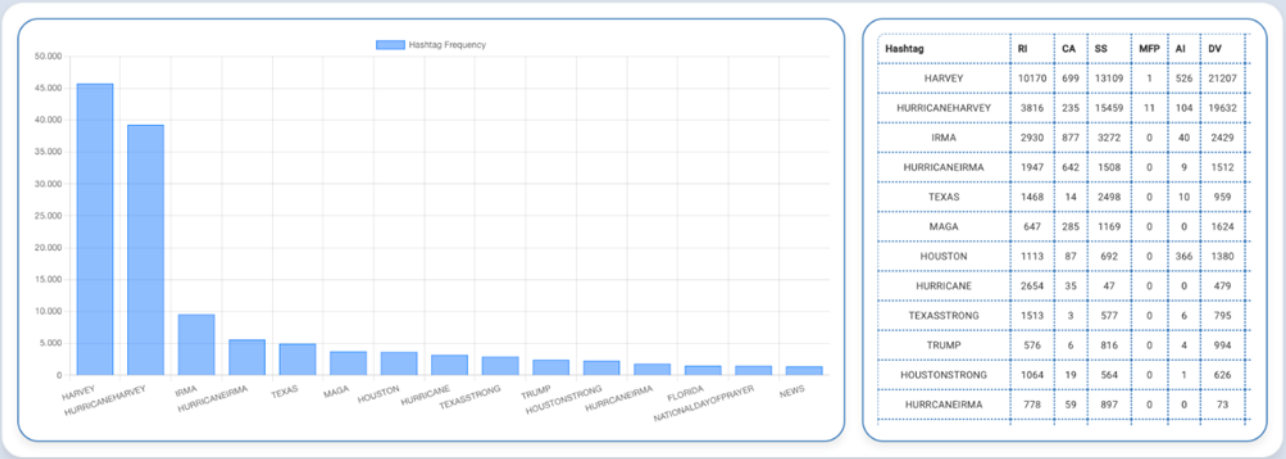
La query risulta essere utile per identificare e ottenere informazioni sui principali utenti che contribuiscono con il maggior numero di tweet nelle categorie specificate e comprendere e/o valutare le competenze degli stessi in relazione alle diverse tematiche.

4.5 Query 5 - Hashtag più popolari (getFirstNHashtag)

La funzione "getFirstNHashtag" è progettata per restituire gli hashtag più popolari correlati a una specifica categoria.

La prima parte della query filtra il DataFrame utilizzando le categorie selezionate, restituendo solo le righe corrispondenti agli argomenti specificati. Successivamente, viene selezionato l'array di hashtag all'interno delle entità associate a ciascun tweet e trasformato in un vettore denominato `"hashtags_vector"`, mantenendo l'etichetta del tweet associata a ciascun hashtag. Nella parte successiva si genera un nuovo DataFrame "expandedDF", ottenuto espandendo il vettore di hashtag, in cui ogni riga rappresenta un singolo hashtag associato a una categoria.

Si prosegue raggruppando per hashtag e calcolando il numero totale di occorrenze in cui ciascuno di essi compare, insieme al conteggio specifico per ciascuna delle etichette dei tweet selezionate. Questi conteggi vengono restituiti come colonne nel DataFrame risultante. Successivamente, il DataFrame viene ordinato in base al numero totale di occorrenze degli hashtag in ordine decrescente e viene limitato ai primi "n" risultati.



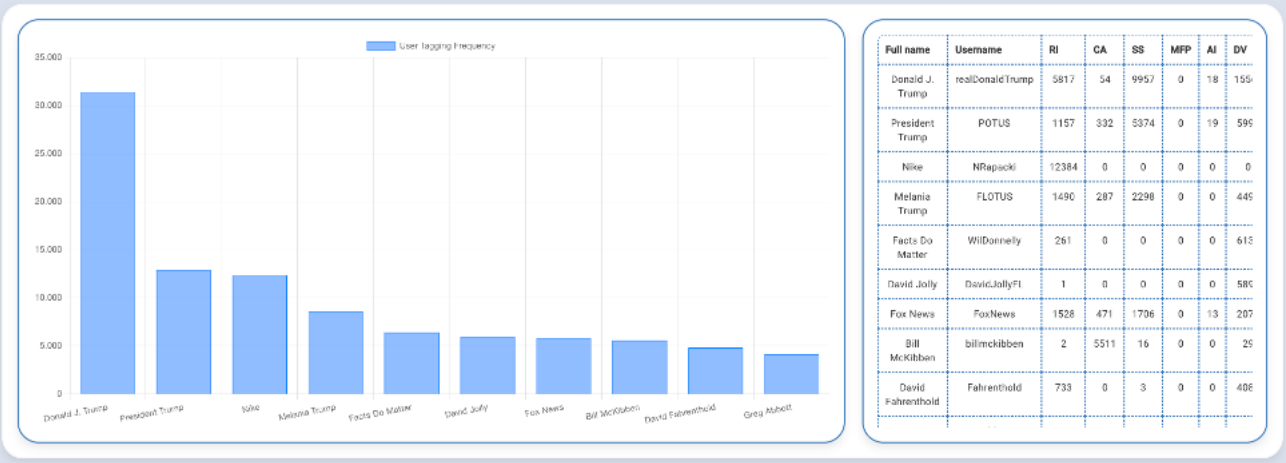
Hashtag	RI	CA	SS	MFP	AI	DV
HARVEY	10170	699	13109	1	526	21207
HURRICANEHARVEY	3816	235	15459	11	104	19632
IRMA	2930	877	3272	0	40	2429
HURRICANEIRMA	1947	642	1508	0	9	1512
TEXAS	1468	14	2498	0	10	959
MAGA	647	285	1169	0	0	1624
HOUSTON	1113	87	692	0	366	1380
HURRICANE	2654	35	47	0	0	479
TEXASSTRONG	1513	3	577	0	6	795
TRUMP	576	6	816	0	4	994
HOUSTONSTRONG	1064	19	564	0	1	626
HURRCANEIRMA	778	59	897	0	0	73

Figura 7: Query 5 - getFirstNHashtag

Queste informazioni possono essere utilizzate per comprendere meglio le tendenze e i modelli di utilizzo degli hashtag relativi a determinati argomenti. Tale analisi può fornire indicazioni sulle preferenze degli utenti, suggerire tematiche rilevanti o identificare influencer o comunità specifiche associate a un determinato argomento.

4.6 Query 6 - Menzioni più popolari (getFirstNMention)

La query "getFirstNMention" opera in modo analogo alla query precedente, ma con l'obiettivo specifico di restituire le menzioni più ricorrenti. Questa espande il vettore delle menzioni, creando righe distinte per ciascuna di esse, al fine di calcolarne la frequenza. Il risultato viene quindi raggruppato per menzione e viene conteggiato il numero di occorrenze per ognuna. L'output viene poi ordinato in modo decrescente e restituiti solo i primi "n" risultati.



Full name	Username	RI	CA	SS	MFP	AI	DV
Donald J. Trump	realDonaldTrump	5817	54	9957	0	18	155
President Trump	POTUS	1157	332	5374	0	19	595
Nike	NRepacki	12384	0	0	0	0	0
Melania Trump	FLOTUS	1490	287	2298	0	0	449
Facts Do Matter	WilDonnelly	261	0	0	0	0	613
David Jolly	DavidJollyFl	1	0	0	0	0	589
Fox News	FoxNews	1528	471	1706	0	13	207
Bill McKibben	billmckibben	2	5511	16	0	0	29
David Fahrenthold	Fahrenthold	733	0	3	0	0	408

Figura 8: Query 6 - getFirstNMention

La query fornisce un'analisi dettagliata per identificare gli utenti che esercitano un'influenza significativa sulla comunità o sull'argomento in questione, contribuendo così a comprendere le dinamiche sociali coinvolte.

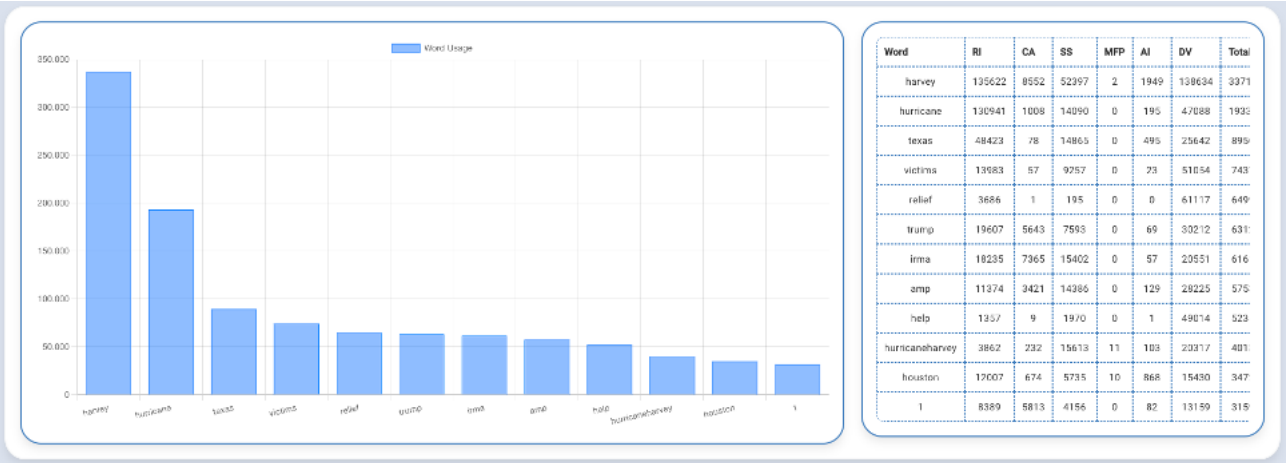
4.7 Query 7 - Parole piu popolari (getTopNWordsByTopic)

La funzione ‘getTopNWordsByTopic‘ è progettata per estrarre le N parole più comuni associate alle categorie selezionate.

Inizialmente, il DataFrame viene filtrato in base agli argomenti selezionati, creandone uno nuovo chiamato `"filteredByTopic_Df"`. Successivamente, vengono applicate una serie di operazioni di pulizia del testo al fine di rimuovere link, emoji e caratteri speciali, utilizzando la segunete espressione regolare:

```
val regex = "(?i)\\B@\\w+\\b|\\B#\\w+\\b|https?://\\S+|[\\p{So
  }\\p{Cntrl}\\n\\p{Punct}]+|RT|[^\\\\x00-\\\\x7F]+"
```

Il testo pulito viene quindi suddiviso in parole, utilizzando un tokenizzatore, si procede poi alla rimozione delle ”stop words”, ossia quelle parole comuni ma poco informative. Queste informazioni vengono raggruppate per parola e aggregate per ottenere il numero totale di occorrenze, i risultati vengono ordinati e limitati ai primi N.



Word	RI	CA	SS	MFP	AI	DV	Total
harvey	135622	8552	52397	2	1949	138634	3371
hurricane	130941	1008	14090	0	195	47088	1933
texas	48423	78	14865	0	495	25642	895
victims	13983	57	9257	0	23	51854	743
relief	3686	1	195	0	0	61117	649
trump	19607	5643	7593	0	69	30212	631
irma	18235	7365	15402	0	57	20551	616
amp	11374	3421	14386	0	129	28225	575
help	1357	9	1970	0	1	49014	523
hurricaneharvey	3862	232	15613	11	103	20317	481
houston	12607	674	5735	10	868	15480	347
1	8389	5813	4156	0	82	13159	315

Figura 9: Query 7 - getTopNWordsByTopic

La query è utile per analizzare il contenuto dei tweet relativi a specifiche categorie e identificare le parole più frequenti, rivelando potenziali tendenze o temi ricorrenti.

4.8 Query 8 - Distribuzione dei 10 Hashtag più popolari per giorno (getDistributionOfTop10HashtagByDay)

La query ”getDistributionOfTop10HashtagByDay” restituisce la distribuzione degli hashtag durante un intervallo di tempo specificato in input.

Inizialmente si creano le colonne corrispondenti a ciascuna categoria, dopodiché vengono estratti gli hashtag dai tweet durante l’intervallo di tempo. Viene creata una colonna `"hashtags_vector"` contenente gli hashtag estratti per ogni tweet, insieme alla data (troncata al giorno) e all’etichetta della categoria associata al tweet. Successivamente, vengono identificati i 10 hashtag più popolari.

Infine, eseguiamo un raggruppamento per hashtag e giorno, calcolando il conteggio totale per ogni giorno e per ciascuna categoria. Questo ci fornisce informazioni dettagliate sugli hashtag selezionati. In seguito, viene creata una struttura dati che include le informazioni necessarie per ogni giorno, come gli attributi delle categorie, la data e il conteggio totale.

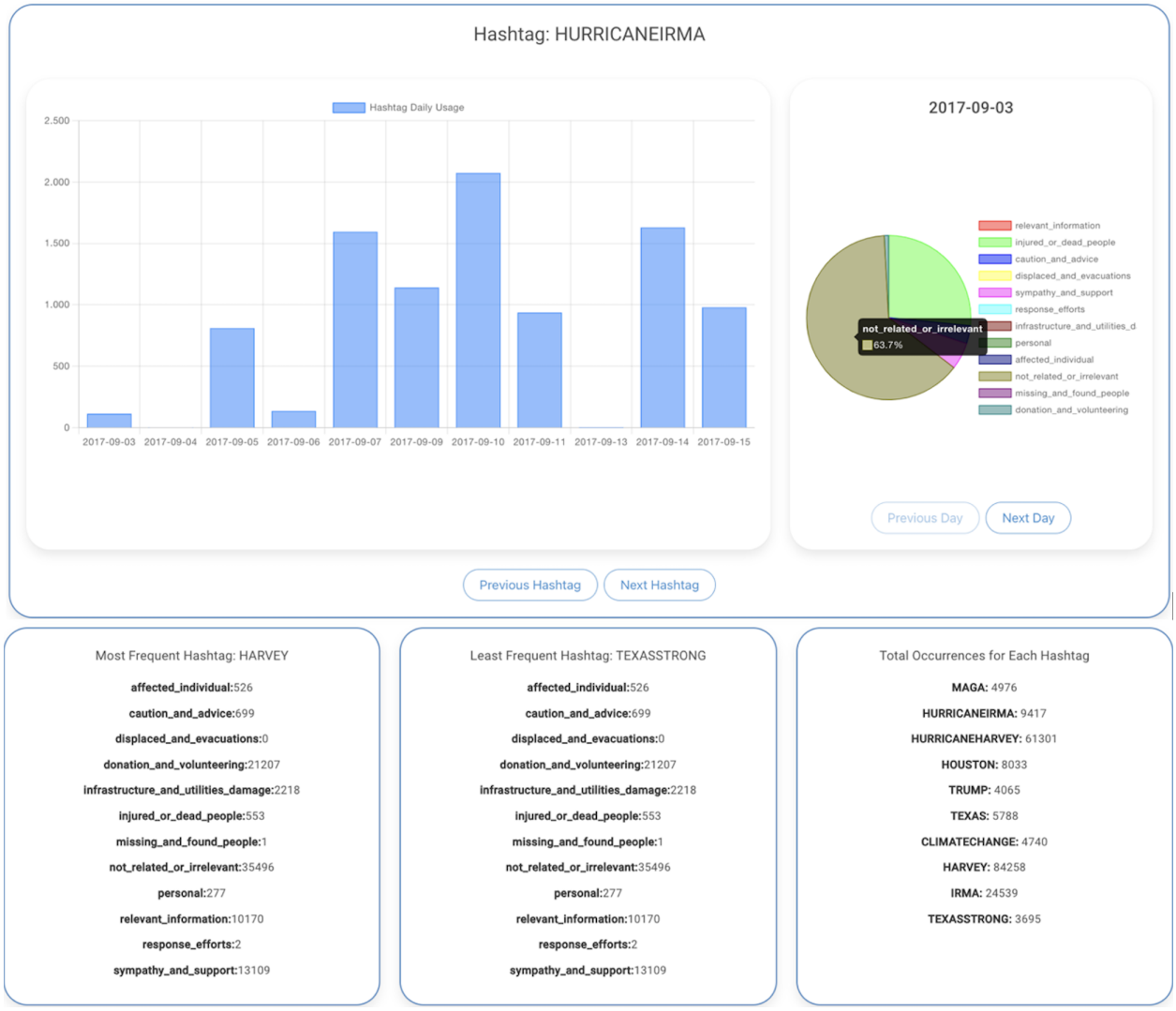


Figura 10: Query 8 - getDistributionOfTop10HashtagByDay

La rappresentazione comprende un istogramma che visualizza la distribuzione degli hashtag per ogni giorno nell'intervallo di tempo selezionato. Accanto all'istogramma, è stato incluso un grafico a torta che fornisce la percentuale di ogni categoria associata agli hashtag.

Attraverso questa query, è possibile ottenere una panoramica dettagliata della distribuzione degli hashtag, consentendo l'identificazione dei principali trend e la comprensione delle categorie di tweet più correlate agli hashtag popolari.

4.9 Query 9 - Raggio di Tweet: Trovare i Tweet entro una certa Distanza (getTweetsIDNearestRFromP)

La query 'getTweetsIDNearestRFromP' consente di visualizzare la geolocalizzazione dei tweet entro un raggio di km specificato. Nella sua implementazione, vengono passati due parametri: la posizione di riferimento P, espressa come una coppia di coordinate (latitudine, longitudine), e il raggio r, in km, entro il quale cercare i tweet. La funzione calcola la distanza geografica tra la posizione di ogni tweet e la posizione di riferimento. A tal fine viene utilizzata la formula di Haversine che permette di determinare la distanza sulla circonferenza massima tra due punti su una sfera, dati i loro valori di longitudine e latitudine.

$$\text{Distanza} = 2R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\text{lat}}{2}\right) + \cos(\text{lat1}) \cdot \cos(\text{lat2}) \cdot \sin^2\left(\frac{\Delta\text{lon}}{2}\right)}\right)$$

Dove:

R = Raggio della Terra in km

lat1 = Latitudine del primo punto in radianti

lat2 = Latitudine del secondo punto in radianti

Δlat = Differenza tra le latitudini in radianti

Δlon = Differenza tra le longitudini in radianti

Attraverso l'utilizzo di un filtro personalizzato 'haversineUDF' che implementa la formula sopra riportata, vengono selezionati solo i tweet che si trovano entro il raggio specificato rispetto alla posizione di riferimento. Successivamente, viene effettuata una selezione dei campi di interesse come l'ID del tweet, le informazioni geografiche, l'etichetta di categoria e la data.

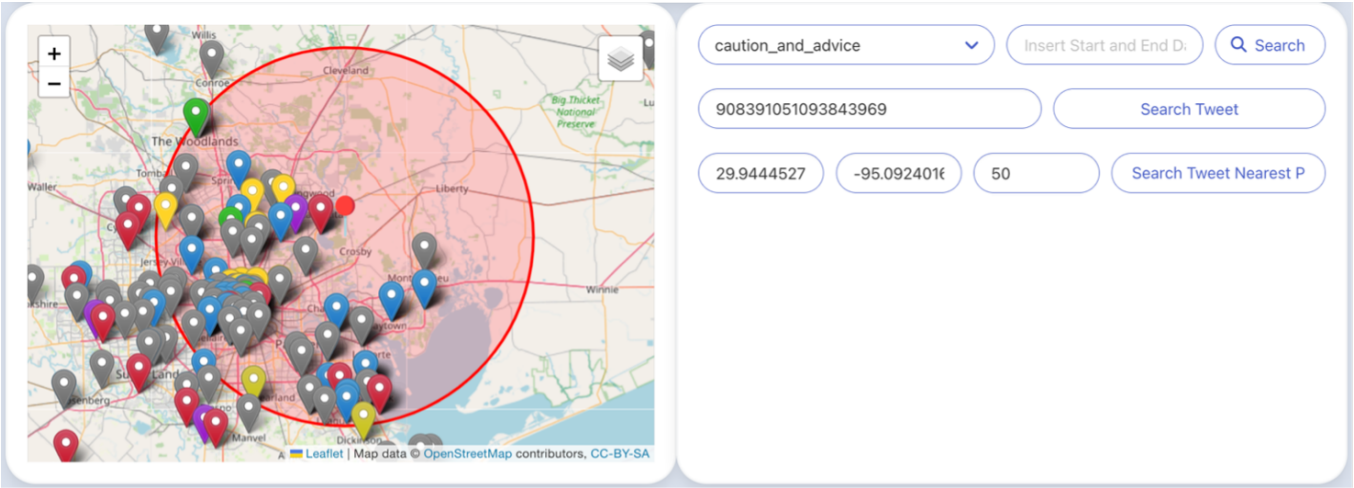


Figura 11: Query 9 - getTweetsIDNearestRFromP

La query consente di ottenere una panoramica immediata delle attività e/o degli eventi accaduti nella zona di interesse. Inoltre, l'uso della distanza geografica può contribuire all'analisi delle dinamiche territoriali e alla comprensione delle correlazioni tra la posizione geografica e il contenuto dei tweet.

5 Modello di classificazione - Naive Bayes

Per potenziare le capacità dell'applicazione, è stato integrato un modello predittivo di classificazione al suo interno. Tale modello consente di identificare il contenuto di un tweet e assegnarlo a una specifica categoria (Tabella 1). Per addestrare il modello, sono stati utilizzati due dataset (Descritti nel paragrafo 2.0.1), denominti "001" e "002", che raccolgono complessivamente 1.347.753 record categorizzati. Al fine di garantire l'accuratezza del modello, i dati sono stati separati in un set di addestramento e uno di test, consentendo di addestrare il modello sui dati di addestramento e valutarne le prestazioni sui dati di test.

La classe che implementa il modello di classificazione è PredictiveModel all'interno della quale vengono utilizzati vari componenti di Apache Spark MLlib, come trasformatori e valutatori, per l'elaborazione dei dati e la creazione del modello. Il modello viene addestrato utilizzando il metodo Naive Bayes e valutato utilizzando diverse metriche di valutazione. Infine, il modello addestrato viene salvato in un percorso specificato per essere utilizzato quando necessario.

5.1 Pre-processing

Innanzitutto, è stato necessario eseguire diverse fasi di elaborazione del testo sui dati dei tweet. Queste fasi includono la pulizia del testo, la suddivisione in parole, la rimozione delle parole comuni e la rappresentazione delle parole in forma vettoriale per l'addestramento del modello.

Si analizzano, nel dettaglio le diverse componenti utilizzate:

- **TweetTextCleaner:** La classe `TweetTextCleaner` è una classe che implementa un trasformatore personalizzato per la pulizia del testo dei tweet. Utilizza espressioni regolari per individuare e rimuovere elementi indesiderati o di poco significato come tag degli utenti, hashtag, URL e caratteri speciali.

Listing 3: 2. getNUserWithMostOfTweet

```
class TweetTextCleaner extends ... {
  val regex = ...

  override protected def createTransformFunc: String => String = {
    (text: String) => text.replaceAll(regex, "␣").toLowerCase()
  }

  override def copy(extra: ParamMap): TweetTextCleaner =
    defaultCopy(extra)
}

object TweetTextCleaner extends
    DefaultParamsReadable[TweetTextCleaner] {
  override def load(path: String): TweetTextCleaner =
    super.load(path)
}
```

In particolare si ha che "createTransformFunc" definisce la funzione di trasformazione del testo dei tweet. Questo metodo prende in input una stringa e restituisce una versione pulita del testo. La pulizia viene effettuata sostituendo con uno spazio vuoto le espressioni

regolari definite nel parametro 'regex'. Il metodo 'outputDataType' restituisce il tipo di dati dell'output della trasformazione. Il metodo 'copy' è un metodo che crea una copia dell'istanza corrente della classe 'TweetTextCleaner'. Questo metodo è utilizzato per supportare la serializzazione. L'oggetto companion 'TweetTextCleaner' è invece utilizzato per supportare la deserializzazione, infatti implementa il metodo 'load' che carica un'istanza della classe 'TweetTextCleaner' da un percorso specificato.

- **RegexTokenizer:** La classe `RegexTokenizer` viene utilizzata per suddividere il testo dei tweet pulito in parole. Utilizza un'espressione regolare (
`W+`) per individuare i separatori di parole, che corrispondono a tutti i caratteri non alfanumerici. In questo modo, il testo viene suddiviso in una sequenza di parole che costituiranno l'input per le fasi successive.

- **StopWordsRemover:** La classe `StopWordsRemover` viene utilizzata per rimuovere le parole comuni, note come "stop words", dal testo dei tweet. Queste parole, come articoli, preposizioni e congiunzioni, non apportano un contributo significativo alla comprensione del contesto e possono essere eliminate senza perdere informazioni rilevanti.

- **HashingTF:** `HashingTF` è una classe che estrae le features dal testo utilizzando la tecnica del "Term Frequency" (TF). Converte la sequenza di parole in un vettore di conteggio, in cui ogni parola viene mappata a un indice nel vettore. Questa rappresentazione vettoriale consente di rappresentare il testo in una forma numerica che può essere utilizzata come input per l'addestramento del modello di classificazione.

- **IDF:** La classe `IDF` (Inverse Document Frequency) viene utilizzata per calcolare la "Inverse Document Frequency" di un termine. Questa metrica misura l'importanza di un termine sulla base della sua frequenza inversa nell'intero corpus. L'IDF viene calcolato utilizzando un vettore di conteggio delle frequenze dei termini, come quello prodotto da `HashingTF`. L'IDF assegna pesi più alti ai termini più rari e informativi, aiutando a distinguere le parole chiave che potrebbero essere indicative della classe di appartenenza del tweet.

- **StringIndexer e IndexToString:** La classe `StringIndexer` viene utilizzata per convertire etichette testuali in valori numerici. In molti algoritmi di machine learning, le etichette devono essere rappresentate come numeri interi per poter effettuare calcoli e addestrare modelli. `StringIndexer` assegna un indice numerico univoco a ciascuna etichetta in base alla frequenza o all'ordine alfabetico. Ad esempio, se abbiamo un insieme di etichette come `"relevant_information"`, "`caution and advice`" e `"displaced and evacuations"`, `StringIndexer` assegnerà i valori 0, 1 e 2, rispettivamente.

  D'altra parte, la classe `IndexToString` svolge l'operazione inversa di `StringIndexer`. Viene utilizzata per convertire gli indici di etichette predette in etichette di testo leggibili dall'utente. Questo è particolarmente utile nell'interpretazione dei risultati dei modelli di machine learning e nella loro presentazione in un formato comprensibile agli utenti. Ad esempio, se un modello predice l'etichetta 1, `IndexToString` convertirà quel valore nella corrispondente etichetta "`caution and advice`".

  L'aggiunta di `StringIndexer` e `IndexToString` al pipeline di elaborazione dei dati è necessaria per garantire una corretta codifica delle etichette in formato numerico e una successiva decodifica per la presentazione dei risultati.

5.2 Creazione del modello

Dopo aver completato la fase di pre-processing, è stata effettuata la suddivisione del dataset in set di addestramento, validazione e test. Questa suddivisione è stata eseguita in modo random, assegnando $l'80\%$ dei dati al set di addestramento, e il restante 20% al set di test.

Successivamente, è stato creato un oggetto NaiveBayes, che implementa l'algoritmo di classificazione testuale basato sul teorema di Bayes, che si è rivelato efficace e semplice per la classificazione dei testi dei tweet.

Dopodichè, per favorire l'apprendimento delle relazioni tra le caratteristiche dei tweet e le etichette di classificazione, il modello è stato addestrato utilizzando la cross-validation. Durante questa fase, è stata adottata la metrica "F1-Score" per valutare le prestazioni del modello. Inoltre, sono stati utilizzati 5 fold, ovvero il dataset di addestramento è stato suddiviso in 5 parti, permettendo così di valutare il modello su diverse combinazioni di dati. Questa suddivisione in fold consente di ottenere una valutazione più robusta e rappresentativa delle capacità del modello nel generalizzare bene i dati e classificare correttamente i tweet.

Infine, il modello è stato valutato utilizzando il test set. È importante sottolineare che questi dati di test sono indipendenti da quelli utilizzati durante l'addestramento del modello. Nello specifico, sono state calcolate diverse metriche di valutazione, tra cui l'accuratezza, la precisione, il recupero (recall) e l'F1 score, che hanno fornito indicazioni sulle prestazioni del modello e sulla sua capacità di classificare correttamente i tweet.

Metrica	**Valore**
Test Accuracy	0.7857
Test Precision	0.8188
Test Recall	0.7857
Test F1 Score	0.7943