Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

TWEET TRACKER: ANALISI E VISUALIZZAZIONE DI TWEET GEOLOCALIZZATI

Relatore:
Chiar.mo Prof.
Paolo Ciancarini

Presentata da:
Giacomo Puligheddu

Sessione II
2019/2020

Indice

Elenco delle figure

Capitolo 1

Introduzione

1.1 Motivazioni della tesi

La quantità di dati che circola su Internet aumenta di anno in anno e, in particolare, aumenta sempre di più l'utenza sui social network. Si stima che a Luglio del 2020 su Internet vi siano stati circa 4,57 miliardi di utenti attivi, 3,96 miliardi di questi erano attivi sui social [1]. Entrando a far parte della vita quotidiana della maggioranza della popolazione mondiale, è evidente ormai da tempo che i social sono i luoghi in cui si concentra la maggiore quantità di dati condivisi dalle persone. Quest'anno Facebook conta 2,7 miliardi di utenti attivi mensilmente [2] e Twitter nel 2019 ne contava 330 milioni [3], per non parlare di YouTube, Instagram, Snapchat e altri. È ormai la norma condividere esperienze, opinioni, pensieri e persino la propria posizione tramite questi srumenti; pertanto, tramite un'accurata analisi, dev'essere possibile ricavare informazioni significative dalla raccolta di dati provenienti dai social. Le persone che vivono un'emergenza e la comunicano attraverso queste piattaforme, possono essere considerate dei sensori umani sul territorio: le informazioni generate, se aggregate in tempo reale, rappresentano una serie di conoscenze utili proprio per gestire l'emergenza stessa. Oltre alle emergenze, tramite i social è

possibile ricostruire informazioni di eventi (concerti, fiere, manifestazioni, etc...) in modo da ottenere il punto di vista collettivo su di essi; ad esempio si potrebbe effettuare un'analisi sui contenuti pubblicati dagli utenti nei giorni precedenti a delle elezioni. Tra tutti i social network che hanno anche contenuti interamente testuali (cioè che possono non contenere immagini, video o altri elementi multimediali) i più popolari sono Facebook e Twitter. I motivi della scelta di quest'ultimo sono costituiti da due ragioni principali:

1. I *tweet*[1] sono limitati a una lunghezza di 280 caratteri (140 fino al 2017) e questo comporta che gli utenti dovranno condividere le proprie informazioni in modo conciso, utilizzando parole significative [4].

2. Le API di Twitter permettono, contrariamente a quelle di Facebook, di raccogliere tweet in tempo reale provenienti da qualsiasi account pubblico.

Questa tesi descrive progettazione e sviluppo di un programma, Tweet Tracker, che consente di raccogliere tweet in tempo reale, mostrarli su una mappa (se geolocalizzati) e fornire grafici relativi ad essi. Sarà inoltre possibile salvare (caricare) i tweet in (da) un file in formato JSON. Lo scopo di questo progetto è quello di testare e mostrare le potenzialità degli strumenti messi a disposizione da Twitter per la raccolta di dati dalla sua piattaforma.

[1] "Tweet" è il nome con cui vengono chiamati i messaggi di testo pubblicati dagli utenti su twitter.

1.2 Struttura della tesi

La tesi è così strutturata:

- in questo capitolo è presente un' introduzione al progetto e ai motivi della sua realizzazione;
- nel secondo capitolo viene introdotto Twitter, vengono spiegate le sue funzionalità e viene descritta la struttura delle API;
- nel terzo capitolo viene spiegato il funzionamento del programma e il motivo di alcune scelte implementative;
- nel quarto capitolo vengono esposte le conclusioni e si trattano alcuni spunti per lavori futuri.

Il codice sorgente è disponibile al link "*https://github.com/Zacomo/Tweet-Tracker*".

Capitolo 2

Twitter

2.1 La nascita della piattaforma

Twitter è un *social network* sulla quale vengono pubblicate notizie nella forma di *microblogging*[1] ed è attualmente proprietà della società **Twitter, Inc.** con sede a San Francisco (Stati Uniti). È stato creato da Jack Dorsey nel 2006 e con il tempo ha riscosso un grande successo in tutto il mondo: attualmente conta 330 milioni di utenti attivi mensilmente, 145 milioni giornalmente [5]. Ciò che distingue Twitter dagli altri social è la brevità dei messaggi che è possibile pubblicare: parliamo soltanto di 280 caratteri (in origine 140). Probabilmente il successo della piattaforma è dovuto anche a quest'ultima particolare caratteristica perché, data la sua brevità, un *tweet* risulta più immediato rispetto ai post presenti su altri social. L'idea di base era quella degli SMS ed effettivamente, per la sua lunghezza, chi pubblica un tweet è come che stia mandando un SMS a tutte le persone che lo seguono [6]. Nel 2009, Twitter ha riscosso un grande successo in

[1]Si tratta di una forma di pubblicazione costante di piccoli contenuti in Rete, sotto forma di brevi messaggi di testo, immagini, video, audio MP3, ma anche segnalibri, citazioni, appunti. Questi contenuti vengono pubblicati in un servizio di rete sociale, visibili a tutti o soltanto alle persone della propria comunità.

quanto è stato utilizzato come uno dei mezzi principali per seguire i disordini delle proteste elettorali in Iran [7]. Tuttavia, la piattaforma da questo evento non ha semplicemente trovato un maggior numero di utenti ma si è anche affermata nel mondo della politica: oggi viene utilizzata da politici ed istituzioni in tutto il mondo, i quali hanno intuito il suo potenziale comunicativo [8].

2.2 Le API di Twitter

2.2.1 API in generale

Per API (Application Programming Interface) si intende un insieme di funzionalità appartenenti a un programma che vengono messe a disposizione di altri sviluppatori. Le API consentono di sfruttare gli strumenti del servizio che le offre per ottenere nuove funzionalità. Twitter fornisce principalmente due tipi di API: API REST[2] e STREAMING API. Attualmente sono presenti tre livelli [9] di accesso ai dati di Twitter. Ogni livello offre accesso a maggiori quantità di dati rispetto a quello precedente, oltre ad includere funzionalità aggiuntive; in ordine di numero di funzionalità crescente, i tre livelli sono:

- Standard;
- Premium (a pagamento);
- Enterprise (a pagamento).

Il livello delle API in questo progetto è quello Standard.
Attualmente le API di Twitter sono alla versione 1.1 e sono quelle utilizzate nel lavoro descritto da questa tesi, in quanto erano le uniche disponibili nel momento in cui è iniziato lo sviluppo. Tuttavia, esiste una versione

[2]Representational State Transfer, si tratta di un'architettura che descrive un insieme di regole per realizzare API che utilizzano correttamente i metodi HTTP.

2.0 (attualmente ancora in via di sviluppo) che sostituirà la 1.1 e alcuni suoi metodi sono già disponibili all'uso [10]. La nuova versione promette di migliorare l'esperienza per gli sviluppatori aggiungendo le seguenti funzionalità:

- La possibilità di specificare quali campi ricevere, ad esempio è possibile ottenere in risposta solo il testo e la data di un tweet.
- Nuovi dati relativi ai tweet, tra cui account menzionati, tweet citati e posti associati al tweet.
- Metrics (statistiche relative al tweet come visualizzazioni, interazioni, etc...) più approfonditi
- La possibilità di filtrare i tweet tramite gli operatori *entity* e *context*
  - Si tratta di annotazioni che indicano il contenuto dei tweet. Le *entity* sono assegnate in base a ciò che viene menzionato esplicitamente nel tweet; le *context annotations* sono dedotte in seguito all'analisi del testo del tweet.
- Un miglioramento nel tracciamento dei tweet che fanno parte di una stessa conversazione; in particolare, i tweet che appartengono alla stessa conversazione avranno un campo chiamato "*conversation id*"

Considerando le nuove funzionalità appena citate e il fatto che le API 2.0 sostituiranno le API 1.1, è consigliabile partecipare all'accesso anticipato della nuova versione per familiarizzare con le novità e sfruttarne i vantaggi. Per quanto riguarda la localizzazione dei tweet, è importante evidenziare che Twitter ha rimosso nel 2019 la possibilità per gli utenti di localizzarsi con precisione. Ad oggi è possibile solamente scegliere un luogo vicino (un locale, un monumento, una piazza, etc...). Tuttavia è possibile abilitare la geolocalizzazione precisa per i tweet contenenti foto scattate tramite la fotocamera di Twitter (via smartphone) [11] [12].

2.2.2 Account sviluppatore

L'accesso alla lettura e alla scrittura di dati su Twitter è basato tramite identificazione OAuth[3] [13]. Per utilizzare le API è necessario avere un account Twitter normale tramite la quale inviare una richiesta per ottenere un account da sviluppatore. In particolare occorre compilare un modulo sul sito "*developer.twitter.com*" in cui viene richiesto di specificare:

- Il profilo twitter a cui si intende associare l'account da sviluppatore
- Per conto di chi si sta facendo richiesta di tale account (ovvero se per uso personale o per l'uso di un' organizzazione)
- Nazione da cui si opera
- Motivo della richiesta, ovvero se per uso accademico, pubblicitario, creazione di bot etc...
- Una breve descrizione del progetto che si intende realizzare

Una volta inviato, il modulo viene valutato e, se rispetta i termini di servizio di Twitter, il richiedente riceverà una conferma via mail tramite la quale potrà attivare l'account da sviluppatore. A questo punto, dopo aver ottenuto il suddetto account, è necessario recarsi nuovamente su "*developer.twitter.com*" e creare un'applicazione: a questa app Twitter associa dei token e delle chiavi che dovranno poi essere utilizzati dallo sviluppatore per poter usufruire delle API:

- *consumer key*
- *consumer secret*
- *access token*

[3]Protocollo che si basa sull'emissione di un token di accesso da parte di un server autorizzativo.

- *access token secret*

Le chiavi ovviamente sono personali e legate all'account utilizzato, quindi vanno mantenute segrete; nel caso in cui vengano erroneamente condivise o pubblicate è bene rigenerarle in quanto forniscono l'accesso all' account personale.

2.2.3 API REST

Le API sono HTTP-based: la maggior parte dei metodi che interrogano Twitter utilizzano una richiesta GET e i dati vengono restituiti in formato JSON[4]; i metodi che inseriscono, modificano o eliminano dati utilizzano una richiesta POST o DELETE. Le richieste non devono superare delle determinate soglie limite per ogni endpoint. Tali soglie sono descritte nella documentazione ufficiale delle API v1 ("Standard API v1.1 rate limits per window") [14]. Per evitare il superamento delle soglie limite, è possibile prendere alcuni provvedimenti tra cui:

- Se è necessario utilizzare spesso le stesse informazioni, è bene memorizzarle con un sistema di caching anziché interrogare continuamente le API.
- Se il servizio realizzato è utilizzato da molti utenti è meglio richiedere i dati solo per gli utenti che hanno fatto l'accesso di recente.
- Se si stanno monitorando diversi termini tramite query, si possono cercare meno frequentemente i termini che producono meno risultati (o alternativamente si possono usare le API Streaming).

Un'applicazione che supera ripetutamente le soglie limite verrà inserita in una black-list: le applicazioni in questa lista non potranno più utilizzare le API di Twitter.

[4]JavaScript Object Notation, si tratta di un formato di rappresentazione di dati.

2.2.4 Search API

Questa API è la più importante tra quelle REST e permette di effettuare delle query tra i tweet attualmente più popolari; restituisce risultati simili alla barra di ricerca presente sul sito di Twitter e sulle rispettive app per smartphone. Nella sua versione standard (ovvero gratuita) ricerca tra tweet non più vecchi di sette giorni. La search si basa sulla rilevanza dei contenuti e non sulla completezza; questo significa che non verranno restituiti tutti i risultati corrispondenti ai parametri di ricerca ma solo quelli più rilevanti [15]. Esistono diversi parametri [16] su cui basare la ricerca:

- Parametro "**q**": stringa di massimo 500 caratteri (operatori AND e OR compresi) che rappresenta il contenuto ricercato nei tweet. Utilizzando caratteri particolari è possibile affinare la ricerca:
  - inserendo più parole all'interno di doppi apici ("") è possibile ricercare esattamente quella frase.
  - utilizzare il segno "-" permette di escludere dalla ricerca i tweet contenenti la parola che lo segue.
  - utilizzare "from:" e "at:" cercano, rispettivamente, tweet pubblicati dall'account che segue "from:" (es: from:NASA restituisce i tweet pubblicati dal profilo NASA) o tweet pubblicati in risposta all'account che segue "at:" (es: at:NASA).
  - until:dd-mm-yyyy e since:dd-mm-yyyy trovano, rispettivamente, i tweet prima e dopo la data indicata.
- Parametro "**geocode**": seleziona i tweet localizzati all'interno di un dato raggio (occorre fornire altitudine e latitudine di un punto assieme al raggio entro la quale si vuole effettuare la ricerca). Se il tweet è geolocalizzato allora utilizza il parametro del tweet che ne indica la posizione, altrimenti si utilizza la posizione geografica legata all'account di chi ha pubblicato il tweet.

- Parametro "**lang**": seleziona solo i tweet del linguaggio desiderato. occorre fornire un codice ISO 639-1[5].

- Parametro "**result_type**": specifica il tipo di risultati che si *preferirebbe* ricevere. È possibile scegliere tra: *mixed* (sia tweet popolari che tweet recenti), *recent* (solo i tweet più recenti), *popular* (solo i tweet più popolari).

- Parametro "**count**": dato un numero intero, restituisce tanti tweet quanti indicati. Il valore di default è 15 e il massimo è 100.

2.2.5 Streaming API

A differenza della search API (che cerca tra tweet già pubblicati), con la streaming API si mette in ascolto per la pubblicazione di nuovi tweet in tempo reale. La differenza dalle API REST sta nel fatto che queste ultime processano ogni richiesta in modo indipendente, mentre lo streaming richiede una connessione HTTP persistente e non deve essere chiusa immediatamente. Dopo aver impostato una serie di parametri, questa API riceve da Twitter tutti i tweet che rispettano i parametri scelti, a patto che la ricerca non sia troppo ampia: in tal caso il flusso di tweet sarebbe troppo grande e molti di questi andrebbero persi [17]. Tramite i parametri è possibile filtrare i tweet in base a:

- Follow: occorre fornire una lista di ID appartenenti a degli utenti e verranno selezionati solo i tweet pubblicati dagli utenti in lista. Verranno restituiti:

  - i tweet creati dall'utente;
  - i "retweet" dell'utente;

[5]Si tratta di una nomenclatura standard utilizzata per classificare i linguaggi. Ogni codice indica un linguaggio ed è composto da due lettere (es: it per l'italiano).

  - i tweet in risposta a quello creato dall'utente.

  Non sono compresi i tweet in cui l'utente viene menzionato e nemmeno i tweet appartenenti ad utenti che hanno il profilo "privato".

- Contenuto: per questo filtro è necessario dare in input una lista di stringhe separate da virgole che verranno cercate nel contenuto dei tweet. Le stringhe vengono cercate così come sono, quindi si tratta di un filtro case sensitive.

- Localizzazione: per questo filtro occorre fornire una *bounding box*, ovvero le coordinate (longitudine, latitudine) per due punti che andranno ad indicare gli angoli opposti di un rettangolo; verranno restituiti tutti i tweet localizzati all'interno dell'area del suddetto rettangolo.

- Lingua: è possibile scegliere di ricevere solo i tweet in un linguaggio in particolare.

Capitolo 3

Tweet Tracker

3.1 Introduzione

Tweet Tracker è un programma realizzato da zero per questo progetto, sfruttando strumenti e librerie descritti in seguito. Il suo scopo è sfruttare le funzioni offerte da Twitter per tracciare tweet geolocalizzati e dai contenuti interessanti. In particolare è possibile trovare o mettersi in ascolto per dei tweet provenienti da una determinata area, scritti in una determinata lingua e contenenti determinate parole chiave. Questo rende possibile ricercare tweet geolocalizzati correlati ad un evento particolare, che può essere di tipo sportivo, mediatico o persino un'emergenza di competenza della protezione civile (es. allerte meteo, alluvioni, sismi). I tweet, se geolocalizzati, forniscono delle informazioni significative direttamente dal luogo interessato e costituiscono un'importante risorsa.
Il programma si presenta all'utente con un'interfaccia tramite la quale è possibile compiere diverse operazioni. In tutto sono presenti tre finestre, ciascuna gestita dal controller corrispondente. La finestra primaria 3.2 (pag. 28) è quella che si presenta all'avvio ed è anche la più ricca in termine di contenuti e funzionalità; le altre due contengono la mappa e i grafici. I controller gestiscono la presentazione delle finestre e parte della logica

per elaborare i dati. Questa struttura semplice è dovuta alla dimensione non eccessiva del progetto e alla sua natura sperimentale. Per espandere le funzionalità del programma, la prima modifica da effettuare è quella di introdurre una classe Model che gestisca la logica per i dati e lasci ai controller il solo compito di implementare la logica di presentazione. Tale modifica comporterebbe un corretto utilizzo del pattern MVC[1], facilitando l'aggiunta di nuove funzionalità. Infine, le funzioni che gestiscono le chiamate per le API di Twitter, sono contenute in un'unica classe, richiamata poi dal controller principale.

3.2 Casi d'uso

In questo paragrafo vengono presentati i casi d'uso del programma, in modo da illustrare chiaramente il suo funzionamento.

3.2.1 Raccolta di informazioni su degli avvenimenti

L'utente è interessato a raccogliere dei dati geolocalizzati su un avvenimento in particolare, ad esempio sulle elezioni regionali.

- All'interno della barra di ricerca della "Stream Search", scrive una lista di parole separate da una virgola e seleziona la checkbox "localizzati" (es. "elezioni,regionali,voti"); la ricerca non tiene conto dell'uso di maiuscole o minuscole. In seguito clicca sul pulsante "Stream Search".

- A questo punto il programma manderà una richiesta a Twitter per utilizzare l'API Stream. In particolare viene aperta una connessione HTTP persistente, tramite la quale il server inoltra tweet localizzati in Italia.

[1]Model-View-Controller, è un pattern architetturale che separa la logica di presentazione dalla logica di business.

- Il programma filtra i tweet ricevuti e mostra, nella lista della schermata principale, solo i tweet contenenti le parole chiave cercate; gli altri risultati non vengono memorizzati.

- Quando l'utente ritiene di aver collezionato un numero di tweet sufficiente, può cliccare sul pulsante "Stream Stop". A questo punto può consultare:

  - Il contenuto dei tweet tramite la lista nella parte a sinistra.
  - La loro posizione: aprendo la mappa tramite un click sul pulsante "Mappa" viene aperta una finestra in cui sono presenti dei segnali (*marker*) che indicano la posizione dei tweet. Se l'utente posiziona il puntatore sopra un marker, viene mostrato il testo del tweet corrispondente.
  - Le statistiche di un tweet: cliccando su un tweet in lista, nella tabella a destra vengono mostrate alcune statistiche, in particolare numero di *like*, *retweet* e *follower* (dell'utente).
  - La cloud word per quel gruppo di tweet, dove vengono rappresentate le parole contenute nei testi con una dimensione correlata alla frequenza con cui compaiono. Cliccando su una di queste parole, viene generata una nuova cloud word relativa ad essa (ovvero composta da termini utilizzati assieme alla parola cliccata).

- Infine l'utente può scegliere di salvare la lista di tweet in un file JSON tramite l'apposito pulsante, in modo da poterli consultare nuovamente in futuro.

3.2.2 Analisi di tweet raccolti in precedenza

L'utente intende analizzare nel dettaglio una lista di tweet salvata in precedenza. Il formato accettato dal programma è quello di un file JSON

che consiste in un array di oggetti JSON (ovvero dei tweet).

- L'utente clicca sul pulsante "Carica JSON" e, tramite un selettore di file, seleziona il file contenente i tweet che intende analizzare.
- Dopo aver selezionato il file, tutti i tweet al suo interno vengono caricati nella lista, viene generata la cloud word corrispondente ed è possibile consultare statistiche, grafici e mappa.
- I tweet chiaramente potrebbero non essere recenti, di conseguenza le statistiche vengono recuperate con una nuova richiesta a Twitter.

3.2.3 Ricerca dei tweet più popolari

L'utente intende capire quali sono gli argomenti più discussi negli ultimi giorni, correlati a determinate parole.

- All'interno della barra di ricerca della "Popular Search", l'utente scrive una stringa di ricerca (maiuscole e minuscole non contano ma gli spazi sì). La stringa può contenere degli operatori che aiutano ad affinare la ricerca. Tali operatori sono descritti nella finestra 3.3 (pag. 29) che si apre al clic del pulsante "?", in questo modo l'utente può controllare il loro funzionamento. Una volta inserita la stringa, l'utente clicca il pulsante "Popular Search".
- Dopo la pressione sul pulsante per la ricerca, il programma manda una richiesta GET a Twitter con la stringa e i parametri inseriti. La risposta conterrà massimo cento tweet (limite imposto dall'uso dell'API standard) e verranno immediatamente mostrati sulla lista della finestra principale.
- Come per la "Stream Search", è possibile vedere le statistiche e la cloud word, oltre a poter salvare i tweet in JSON. Anche la mappa è consultabile, ma raramente sono presenti dei marker in quanto, su cento tweet, al massimo solo un paio sono geolocalizzati.

3.3 Strumenti utilizzati

Dato che la scelta del linguaggio non influisce su quelli che sono gli obiettivi del progetto, la scelta è ricaduta su Java per una semplice preferenza personale. Per quanto riguarda gli altri strumenti, sono stati utilizzati: Git e GitLab per il versioning, JavaFX 15 per la realizzazione dell'interfaccia grafica, Twitter4J per l'utilizzo delle API di Twitter, Google Maps per la segnalazione dei tweet sulla mappa, AnyChart per la realizzazione di word cloud, Gson per convertire i tweet in JSON.

3.3.1 JavaFX

JavaFX [18] è una famiglia di software applicativi, basati sulla piattaforma Java che consentono di realizzare applicazioni web e desktop. Un software realizzato con JavaFX contiene uno o più "*Stage*" che corrispondono a finestre; ogni Stage è collegato a una "*Scene*" e ogni Scene contiene un oggetto "*Scene Graph*" (vedere la figura 3.1 a pagina 22.

- **Stage**: lo Stage è il contenitore più esterno dell'applicazione e corrisponde a una finestra. Un programma che utilizza JavaFX può avere molteplici finestre e ciascuna ha uno Stage corrispondente.
- **Scene**: ogni Stage può mostrare una Scene per volta ma è possibile scambiare le Scene a runtime.
- **Scene Graph**: contiene tutti i componenti visibili (controlli, layout, menu, grafici,...) e dev'essere collegato a una Scene.

Tutti i componenti collegati allo Scene Graph sono chiamati nodi; alcuni nodi possono avere nodi figli, altri no. I nodi sono suddivisi per categorie:

- Nodi di controllo: sono componenti che permettono un'interazione da parte dell'utente. Parliamo di elementi tipo pulsanti, checkbox, caselle di testo e così via.

- Layout: sono nodi che possono contenerne altri. Vengono utilizzati per gestire la presentazione della finestra e la posizione dei componenti al suo interno. Componenti di questo tipo sono: HBox (letteralmente "scatola orizzontale", permette di disporre i nodi figli in orizzontale), VBox (come HBox ma in verticale), GridPane (disposizione dei nodi figli secondo una griglia),...

- Grafici: sono componenti già pronti per implementare dei grafici all'interno dell'applicazione. Alcuni nodi di questo tipo sono: PieChart (grafico a torta), BarChart (grafico a barre),...

- WebView: si tratta di un componente in grado di mostrare pagine web.

Per la realizzazione dell'interfaccia grafica di questo progetto, è stato utilizzato "*Scene Builder*". Si tratta di uno strumento che consente la realizzazione di interfacce in modo visivo, aggiungendo elementi semplicemente tramite "*drag and drop*". Tali componenti possono poi essere modificati anche tramite il codice, in quanto Scene Builder genera comunque un file FXML[2]. I componenti utilizzati per l'interfaccia di Tweet Tracker sono i seguenti:

- *AnchorPane*: è un contenitore per altri nodi. La particolarità di questo elemento è che permette di ancorare i nodi figli ai propri bordi, secondo un margine regolabile.

- *HorizontalSplitPane e VerticalSplitPane*: container di nodi che dividono lo spazio disponibile in due (il primo in orizzontale, il secondo in verticale). Il divisore delle due aree create da uno SplitPane è regolabile dall'utente che può quindi decidere di dare più spazio ad un'area a discapito dell'altra.

[2]Linguaggio di markup basato su XML utilizzato da JavaFX per rappresentare le interfacce di una scena.

- *TextField*: un semplice campo di testo.
- *Button*: un pulsante alla cui pressione può essere associata un'azione.
- *ListView*: una lista che permette di mostrare delle scelte all'utente. Al click su una scelta può venire associata un'azione.
- *TableView*: una tabella.
- *WebView*: si tratta di un componente in grado di mostrare pagine web. Si tratta di una sorta di browser a cui può essere chiesto di mostrare sia una pagina online (fornendogli l'indirizzo), sia una pagina HTML creata dall'utente.

3.3.2 Twitter4J

Twitter4J [19] è una libreria Java non ufficiale per gli API di Twitter. Permette di integrare comodamente i servizi di Twitter in un programma Java. Prima di tutto, per usufruire di questa libreria, occorre scaricarla (dal sito ufficiale *twitter4j.org*) e aggiungere i relativi file jar alle librerie del progetto; alternativamente è possibile utilizzare Maven[3] e modificare il file pom.xml come indicato nel sito ufficiale di Twitter4J. Prima di utilizzare qualsivoglia funzione, è necessario configurare Twitter4J inserendo i token per l'utilizzo delle API di Twitter. La configurazione è possibile:

- Tramite la creazione di un file chiamato "twitter4j.properties", da compilare secondo le istruzioni ufficiali.
- A livello di codice, utilizzando l'oggetto "*"ConfigurationBuilder*".
- Tramite le proprietà di sistema o tramite variabili d'ambiente.

[3]Maven è uno strumento per la gestione di progetti che si basa sul file pom.xml (Project Object Model), al cui interno vengono indicate le dipendenze del progetto.

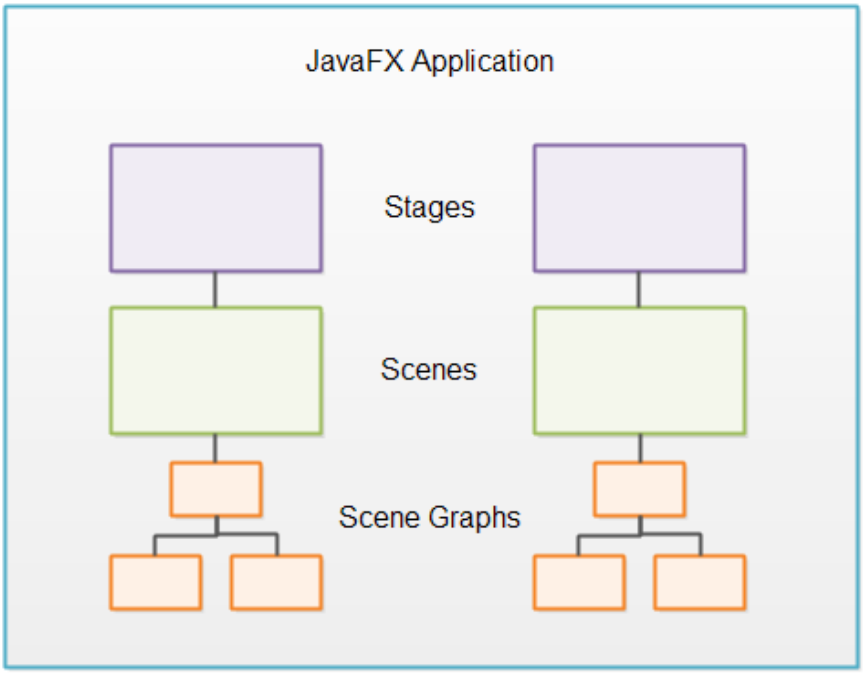

Figura 3.1: Struttura di un'applicazione JavaFX

Una volta configurata la libreria, è possibile effettuare diverse azioni, tra cui:

- Pubblicare un tweet

```
Twitter twitter = TwitterFactory.getSingleton();
    Status status = twitter.updateStatus(latestStatus);
    System.out.println("Successfully updated the status to [" +
                        status.getText() + "].");
```

- Ricevere una lista dei tweet della propria timeline

```
Twitter twitter = TwitterFactory.getSingleton();
    List<Status> statuses = twitter.getHomeTimeline();
    System.out.println("Showing home timeline.");
    for (Status status : statuses) {
        System.out.println(status.getUser().getName() +
        ":" +status.getText());
    }
```

- Mandare o ricevere messaggi privati

```
Twitter sender = TwitterFactory.getSingleton();
    DirectMessage message = sender.sendDirectMessage(recipientId,
                                                     message);
    System.out.println("Sent: " message.getText() +
                       " to @" + message.getRecipientScreenName());
```

- Cercare tweet tramite l'API Search. Esempi a pag. 38
- Rimanere in ascolto per dei tweet tramite l'API Stream. Esempi a pag. 38

3.3.3 Google Maps

Google offre diversi servizi e per molti di questi sono disponibili delle API che consentono di integrare delle interessanti funzionalità nelle proprie applicazioni. Tra i servizi offerti, sono presenti "*Translate*", "*Gmail*" e "*Google Maps*". Per l'utilizzo di ognuno di questi servizi è necessaria l'autenticazione tramite il protocollo *Oauth 2.0*; le chiavi per il protocollo vengono fornite dalla "*Developer Console*" sul sito ufficiale di Google per le API (*developers.google.com*). Lo strumento utilizzato in questo progetto è Google Maps [20]. La piattaforma di Google Maps offre tre servizi specifici, ciascuno con la sua API:

- Maps: è il servizio che si occupa di fornire effettivamente una rappresentazione della mappa, di inserirvi dei segnali (Marker) e di spostarsi su di essa.

- Routes: è il servizio che permette a un utente di trovare il miglior modo per spostarsi da un luogo all'altro, calcolando il percorso più breve tenendo conto anche del traffico in tempo reale.

- Places: è il servizio che offre dati dettagliati su oltre 150 milioni di luoghi; gestisce le informazioni sui luoghi, come indirizzi completi, numeri di telefono e altre indicazioni sulla loro struttura.

Tra questi tre servizi, l'unico utilizzato nel progetto è Maps, in particolare con le API JavaScript. Maps permette di rappresentare una mappa e di personalizzarla con i propri contenuti. Può mostrare quattro tipi di mappe ("*roadmap*", "*satellite*", "*hybrid*" e "*terrain*") modificabili aggiungendo stili, controlli ed eventi. Affinché la mappa possa essere mostrata, occorre crearle un "*container*" nel file HTML (generalmente si utilizza un *div*) e darle un id o comunque un modo per identificarla all'interno del DOM[4]. Ovviamente è possibile regolare la dimensione del container della mappa tramite CSS[5]. La classe che rappresenta l'oggetto della mappa è "*Map*"; un singolo oggetto di questa classe rappresenta una mappa nella pagina (è possibile creare più istanze). Il costruttore richiede due parametri: un identificatore per l'elemento che contiene la mappa e un oggetto "*Options*". Ogni oggetto options contiene sempre le due opzioni "*center*" e "*zoom*". Center richiede di fornire le coordinate di un punto che sarà il centro della mappa al momento della prima visualizzazione; zoom invece regola appunto il livello di zoom della mappa, con 0 che rappresenta il punto di vista più lontano possibile.

[4]Document Object Model, è uno standard del W3C che consiste in una forma di rappresentazione dei documenti strutturati come modello orientato agli oggetti.

[5]Cascading Style Sheets, linguaggio utilizzato per la formattazione di documenti HTML, XHTML e XML.

3.3.4 AnyChart

AnyChart [21] è una libreria JavaScript che permette l'integrazione di diversi tipi di grafici all'interno di un'applicazione, indipendentemente dal fatto che sia desktop, mobile o web. Il codice è pubblico e visibile su Github[6]. Nella libreria sono compresi sia grafici più comuni (es. grafico a torta, a barre, a linee,...) sia grafici più particolari, come ad esempio il "*tag cloud*". In questo progetto è stato fatto utilizzo della libreria AnyChart proprio per realizzare quest'ultimo tipo di grafico. Un tag cloud o *word cloud* (nuvola di parole), è una rappresentazione di dati provenienti da un testo. In particolare si tratta di un'immagine composta da parole provenienti dal suddetto testo, in cui la dimensione di ogni parola indica la frequenza con la quale questa compare (maggiore la frequenza, maggiore la dimensione della parola). I dati da inserire nella cloud word possono essere passati al costruttore (*anychart.tagCloud()*) o al metodo *data()*. I dati possono essere passati o come lista di parole con le relative frequenze o come testo.

- Tramite costruttore: si crea una variabile che contenga i dati e si passano come parametro al costruttore.

```
var data = [
    {"x": "learning", value: 80},
    {"x": "includes", value: 56},
    {"x": "lists", value: 44}
    ];
    chart = anychart.tagCloud(data);
```

- Tramite metodo: è utile quando si vuole passare solamente il testo da rappresentare, facendo in modo che sia la libreria a calcolare la

[6]Servizio di hosting per progetti software. Consente agli sviluppatori di caricare il proprio codice online per permettere ad altri utenti di scaricarlo e/o modificarlo.

frequenza delle parole. In questo modo è possibile impostare delle operazioni per il parsing del testo, tra cui un numero massimo di elementi, una lunghezza minima e/o massima per le parole, una lista di parole da escludere dal conteggio e soprattutto il parametro mode. Quest'ultimo parametro ha due valori: "byWord" che serve per dividere il testo in parole e "byChar" che divide il testo in caratteri.

```
var data = "Tyger, tyger, burning bright " +
           "In the forests of the night, " +
           "What immortal hand or eye " +
           "Could frame thy fearful symmetry? ";
chart.data(data, {
        mode: "byWord",
        maxItems: 16,
        ignoreItems: [
                       "the",
                       "and",
                       "he",
                       "or",
                       "of",
                       "in",
                       "thy"
                      ]
});
```

3.4 Classi e componenti

In questo paragrafo viene spiegato il funzionamento dei componenti del programma; vengono dunque illustrate alcune classi e alcuni metodi importanti, accompagnati dal corrispondente snippet di codice. Tale snippet di codice sarà opportunamente semplificato per questioni di leggibilità.

3.4.1 Interfaccia

La finestra principale (fig. 3.2, pag. 28), così come le altre, è realizzata tramite Scene Builder ed è contenuta in un file FXML. Il controller di tale file è il controller primario ("*MainController*"). L'interfaccia si compone di un *AnchorPane* che contiene uno *SplitPane* orizzontale:

- La parte superiore contiene
  - Due barre di ricerca (una per la ricerca per popolarità e un'altra per la ricerca stream).
  - Due pulsanti corrispondenti ai due tipi di ricerca, più un pulsante ("Stream Stop") per far terminare la Stream Search.
  - Un pulsante "?" che apre una finestra con all'interno gli operatori utilizzabili nella ricerca per popolarità (fig. 3.3, pag. 29).
  - Un pulsante per aprire la finestra della mappa.
  - Un pulsante per aprire la finestra dei grafici.
- La parte inferiore contiene uno SplitPane verticale in cui troviamo
  - Una *ListView* in cui compariranno i tweet trovati o caricati
  - Una *TableView* in cui compaiono le statistiche dei tweet su cui l'utente clicca
  - Una *WebView* in cui compare la cloud word relativa ai tweet trovati o caricati.

3.4.2 Controller primario

Il controller primario gestisce la presentazione e il funzionamento degli elementi della schermata principale. Come ogni controller, implementa

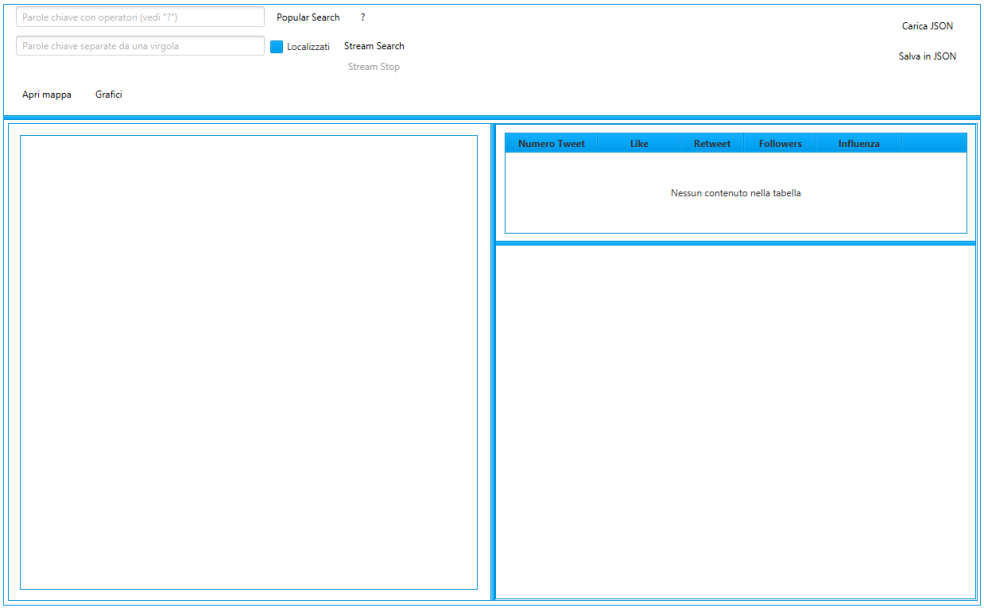

Figura 3.2: Finestra principale

l'interfaccia *Initializable* e pertanto deve contenere un *override* del metodo *initialize()*. All'interno di questo metodo sono presenti semplici operazioni di inizializzazione degli elementi della finestra principale. Gli oggetti dichiarati in questa classe (esclusi quelli legati al file FXML) sono:

- *tweetHandler*: istanza della classe *TweetHandler* descritta nella sezione che tratta la gestione delle richieste di tweet (3.4.5 a pag. 38).

- *positions*: Lista di oggetti *Position* utilizzata per passare le posizioni dei tweet al controller della mappa (3.4.4 pag. 36). Ogni oggetto *Position* è composto da: due numeri Double (che conterranno latitudine e longitudine del tweet) e da una stringa (che conterrà il testo del tweet).

Operatore	Trova tweet...
watching now	contenenti sia "watching" che "now".
"happy hour"	contenenti esattamente la frase "happy hour".
beer -root	contenenti "beer" ma non "root".
#haiku	contenenti l'hashtag "haiku".
from:interior	mandati dall'account "interior".
list:NASA/astro...	mandati da un account nella lista NASA "astronauts-in-space-now".
to:NASA	mandati in risposta all'account "NASA".
@NASA	che menzionano l'account "@NASA".
filter:safe	senza contenuti potenzialmente sensibili.
-filter:retweets	che non siano retweet.
filter:links	contenenti un link.
url:amazon	contenenti un url con la parola "amazon" al suo interno.
since:2015-12-21	mandati dalla data "2015-12-21" (yyyy-mm-dd) in poi.
until:2015-12-21	mandati prima della data "2015-12-21" (yyyy-mm-dd).
movie -scary :)	contenenti "movie" ma non "scary" e con un tono positivo.
flight :(	contenenti "flight" e con un tono negativo.
traffic ?	contenenti "traffic" e posti come domanda.

Figura 3.3: Finestra che descrive gli operatori e il loro funzionamento.

- *jsonTweetList*: oggetto del tipo *JsonArray*, contiene la lista dei tweet sulla quale il programma sta lavorando al momento.
- *tweetsObsList*: si tratta di una *ObservableList* di stringhe che contiene il testo da visualizzare nella ListView.
- *gson*: istanza di *Gson*, serve per utilizzare le funzionalità dell'omonima libreria.

I metodi più rilevanti inseriti in questa classe sono:

- *PopularSearch*
  Questo metodo gestisce il click sul pulsante "Popular Search". Prima

di tutto recupera il testo inserito nella barra di ricerca corrispondente; se la stringa non è vuota, chiama il metodo *search* dell'oggetto *tweetHandler* con parametro uguale al contenuto della barra di ricerca. In seguito chiama il metodo per inserire i tweet ottenuti nella ListView (*loadTweetList()*) e quello per creare la word cloud (*createWordCloud()*).

```
1 public void popularSearch(){
2     String searchText = searchBar.getText();
3     if (searchText.length()>0){
4         jsonTweetList = tweetHandler.search(searchText);
5         loadTweetList();
6         createWordCloud();
7     }
8 }
```

- *StreamSearch*
  Questo metodo gestisce il click sul pulsante "Stream Search". Inizialmente recupera il testo dalla barra di ricerca corrispondente; se la stringa è valida, allora viene chiamato il metodo *startStreamSearch* dell'oggetto *tweetHandler* con parametri: la stringa di ricerca, il valore della checkbox "Localizzati" (booleano). Infine viene abilitato il pulsante "Stream Stop" e vengono disabilitati gli altri.

```
1 public void streamSearch(){
2     String searchText = streamSearchBar.getText();
3         if (searchText.length() > 0){
4             tweetHandler.startStreamSearch(searchText,
5                                 localizedCheckBox.isSelected());
6         }
```

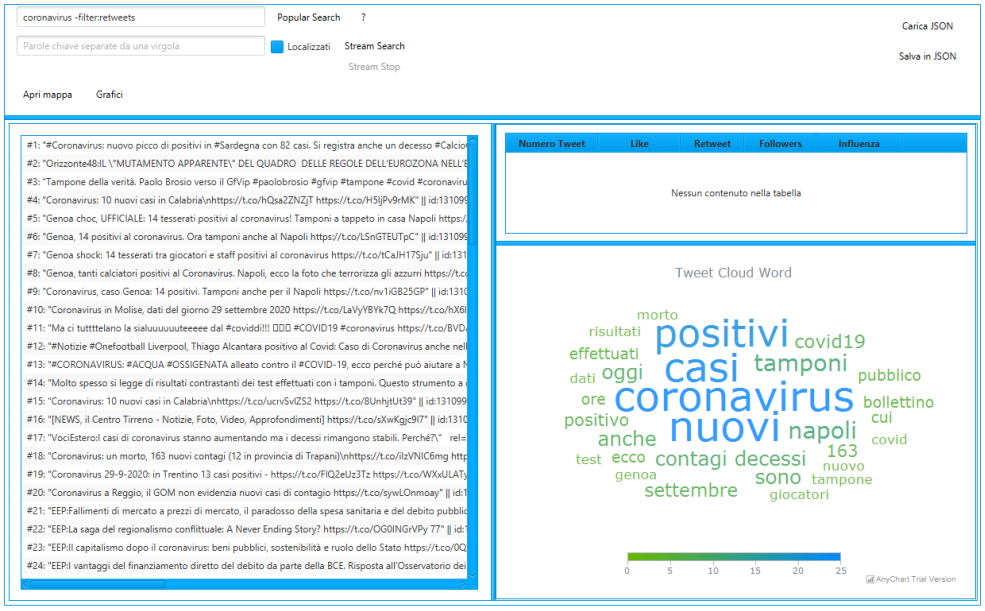

Figura 3.4: Stato dell'interfaccia principale dopo aver effettuato una ricerca per popolarità con stringa "coronavirus -filter:retweets".

```
7       //qui vengono disabilitati i pulsanti
8   }
```

- *StopStreamSearch*
  Questo metodo semplicemente chiama l'omonimo metodo stopStream-Search appartenente all'oggetto tweetHandler, il quale restituisce una lista Json di tweet, assegnata poi all'oggetto jsonTweetList. In seguito vengono riabilitati i pulsanti e vengono chiamate le funzioni per popolare la lista di tweet, per generare la word cloud e per aggiungere le posizioni dei tweet alla lista *positions*.

```
1   public void stopStreamSearch(){
2           jsonTweetList = tweetHandler.stopStreamSearch();
```

```
        /*
            qui vengono riabilitati i pulsanti
        */
        loadTweetList();
        createWordCloud();
        initializeTweetPositions();
}
```

- *CreateWordCloud*
  Questo metodo gestisce la configurazione della webView che mostrerà la cloud; dato che la libreria utilizzata è per JavaScript, l'utilizzo delle sue funzioni è implementato in un file HTML (*wordCloud.html*) caricato dal web engine della view. Tramite uno *StringBuilder* e un ciclo for, viene generata la stringa da passare alla libreria AnyChart per la realizzazione della word cloud: "*data*". Una volta che la pagina HTML è stata caricata, viene chiamato il metodo Javascript *initialize* presente nel file HTML con la stringa "*data*" come parametro. Il metodo *webViewClickListener* verrà spiegato più avanti.

```
public void createWordCloud(){
    StringBuilder cloudWordText = new StringBuilder();
    for (JsonElement o: jsonTweetList){
        cloudWordText.append(o.getAsJsonObject().get("text")
                                                .toString());
    }
    String data = cloudWordText.toString();
    WebEngine engine = wordCloudWebView.getEngine();
    //inizio operazioni da eseguire una volta che la pagina è
    //stata caricata
      JSObject jsObject =(JSObject)engine.executeScript("window");
      jsObject.call("initialize", data);
      webViewClickListener(engine, jsObject);
    // fine operazioni
```

```
15      engine.load(getClass().getResource("wordCloud.html").
16      toString());
17      engine.setJavaScriptEnabled(true);
18  }
```

Il metodo JavaScript *initialize* è quello che si occupa di generare la word cloud tramite le librerie AnyChart. Come parametro riceve la variabile "*data*" che conterrà un testo su cui effettuare il parsing; le opzioni per tale parsing sono ben chiare nel codice sorgente:

```
1  function initialize(data) {
2    var chart = anychart.tagCloud();
3    chart.data(data, {
4     mode:"byWord",
5     maxItems:30,
6     minLength:3,
7  //qui sono state inserite delle parole da non inserire nella
8  //word cloud. L'elenco comprende articoli e preposizioni italiane.
9  //ignoreItems: [...]
10   });
11 //qua sono presenti opzioni per regolare l'aspetto della wordcloud
12   chart.draw();
13 };
```

Infine, per quanto riguarda il metodo *webViewClickListener*, questo si occupa di gestire appunto l'evento click all'interno della view della word cloud. In particolare richiama un metodo JavaScript nel file wordCloud.html che restituisce la parola su cui l'utente ha cliccato; ottenuta questa parola, il metodo Java crea una nuova word cloud composta solo dai termini contenuti nei tweet in cui compare la parola cliccata. In questo modo, la cloud word generata è quella relativa al termine cliccato.

- *loadJson* e *saveToJson*
  Questi due metodi gestiscono il salvataggio e il caricamento dei tweet in formato JSON, in particolare vengono inseriti in un *JSON Array*. Entrambe le funzionalità sono implementate tramite *FileChooser*, una classe appartenente alle librerie di JavaFX che consente di mostrare finestre per la selezione o per il salvataggio di file.

- *initializeTweetPositions*
  Il metodo si occupa di popolare la lista *positions* composta da oggetti *Position*. I tweet ottenuti in risposta alle richieste delle API, hanno due campi interessanti per la posizione: *geoLocation* e *place*. GeoLocation contiene delle coordinate (latitudine e longitudine) che indicano una posizione precisa: i tweet che hanno dei dati in questo campo (cioè quelli in cui il campo geoLocation non è nullo) sono tweet geolocalizzati con precisione; basterà creare un nuovo oggetto Position e passare al suo costruttore le coordinate contenute nel suddetto campo. Nel caso in cui geoLocation sia vuoto, la prossima posizione più precisa che si può ricavare dal tweet è quella presente nel campo place (a meno che non sia vuoto anch'esso, ovviamente). Tuttavia place indica un'area e non un punto preciso, di conseguenza conterrà una bounding box delimitata da 4 punti (8 coordinate); per trovare la posizione più precisa occorre calcolare le coordinate del centro di quest'area e passare quelle al costruttore del nuovo oggetto Position. Per i motivi riportati nel paragrafo 2.2.1 a pagina 8, è fondamentale lo sfruttamento del campo place, in quanto geoLocation sarà vuoto nella maggior parte dei casi.

3.4.3 Controller Grafici

La finestra contenente i grafici viene apera tramite un click sul pulsante "Grafici"; tale click è associato a *openChart*, metodo che si occupa di

Figura 3.5: Word cloud generata dalla ricerca per popolarità dei tweet contenenti la parola "Covid".

creare una nuova finestra per mostrare il file *chart.fxml*. La visualizzazione di questa finestra è gestita dalla classe *ChartController*. Qui, con la lista di tweet passati dal controller principale, vengono generati i grafici corrispondenti. In particolare l'utente può scegliere se visualizzare il numero di tweet totali raccolti nel tempo o il numero di tweet per regione raccolti nel tempo. Un esempio di grafico è la figura 3.6 qui di seguito.

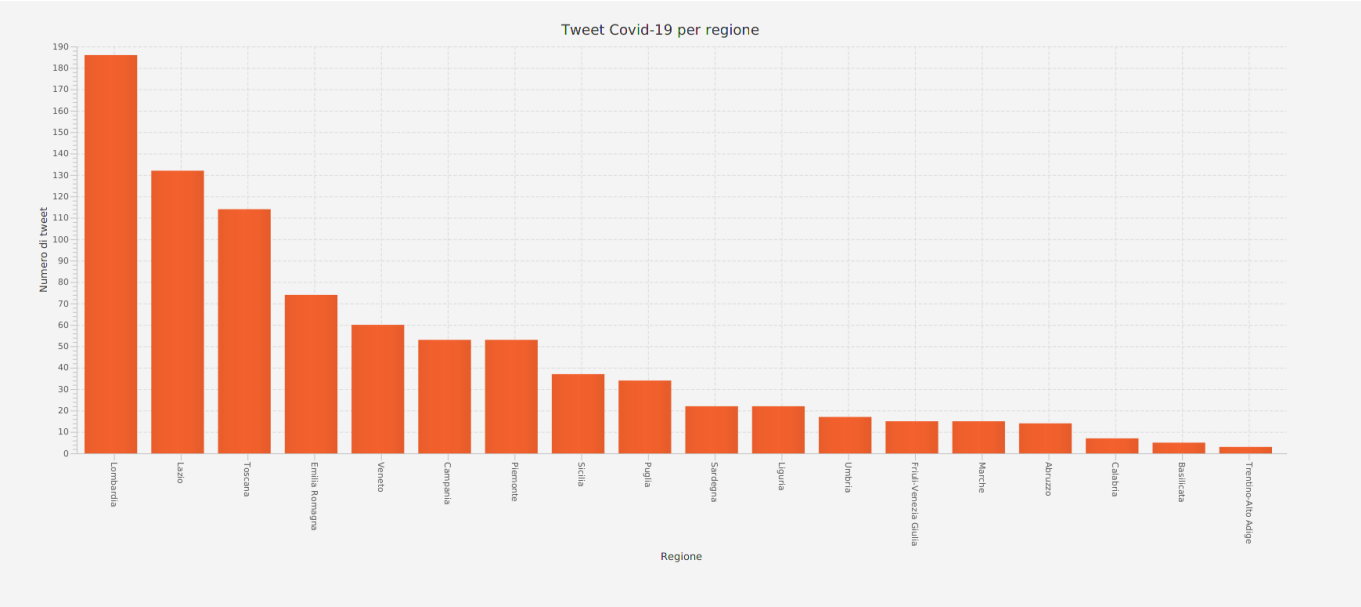

Figura 3.6: Grafico a barre che mostra il numero di tweet geolocalizzati (con posizione esatta) pubblicati dalle regioni italiane sul Covid-19 tra il 20 e il 26 Maggio 2020.

3.4.4 Controller mappa

La mappa viene aperta tramite il click sul pulsante "Apri mappa" nella finestra principale. In particolare viene chiamato il metodo *openMap* all'interno del controller principale, il quale gestisce la creazione di una nuova finestra che mostra il file FXML *map.fxml*. All'interno di *openMap* viene

passata la lista delle posizioni dei tweet (contenute in degli appositi oggetti) al controller della mappa(*MapController*). All'interno di quest'ultimo troviamo un ovverriding del metodo *initialize*, dovuto al fatto che ogni controller implementa l'interfaccia *Initializable*. Questo metodo si comporta similmente a quello della webView per la word cloud: carica il file *map.html* che contiene il codice JavaScript per utilizzare le funzionalità della libreria di Google Maps. Dopo aver verificato che la pagina è stata caricata, viene chiamato il metodo JavaScript *initMap* e, per ogni oggetto posizione passato in precedenza, il metodo *addMarker*.

```
1  @Override
2  public void initialize(URL arg0, ResourceBundle arg1) {
3      engine = mapView.getEngine();
4      engine.load(getClass().getResource("map.html").toString());
5      engine.setJavaScriptEnabled(true);
6      JSObject window = (JSObject) engine.executeScript("window");
7      window.setMember("app", this);
8      //inizio operazioni da eseguire una volta che la pagina è
9      //stata caricata
10         engine.executeScript("initMap()");
11         for (Position p: positions){
12           window.call("addMarker",p.getLatitude(),
13                     p.getLongitude(),p.getDescription());
14         }
15     //fine operazioni
16 }
```

Per quanto riguarda la funzione Javascript *initMap*, al suo interno troviamo l'inizializzazione della mappa e delle relative opzioni. In particolare è qua che vengono decisi il livello di zoom, il punto iniziale da mostrare e il tipo di mappa.

```
function initMap() {
  //Coordinate di Roma
  var latlng = new google.maps.LatLng(41.902782,12.496366);
  var myOptions = {
    zoom: 6,
    center: latlng,
    mapTypeId: google.maps.MapTypeId.ROADMAP,
    mapTypeControl: false,
    navigationControl: false,
    streetViewControl: false,
    backgroundColor: "#666970"
  };
  document.map = new google.maps.Map(
                document.getElementById("map_canvas"),myOptions);
}
```

Infine il metodo *addMarker*, data una posizione (latitudine e longitudine) e una descrizione, aggiunge il corrispondente marker sulla mappa.

```
function addMarker(latitude,longitude,description){
  var marker = new google.maps.Marker({
    position: new google.maps.LatLng(latitude,longitude),
    title: description
  });
  marker.setMap(document.map);
}
```

3.4.5 Gestione richieste di tweet

Le comunicazioni con i server di Twitter sono gestite interamente nella classe *TweetHandler*. I metodi qua contenuti sono:

- *Search*: questo metodo, data una stringa come parametro, richiama l'API search di Twitter tramite la classe *Query* della libreria Twit-

ter4J. All'istanza di *Query* viene passata la stringa di ricerca e alcuni parametri relativi alla posizione in cui cercare e alla lingua dei tweet. In particolare richiede tweet in italiano e localizzati entro un certo raggio da Roma (raggio che comprende tutta l'Italia). Il metodo restituisce i tweet in formato JSON.

```
public JsonArray search(String searchTerm){
    List<Status> tweetList = new ArrayList<>();
    Query query = new Query(searchTerm);
    query.setGeoCode(new GeoLocation(41.902782, 12.496366),
                    650, Query.KILOMETERS);
    query.setLang("it");
    try {
        tweetList = twitter.search(query).getTweets();
    } catch (TwitterException e) {
        e.printStackTrace();
    }
    return gson.toJsonTree(tweetList).getAsJsonArray();
}
```

- *startStreamSearch*: questo metodo avvia la comunicazione con Twitter per la ricezione di tweet in tempo reale. Al suo interno è presente un listener che, alla ricezione di un tweet, se questo contiene le parole cercate lo aggiunge alla lista *streamTweets*. Alla richiesta è applicabile un filtro con diverse opzioni e queste sono combinate con un "OR". Questo significa che, se si cercano tweet in Italia **e** contenenti la parola "Esempio", è necessario usare una sola opzione nel filtro e verificare l'altro requisito nel momento in cui il tweet viene ricevuto.

```
public void startStreamSearch(String searchText,boolean localized)
{
```

```
3       //L'utente può inserire più stringhe separate da virgola
4       String[] keywords = searchText.toLowerCase().split(",");
5       StatusListener listener = new StatusListener() {
6           @Override
7           public void onStatus(Status status) {
8               String statusText = status.getText();
9               if (Arrays.stream(keywords).parallel().
10                  anyMatch(statusText.toLowerCase()::contains)){
11                  streamTweets.add(status);
12              }
13          }
14          //qui sono presenti altri metodi del listener
15      };
16      twitterStream.addListener(listener);
17      FilterQuery filterQuery = new FilterQuery();
18      //italyBB è la bounding box dell'Italia
19      double[][] italyBB = {{long1, lat1}, {long2, lat2}};
20      filterQuery.locations(italyBB);
21      //se l'utente vuole tweet geolocalizzati allora viene
22      //utilizzata la bounding box, altrimenti si richiedono
23      //solo i tweet contenenti le parole chiave
24      if (localized)
25          twitterStream.filter(filterQuery);
26      else
27          twitterStream.filter(keywords);
28  }
```

- *stopStreamSearch*: qui viene interrotta la comunicazione con Twitter e i tweet salvati nella lista *streamTweets* vengono restituiti come *JsonArray*.

Capitolo 4

Conclusioni e sviluppi futuri

4.1 Conclusioni

Un insieme di dati correlati tra loro, se interpretati e quindi resi significativi, forniscono un' informazione. Attualmente il traffico su Internet è sempre maggiore, ma solo una piccola porzione di dati diventa informazione. Il punto di partenza più semplice per ottenere un dato significativo, lo offrono i social; questi ultimi possono essere visti come enormi contenitori di dati, dato il loro traffico giornaliero. In questo studio è stato scelto il social Twitter per la natura dei suoi contenuti, generalmente concisi, e per le funzionalità delle sue API. Tweet Tracker più che essere uno strumento per analizzare efficacemente tutti i tweet, vuole mostrare le potenzialità delle suddette API per scopi di ricerca e recupero di informazioni su un territorio. In casi di emergenza o di eventi significativi di grande portata, le persone tendono a comunicare la loro esperienza sui social; se questi contenuti vengono geolocalizzati, possiamo parlare effettivamente di "sensori umani" sul territorio. Tuttavia, tali esperienze diventano informazione solo se interpretate correttamente da chi le recupera. In conclusione, Tweet Tracker è da considerarsi come un punto di partenza, uno spunto, per la realizzazione di un ambiente che elabori e memorizzi una grande quantità

di tweet e di dati su di essi, in modo da facilitare il lavoro in un ambiente di ricerca su Twitter.

4.2 Sviluppi futuri

Gli sviluppi futuri possibili per Tweet Tracker e per questo progetto in generale, sono numerosi e dipenderanno dai bisogni di chi intende effettuare l'analisi. Le proposte sono:

- Implementare le funzionalità della versione 2.0 delle API di Twitter appena saranno disponibili. Come già detto nel paragrafo 2.2.1, la nuova versione delle API offre strumenti per una ricerca più accurata, migliorando la qualità dei dati raccolti.
- Aggiungere la possibilità di consultare ulteriori grafici diversi da quelli già presenti.
- Aggiungere una bounding box per ogni regione e permettere all'utente di scegliere da quale regione ricevere i tweet.
- Implementare un sistema in grado di gestire i tweet contenenti delle immagini.
- Implementare un sistema per visualizzare gli argomenti in tendenza per una località scelta dall'utente.
- Riorganizzare l'interfaccia per renderla più intuitiva e gradevole per l'utente.

Bibliografia

[1] J. Clement. *Global digital population as of July 2020*. URL: https://www.statista.com/statistics/617136/digital-population-worldwide/. 2020. Consultato il 30 settembre 2020.

[2] J. Clement. *Number of monthly active Facebook users worldwide as of 2nd quarter 2020*. URL: https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/. 2020. Consultato il 30 settembre 2020.

[3] J. Clement. *Number of monthly active Twitter users worldwide from 1st quarter 2010 to 1st quarter 2019*. URL: https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/. 2019. Consultato il 30 settembre 2020.

[4] Rosen, Aliza, Ikuhiro Ihara. *Giving you more characters to express yourself*. URL: https://blog.twitter.com/official/en_us/topics/product/2017/Giving-you-more-characters-to-express-yourself.html. 2017. Consultato il 1 ottobre 2020.

[5] Lin, Ying. *10 Twitter Statistics Every Marketer Should Know in 2020 [Infographic]*. URL: https://www.oberlo.com/blog/twitter-statistics#:~:text=There%20are%20330%20million%20monthly,daily%20active%20users%20on%20Twitter. 2020. Consultato il 30 settembre 2020.

[6] D'Monte, Leslie. *Swine flu's tweet tweet causes online flutter*. URL: https://www.business-standard.com/article/technology/swine-flu-s-tweet-tweet-causes-online-flutter-109042900097_1.html. 2009. Consultato il 24 settembre 2020.

[7] Alex Burns e Ben Eltham. «Twitter Free Iran: an Evaluation of Twitter's Role in Public Diplomacy and Information Operations in Iran's 2009 Election Crisis». In: *Record of the Communications Policy and Research Forum* 298–310 (gen. 2009).

[8] Lüfkens, Matthias. *Twiplomacy 2015: How World Leaders Connect on Twitter*. URL: https://twiplomacy.com/blog/twiplomacy-study-2015/. 2015. Consultato il 24 settembre 2020.

[9] URL: https://developer.twitter.com/en/docs/twitter-api/v1. Consultato il 25 settembre 2020.

[10] URL: https://developer.twitter.com/en/docs/twitter-api/migrate. Consultato il 26 settembre 2020.

[11] Twitter Support (@TwitterSupport). *Most people don't tag their precise location in Tweets, so we're removing this ability to simplify your Tweeting experience. You'll still be able to tag your precise location in Tweets through our updated camera. It's helpful when sharing on-the-ground moments.* URL: https://twitter.com/TwitterSupport/status/1141039841993355264. Tweet. 2019.

[12] Porter, Jon. *Twitter removes support for precise geotagging because no one uses it*. URL: https://www.theverge.com/2019/6/19/18691174/twitter-location-tagging-geotagging-discontinued-removal. 2019. Consultato il 27 settembre 2020.

[13] URL: https://developer.twitter.com/en/docs/authentication/overview. Consultato il 27 settembre 2020.

[14] URL: https://developer.twitter.com/en/docs/twitter-api/v1/rate-limits. Consultato il 1 ottobre 2020.

[15] URL: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/overview/standard. Consultato il 27 settembre 2020.

[16] URL: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/search/api-reference/get-search-tweets. Consultato il 27 settembre 2020.

[17] URL: https://developer.twitter.com/en/docs/twitter-api/v1/tweets/filter-realtime/guides/basic-stream-parameters. Consultato il 28 settembre 2020.

[18] URL: https://openjfx.io/. Consultato il 29 settembre 2020.

[19] URL: http://twitter4j.org/. Consultato il 29 settembre 2020.

[20] URL: https://cloud.google.com/maps-platform/. Consultato il 30 settembre 2020.

[21] URL: https://www.anychart.com/. Consultato il 30 settembre 2020.

Ringraziamenti

Innanzi tutto vorrei ringraziare il Professore Paolo Ciancarini per avere sempre messo a disposizione il suo tempo per guidarmi in questo lavoro.

Un grazie a tutti i nuovi amici che ho incontrato durante questo percorso, amici con cui ho condiviso gioie e dolori da studenti universitari, amici che mi hanno ospitato a casa loro e sono stati ospiti a casa mia.

Ovviamente, un enorme grazie anche ai vecchi amici. Nonostante la distanza che ci ha separato per la maggior parte del tempo di questi ultimi anni, vi siete sempre fatti sentire e l'ho apprezzato molto.

Infine, i ringraziamenti più grandi sono per la mia ragazza Ilaria e per la mia famiglia. Ringrazio Ilaria per avermi supportato e sopportato per 8 lunghissimi anni: sei la persona che mi è stata più vicina e più d'aiuto nei momenti in cui ne avevo bisogno. Per ultima, ma non per importanza, ringrazio la mia famiglia per avermi sempre lasciato libera scelta durante il mio percorso (non solo universitario), dandomi supporto incondizionato.