



**UNIVERSITÀ  
DEGLI STUDI DI BARI  
ALDO MORO**

---

**DIPARTIMENTO DI INFORMATICA**

**CORSO di LAUREA in INFORMATICA**

# **Estensione delle funzionalità di una piattaforma di Social Network Analysis e Content Analytics.**

Relatori:

**Dott. Pasquale Lops**

**Dott. Cataldo Musto**

Laureando:

**Volpicella Savino Davide**

Anno Accademico 2021/2022

# Indice

<b>Introduzione .....</b>	4
<b>Capitolo1: I Social Media .....</b>	5
<b>1.1 L'avvento dei social media nelle nostre vite .....</b>	5
<b>1.2 Cosa sono i Social Media.....</b>	6
<b>1.3 La privacy nei social media .....</b>	8
<b>1.4 Social media analytics .....</b>	9
<b>1.5 Sentiment Analysis.....</b>	11
<b>Capitolo2: Stato dell'arte .....</b>	14
<b>2.1 Introduzione di ITA e CrowdPulse .....</b>	14
<b>2.2 Italian Tweets Analyzer.....</b>	15
<b>2.2.1 Estrazione dei tweets.....</b>	16
<b>2.2.2 Elaborazione dei tweets.....</b>	23
<b>2.2.2.1 Sentiment Analysis.....</b>	23
<b>2.2.2.1.1 SENTIPOLC .....</b>	23
<b>2.2.2.1.2 Feel-it.....</b>	25
<b>2.2.2.2 Natural language processing .....</b>	27
<b>2.2.2.3 Entity Linking .....</b>	31
<b>2.2.2.4 Geocoding.....</b>	33
<b>2.2.2.5 File di configurazione per l'elaborazione dei tweets.....</b>	35
<b>2.2.3 Ritrovamento di informazioni sugli utenti .....</b>	37
<b>2.2.4 Gestione dei tweets .....</b>	38
<b>2.3 CrowdPulse .....</b>	39
<b>2.3.1 Architettura della piattaforma .....</b>	39
<b>2.3.1.1 React.js front end .....</b>	40
<b>2.3.1.2 Express.js e Node.js server .....</b>	40
<b>2.3.1.3 MongoDB.....</b>	41
<b>2.3.2 Backend .....</b>	41
<b>2.3.3 Frontend .....</b>	43
<b>2.3.3.1 Filters .....</b>	44
<b>2.3.3.2 Charts.....</b>	45
<b>2.3.3.3 Table .....</b>	46
<b>2.3.4 Funzionalità principali di CrowdPulse .....</b>	46
<b>2.3.4.1 Sentiment .....</b>	48
<b>2.3.4.2 Word .....</b>	48

<b>2.3.4.3 TimeLines.....</b>	49
<b>2.3.4.4 TweetList .....</b>	49
<b>2.3.4.5 Maps .....</b>	50
<b>Capitolo3: Integrazione di nuove funzionalità .....</b>	51
<b>3.1 HuggingFace.....</b>	51
<b>3.2 Modifiche effettuate su ITA.....</b>	52
<b>3.2.1 Hate speech it.....</b>	52
<b>3.2.2 Genre Classification: Roberta .....</b>	55
<b>3.2.3 Image to text: vit-gpt2 .....</b>	58
<b>3.3 Modifiche effettuate su CrowdPulse.....</b>	62
<b>3.3.1 Aggiunta dell'algoritmo per hate speech .....</b>	62
<b>3.3.2 Integrazione di un nuovo filtro per text categorization .....</b>	66
<b>3.3.3 Modifica dei filtri Algorithm e Sentiment .....</b>	70
<b>3.3.4 Realizzazione di una nuova sezione per le captions .....</b>	72
<b>3.3.5 Generazione di WordCloud tramite captions .....</b>	77
<b>Capitolo 4: Sperimentazione.....</b>	79
<b>4.1 Individuazione delle forme d'odio in Italia .....</b>	79
<b>4.1.1 Misoginia .....</b>	80
<b>4.1.2 Omofoobia.....</b>	86
<b>4.1.3 Xenofobia .....</b>	91
<b>4.2 Valutazione dell'opinione pubblica in merito a determinati eventi.....</b>	96
<b>4.2.1 Russia .....</b>	96
<b>4.2.2 Ucraina.....</b>	101
<b>Sviluppi futuri .....</b>	105
<b>Bibliografia.....</b>	105

## Introduzione

Il lavoro di tesi, iniziato a ottobre del 2022 e terminato a marzo del 2023, ha avuto come obiettivo principale quello di estendere le funzionalità di due piattaforme: Hate Tweet Map e CrowdPulse.

Hate Tweet Map è un programma che permette di effettuare la ricerca di post pubblicati su Twitter, di salvarli all'interno di un database e di elaborarli. Il mio compito in merito a questa piattaforma è stato quello di:

- integrare un nuovo modello in grado di effettuare sentiment analysis (hate speech it);
- aggiungere un classificatore per il testo presente nei tweets;
- aggiungere una nuova funzionalità in grado di effettuare image-to-text.

Dato che su Hate Tweet Map sono presenti sempre più funzionalità, è stato ritenuto necessario ribattezzarlo con un nome più generico, ovvero: Italian Tweets Analyzer (ITA).

Crowdpulse, invece, è un'applicazione che consente di visualizzare i dati raccolti da Twitter sotto forma di grafici e tavole, in modo tale da consentire a chi lo utilizza di effettuare delle analisi. Le modifiche effettuate su questa piattaforma hanno riguardato la possibilità di visionare i nuovi dati ottenuti dai nuovi modelli integrati su ITA, nello specifico:

- integrazione di hate speech it;
- integrazione del modello per la classificazione del testo;
- definizione di una nuova sezione dedicata ai media dei tweet;
- aggiunta la possibilità di definire una word cloud tramite captions.

Le modifiche effettuate sulle piattaforme vengono spiegate in modo dettagliato nel capitolo tre, mentre nel secondo vengono spiegate le strutture dei due tools e loro funzionalità principali. Il primo capitolo, invece, ha lo scopo di spiegare il motivo per cui i due tools possono essere utili, effettuando prima una premessa sull'impatto dei social sulla nostra società. Infine, nell'ultimo capitolo, si riportano delle sperimentazioni effettuate attraverso il loro utilizzo.

## Capitolo1: I Social Media

### 1.1 L'avvento dei social media nelle nostre vite

Nell'era digitale in cui ci troviamo, i social media sono una parte sempre più importante della nostra vita quotidiana. Essi hanno rivoluzionato la comunicazione al punto tale che rappresentano il nostro mezzo preferito di comunicazione quotidiana. Prima dei social media, se si fosse voluto stare al passo con le notizie, si sarebbe dovuto andare in edicola di prima mattina e comprare un'edizione locale che riportava gli eventi dei giorni scorsi. Oggi, riceviamo tutte le ultime notizie sulla guerra in Ucraina o sulle recenti attività di Giorgia Meloni in tempo reale.

I social hanno scaturito un grande impatto nelle nostre vite, al punto tale che l'ultima cosa che si fa prima di andare a dormire è scorrere i post presenti in bacheca, ed è anche l'attività che si svolge anche prima di alzarsi dal letto. Ogni persona lascia online una traccia di sé, come ad esempio il proprio pensiero, le proprie preferenze, le emozioni verso determinati topic. Tale traccia può essere utilizzata in diversi modi, ad esempio le imprese possono sfruttarla per capire quali sono i prodotti di maggior interesse dei diversi utenti. L'analisi dei dati presenti sui social è detta social medial analytics. Prima di entrare nel dettaglio di questo

argomento, si illustra cosa sono i social media e come questi vengono utilizzati.

## 1.2 Cosa sono i Social Media

I social media sono tecnologie che facilitano la condivisione di idee ed informazioni attraverso reti virtuali. Da Facebook e Instagram a Twitter e YouTube, i social media coprono un ampio universo di applicazioni e piattaforme che consentono agli utenti di condividere contenuti, interagire online e costruire comunità. Più di 4,7 miliardi di persone utilizzano i social media, pari a circa il 60% della popolazione mondiale.

Oggi, le applicazioni e le piattaforme di messaggistica dei social media sono i siti più comunemente utilizzati in tutto il mondo. L'articolo "Digital Report 2023" [1] riporta che attualmente ci sono 5.16 persone che si collegano ad internet e che ci sono 4.76 miliardi di utenti di social media in tutto il mondo, pari a poco meno del 60% della popolazione globale totale.

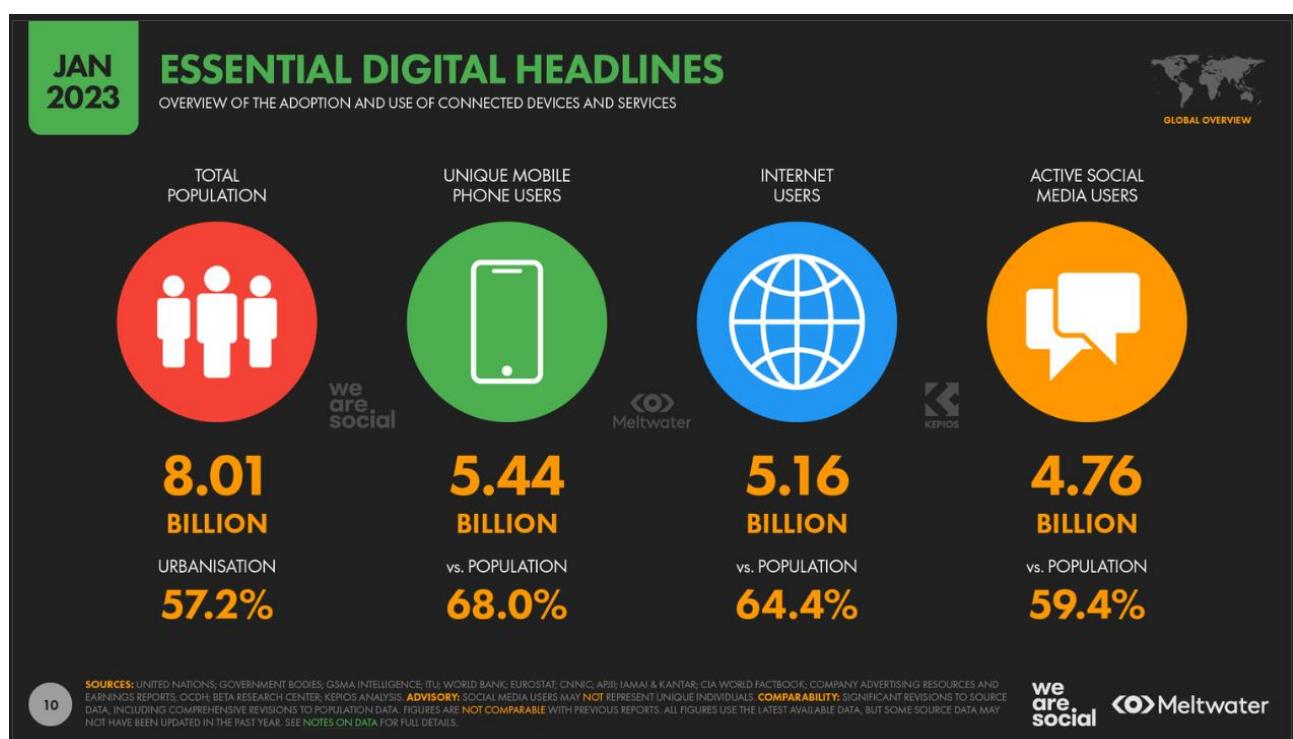
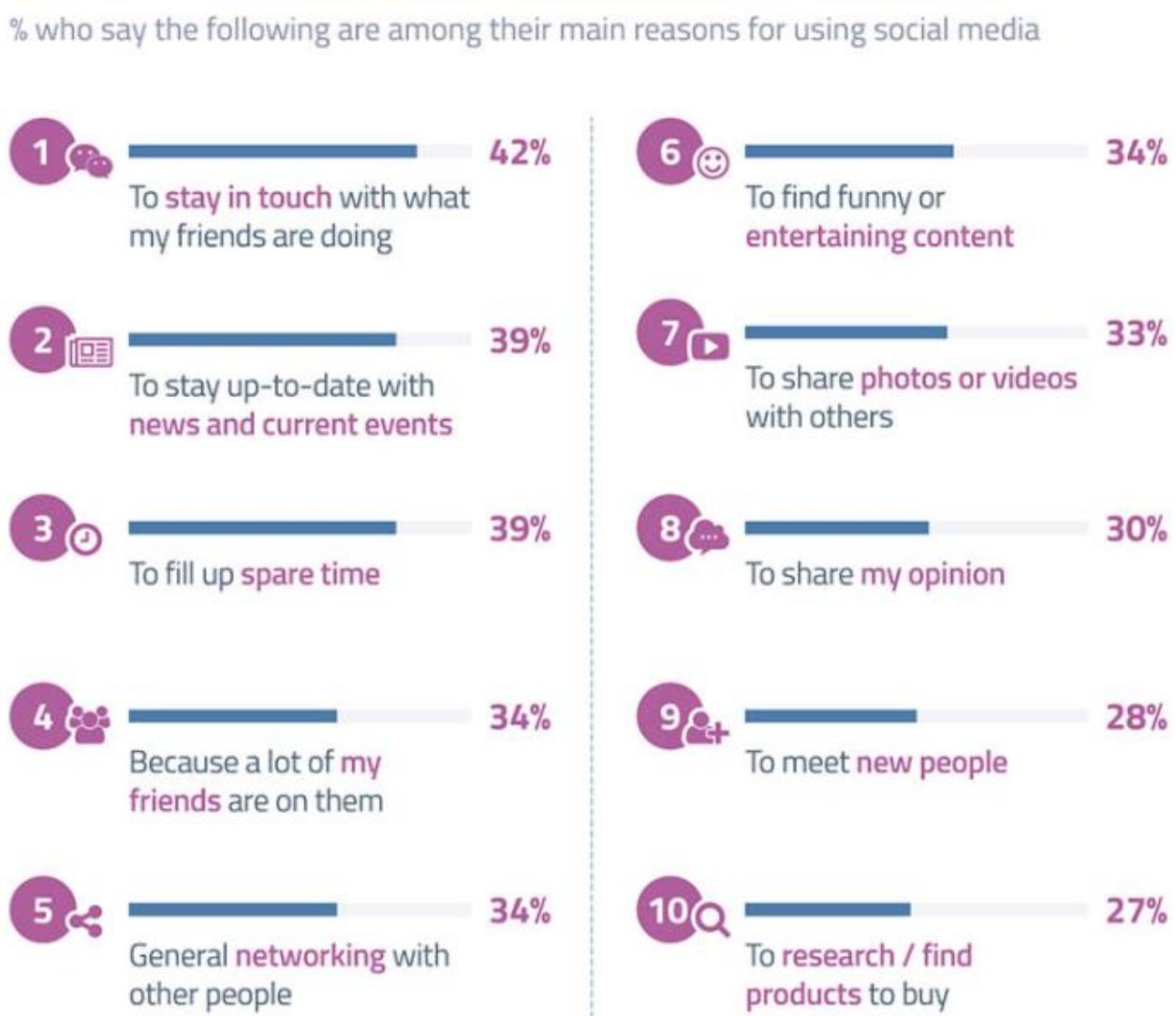


Figura 1 - report numero utenti sui social

Ma quali sono i motivi per cui i social sono molto utilizzati?

Secondo le interviste effettuate dall'organizzazione “Global Web Index” [2], effettuate su un campione di 72892 utenti di una fascia di età compresa tra i 16 e 64 anni, è stato riscontrato che il 42% degli utenti utilizza i social per restare in contatto con i propri amici, mentre il 39% degli utenti usa i social per informarsi, leggere news e restare aggiornato su ciò che accade ogni giorno.



Question: What are your main reasons for using social networking services?

Source: GlobalWebIndex Q1 2017 | Base: 72,892 Internet Users aged 16-64

Figura 2 - utilizzo dei social media

Questi dati riportano che la maggior parte delle persone utilizzano i social in modo passivo, in altre parole l'utente medio percepisce le reti social come fonti di contenuti piuttosto che come piattaforma che richiedono un contributo attivo.

Uno dei motivi per cui i social vengono utilizzati in modo passivo è dovuto dal fatto che gli utenti sono preoccupati per la loro privacy.

### 1.3 La privacy nei social media

La preoccupazione degli utenti nei confronti della propria privacy è scaturita da quando è avvenuto lo scandalo che ha riguardato Facebook e Cambridge Analytica [3], uno dei maggiori scandali politici avvenuti all'inizio del 2018, quando fu rivelato che Cambridge Analytica aveva raccolto i dati personali di 87 milioni di account Facebook senza il loro consenso e li aveva usati per scopi di propaganda politica.

A seguito di queste vicende giudiziarie, sono state introdotte delle leggi per la salvaguardia nella privacy, tra queste ritroviamo il California Consumer Privacy Act del 2018 [4], che richiede ai siti di notificare i propri utenti in merito a quali informazioni personali stanno raccogliendo e cosa ne potrebbero fare. Da quando la legge è entrata in vigore il 1° gennaio 2020, un numero crescente di siti ha utilizzato i pop-up per far sapere alle persone che utilizzano i cookie e offrire loro la possibilità di interrompere la vendita delle proprie informazioni personali.

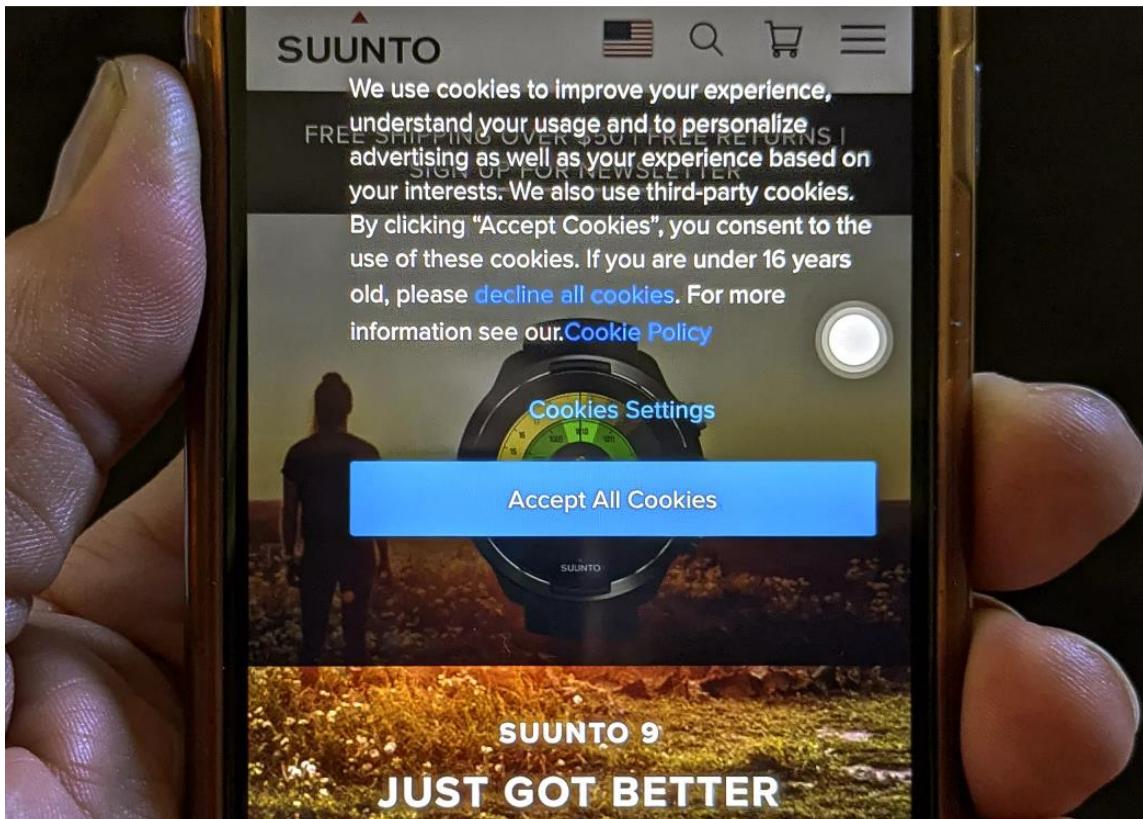


Figura 3 - pop-up per l'accettazione dei cookies

Tuttavia, anche un uso passivo permette ai siti di ottenere diverse informazioni. Azioni come il mettere mi piace, commentare o condividere un post nei social al giorno d'oggi sono viste come fonti di informazione.

#### 1.4 Social media analytics

Con social media analytics [5] si intende la raccolta, integrazione ed analisi dei dati provenienti dai social network come Facebook, Instagram, Twitter, e così via. Questa è un'attività svolta principalmente dai social media marketing.

Quest'ultima è una professione nata da quando i social sono entrati nelle vite di tutti quanti e, al giorno d'oggi, realizzare una strategia di social media marketing è fondamentale per tutti i

brand che vogliono ampliare il mercato aumentando la base dei loro clienti o cercando nuove opportunità di business.

Ma perché i dati presenti sui social sono importanti?

Attraverso i dati generati dai social si possono effettuare analisi con le quali, ad esempio, è possibile:

- Comprendere come un prodotto lanciato sul mercato è stato percepito dagli utenti;
- Capire la direzione di un dibattito esistente in rete su specifiche tematiche e argomenti;
- Capire se le azioni di un personaggio pubblico sono state gradite o meno da altri utenti.

Tuttavia, le informazioni presenti nei social si presentano in maniera non strutturata, ovvero non organizzati in righe e tabelle, ma si presentano sotto forma di testi, audio, video, immagini, mappe, ecc.

Oggi, abbiamo le condizioni ideali per poter estrarre valore in maniera automatica e massiva anche da dati destrutturati. Lo sviluppo di algoritmi di deep learning che, attraverso reti neurali artificiali, simulano il funzionamento del cervello umano e possono elaborare dati molto complessi che non sono presentati in una tabella: tecniche come la computer vision e il natural language processing consentono l'analisi e l'interpretazione di grandi masse di dati non strutturati.

Con i dati presenti sui social è possibile effettuare diversi tipi di analisi, tra cui la sentiment analysis.

## 1.5 Sentiment Analysis

La sentiment analysis [6] è l'analisi computazionale di sentimenti ed opinioni espressi all'interno di testi generati in rete su un prodotto, un servizio, un individuo, un'organizzazione, un evento, etc.

La sentiment analysis, infatti, consente di estrarre, grazie a tecniche di data mining e di Natural Language Processing, *da documenti testuali le emozioni che l'utente ha provato* in determinati contesti al fine di determinare, per poi classificare, un oggetto (documento, espressione, frase) secondo una polarità positiva, negativa o neutrale.



Figura 4 - polarità della sentiment analysis

Come è possibile determinare la polarità di un contenuto?

Per farlo si utilizzano nozioni di machine learning, statistica e natural language processing (NLP). Gli strumenti in grado di effettuare sentiment analysis prendono in input un contenuto scritto e lo elaborano in modo tale da fornire in output un'etichetta che andrà ad indicare se il contenuto risulta essere positivo, negativo o neutro.

Questo processo può essere eseguito in modi diversi:

- Rule-based sentiment analysis: questo modello utilizza un elenco di parole, in cui ad ogni termine viene assegnato un punteggio per il sentimento, ad esempio si avrà un valore maggiore di zero per termini positivi e valori minori di zero per termini negativi. Le frasi sono valutate per la positività o la negatività complessiva utilizzando questo tipo di ponderazioni. Questo tipo di sistema richiede però particolari attenzioni per tener conto anche di frasi in cui sono presenti sarcasmo, idiomi ed altre anomalie verbali.
- Machine learning-based sentiment analysis: ad un classificatore viene assegnato un training set supervisionato, ovvero un insieme di frasi già etichettate in partenza con la loro corrispettiva polarità. Il modello impara, tramite addestramento, quali frasi sono positive o negative e successivamente potrà essere utilizzato su un nuovo set di dati.
- Hybrid Systems: modelli che sfruttano la potenza dell'apprendimento automatico insieme alla flessibilità della personalizzazione.

Qual è l'utilità della sentiment analysis?

- Le aziende che vogliono ottenere informazioni rispetto ad un prodotto da loro venduto, anziché effettuare le tradizionali indagini di mercato, possono ricorrere alla sentiment analysis. In questo modo potranno consultare una grande collezione di dati in modo immediato e spendendo poco;

- La sentiment analysis consente alle imprese di monitorare il sentimento dei propri clienti in tempo reale. Nel caso di commenti negativi, un'azienda può procedere subito con una manovra di miglioramento in modo tale che il feedback dato dagli utenti diventi positivo;
- In seguito ad una strategia di marketing, le aziende possono agire in base ai sentimenti espressi dal cliente. Le informazioni ottenute dalla sentiment analysis possono aiutare le imprese a migliorare la loro strategia di marketing;
- La sentiment analysis consente di capire le emozioni degli utenti verso un determinato evento o nei confronti di un certo personaggio.

Vista l'importanza di analizzare i dati presenti sui social, si sono realizzati due tools in grado di farlo: Italian Tweets Analyzer e CrowdPulse; i quali verranno descritti nel capitolo successivo.

## Capitolo2: Stato dell'arte

### 2.1 Introduzione di ITA e CrowdPulse

Italian Tweets Analyzer e CrowdPulse sono due tools realizzati principalmente per eseguire sentiment analysis. Nello specifico, il primo consente di estrarre i post pubblicati su Twitter, detti Tweet, e di processarli; mentre il secondo permette di visualizzare i dati raccolti attraverso grafici, tabelle e mappe.

Per introdurre il funzionamento dei due tools, sono state realizzate le seguenti pipeline:

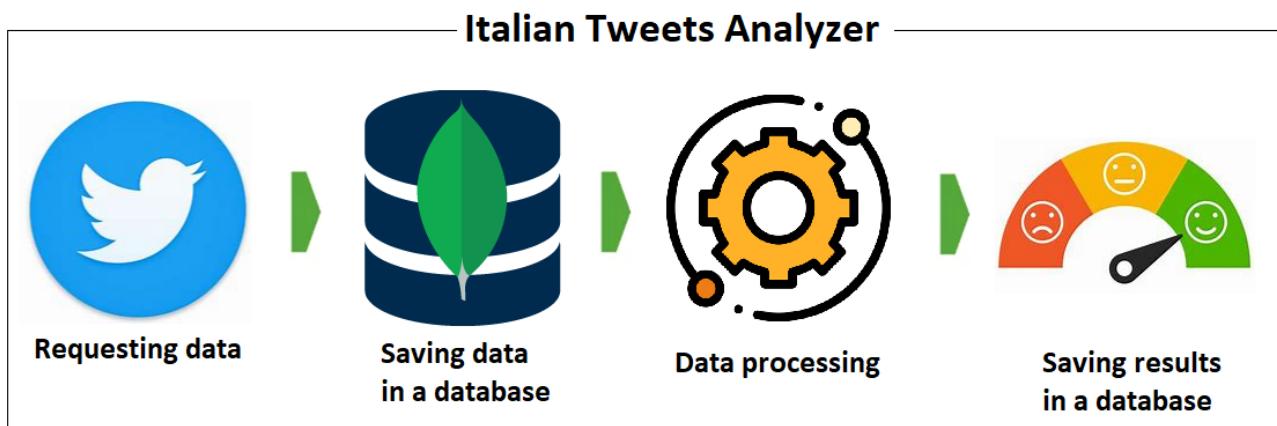
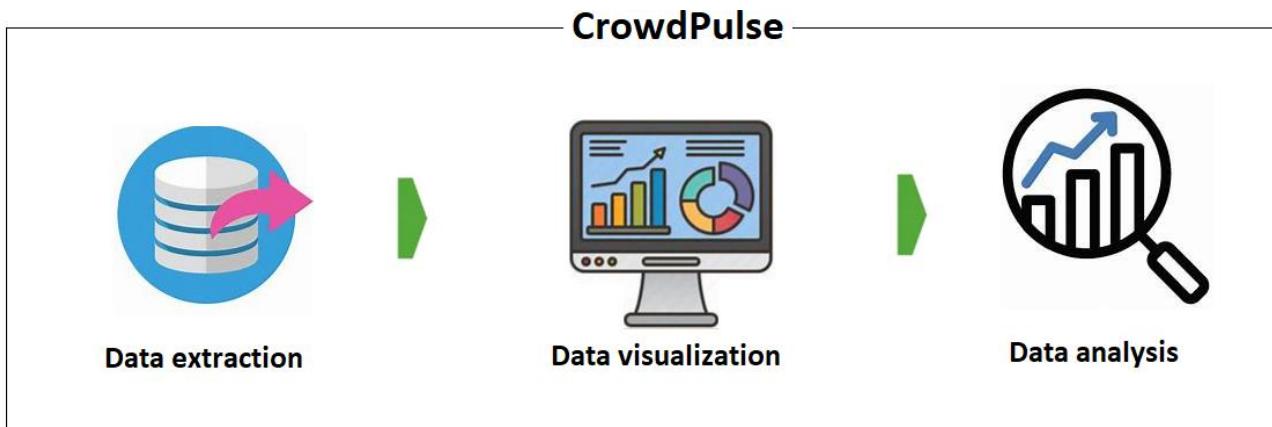


Figura 5 - pipeline di Italian Tweets Analyzer

Italian Tweets Analyzer consente di:

- Effettuare una richiesta dei dati a Twitter;
- Salvare i dati all'interno di un database;
- Processare i dati salvati;
- Salvare i risultati ottenuti all'interno del database.



*Figura 6 - pipeline di crowdpulse*

CrowdPulse consente di:

- Estrarre i risultati ottenuti tramite ITA;
- Visualizzare i dati;
- Effettuare delle analisi.

Nei paragrafi successivi, verranno spiegati i due tools nel dettaglio.

## 2.2 Italian Tweets Analyzer

ITA è un programma realizzato con il linguaggio di programmazione Python e consente di effettuare la ricerca di tweets e di analizzarli. Esso rappresenta la versione aggiornata di “Hate-Tweet-Map” [7] e le sue funzionalità principali sono le seguenti:

- Estrazione dei tweets;
- Elaborazione dei tweets;
- Ritrovamento di informazioni sugli utenti;
- Gestione dei tweets.

### 2.2.1 Estrazione dei tweets

L'estrazione dei tweets alla piattaforma Twitter avviene tramite una richiesta realizzata mediante l'utilizzo della libreria “request”, la quale consente di effettuare richieste http.

Prima di effettuare la richiesta, è necessario definire una query. ITA, infatti, consente all'utente di configurare il tipo di tweets che si vogliono ottenere, ciò è possibile farlo modificando l'apposito file di configurazione chiamato “search\_tweets.config”. In questo file sono presenti diversi campi e il modo in cui questi vengono avvalorati, consente di impostare la query da inviare a Twitter.

I campi che sono presenti nel file di configurazione possono essere suddivisi in due categorie: obbligatori e non obbligatori. Per quanto riguarda quelli obbligatori, nel momento in cui si effettua una ricerca, è necessario che almeno uno tra questi sia avvalorato. I campi obbligatori sono i seguenti:

- Keyword: parola chiave sulla quale si baserà la ricerca dei tweets, infatti, tutti i tweet che presenteranno nel testo la parola chiave indicata verranno ritrovati. In questo campo è possibile indicare più parole chiavi e possono anche essere utilizzati gli operatori logici AND e OR;
- User: campo in cui è possibile specificare gli username degli utenti, in questo modo è possibile ritrovare i tweets di una determinata persona;
- User\_mentioned: campo in cui viene inserito il nome di un utente. Se questo campo viene avvalorato, verranno ritrovati i tweets che menzionano l'utente indicato.
- Hashtag: campo in cui è possibile inserire uno specifico hashtag, in questo modo verranno ritrovati quei tweet che lo contengono.

I field opzionali sono:

- Lang: questo campo indica il linguaggio dei tweets che si vogliono ritrovare. I valori ammessi da questo campo sono quelli riportati dall'ISO 639-1, ad esempio: it, en, es;
- Context\_annotations: questo campo indica a Twitter di includere nei tweets ricercati il context annotation, ovvero un campo che indica il contesto del tweet. Ad esempio, un tweet inerente ad argomenti sportivi sarà avvalorato come “sport”;
- N\_results: campo che indica il numero massimo di tweets che si vogliono ritrovare. Twitter consente una ricerca di minimo 10 tweets e di massimo 500.
- All\_tweets: campo che indica a Twitter di ritrovare tutti i possibili tweets. Se questo campo viene avvalorato a True, allora il valore di N\_results verrà impostato automaticamente a 500.
- Time: campo che indica a Twitter di ricercare i tweets in uno specifico range temporale. Questo campo presenta due sottocampi, che sono: start\_time e end\_time. I valori ammessi da questi devono seguire il formato indicato dall'ISO 8601/RFC 3339, ad esempio: YYYY-MM-DDTHH:mm:ss+Z. Le configurazioni possibili sono le seguenti:
  - Solo start\_time viene specificato: si assumerà end\_time come la data corrente (-30 secondi);
  - Solo end\_time viene specificato: start\_time assumerà la data di 30 giorni prima della data specificata in end\_time;
  - Entrambi vengono specificati: in questo caso verranno ritrovati i tweet indicati nel range temporale;
  - Nessuno viene specificato: di default verranno ritrovati i tweet dalla data attuale fino a 30 giorni indietro.

- Geo: in questa sezione è possibile impostare dei parametri geografici, in questo modo è possibile ottenere tweets in base alla posizione in cui sono stati generati. Questo campo presenta dei sottocampi, che sono:
  - Place: campo in cui è possibile specificare il nome di città. Il nome della località deve essere compreso tra doppi apici, ad esempio: “New York City”;
  - Place\_country: campo in cui è possibile inserire il codice di un paese. I possibili valori ammessi in questo campo sono quelli indicati dall’ISO\_3166-1\_alpha-2 format, ad esempio: IT, FR;
  - Bounding\_box: campo in cui è possibile ritrovare i tweet in un’area geografica specificata da coordinate: sud, nord, ovest, est. È possibile indicare le coordinate nei sottocampi: west\_long, south\_lat, east\_long, north\_lat;
  - Point\_radius: campo in cui è possibile ritrovare i tweet in un’area geografica specificata da longitudine, latitudine e angolazione. È possibile indicare tali valori nei sottocampi appositi: longitude, latitude, radius.
- Filter\_retweet: questo campo indica se devono essere ritrovati anche i retweet. I possibili valori sono True o False;
- Filter\_images: campo che indica se i tweet ritrovati devono contenere un’immagine, ovvero quelli che contengono un URL riferito ad un’immagine. I possibili valori che questo campo può assumere sono True o False.

Si riporta un esempio di configurazione per la ricerca di tweets:

```
mongodb:
  url: mongodb://localhost:27017/
  database: meloni
  collection: Message
twitter:
  configuration:
    bearer_token:
      AAAAaaaaaaaaaaaaAPtPgEAAAAAoVlZ4I0szkcu4dL%2Bhqif%2F%2BF45Oo%
      3DJbvSo773bskLu1GexDv9Dq1HjuSjfSwfxgLdDXEdlPO5mKyE6G
    end_point: https://api.twitter.com/2/tweets/search/all

  search:
    keyword: (Giorgia Meloni)

    user:
      user_mentioned:

    hashtag:

    filter_images:

    lang: it

    context_annotations: False

    n_results: 4000000

    all_tweets: False

    time:
      start_time: "2023-03-01T07:20:50.52+00:00"
      end_time: "2023-03-31T07:20:50.52+00:00"

    geo:
      place:
        place_country:

      bounding_box:

      point_radius:
        # example value: 2.355128
        longitude:
          # 48.861118
        latitude:
          # 16km
        radius:

    filter_retweet: True
```

Inoltre, nel file di configurazione, è possibile indicare il database nel quale salvare i dati ritrovati. Come database viene utilizzato MongoDB, i dati salvati in questo tipo di database sono in formato BSON, ovvero una rappresentazione binaria di un documento JSON.

La scelta di MongoDB come database è dovuta dal fatto che l'API di Twitter utilizza il formato JSON per comunicare con applicazioni di terze parti. Pertanto, è possibile utilizzare qualsiasi linguaggio di programmazione con supporto JSON per lo sviluppo di applicazioni e Python è uno di questi.

Dato che i tweet sono in formato JSON, è possibile modificare il payload che si richiede tramite una richiesta, ovvero modificare i campi che si vogliono richiedere per ogni tweet. Una query, richiederà dunque che i tweet contengano i seguenti campi indicati:

```
self.__query['place.fields'] =
"contained_within,country,country_code,full_name,geo,id,name,place
_type"
self.__query['expansions'] =
'author_id,geo.place_id,referenced_tweets.id,referenced_tweets.id.
author_id,attachments.media_keys'

self.__query['tweet.fields'] =
'lang,referenced_tweets,public_metrics,entities,created_at,possibly_
sensitive,attachments'
self.__query['media.fields'] =
'duration_ms,height,media_key,preview_image_url,public_metrics,type,
url,width,alt_text'

self.__query['user.fields'] =
'username,location'
```

Dopo che la richiesta è stata inoltrata, il payload dei tweet ricevuti si presenta nel seguente modo:

```
{  
  'edit_history_tweet_ids': ['1612909826782253056'],  
  'text': 'https://t.co/JZuBMHWq3g',  
  'author_id': '1603345696760946693',  
  'created_at': '2023-01-10T20:30:43.000Z',  
  'entities': {  
    'urls': [{  
      'start': 0,  
      'end': 23,  
      'url': 'https://t.co/JZuBMHWq3g',  
      'expanded_url': 'https://twitter.com/DavideVolpicel9/status/1612909826782253056/photo/1'  
    }],  
    'display_url': 'pic.twitter.com/JZuBMHWq3g',  
    'media_key': '3_1612909821782523905'  
  }},  
  'attachments': {  
    'media_keys': ['3_1612909821782523905']},  
  'possibly_sensitive': False,  
  'public_metrics': {  
    'retweet_count': 0,  
    'reply_count': 0,  
    'like_count': 0,  
    'quote_count': 0,  
    'impression_count': 5},  
  'lang': 'zxx',  
  'id': '1612909826782253056'  
}
```

Dato che le informazioni contenute nel payload sono disposte in modo disordinato e non tutte vengono utilizzate, la struttura di ogni tweet viene riorganizzata, in modo tale che questi possano apparire in modo ordinato all'interno del database.

I tweet, all'interno del database, si presentano nel seguente modo:

```
_id: "1585169386788179968"
raw_text: "@GCerquetti Vero. Ha provato a rassicurare su tutto, ma sulla Resistenz..."
author_id: "1570534673394139137"
author_name: "Stefano Marzetti 🌎"
author_username: "StefanoMarzett1"
created_at: "2022-10-26T07:20:07.000Z"
lang: "it"
possibly_sensitive: false
referenced_tweets: Array
  0: Object
    id: "1585144728626290688"
    type: "replied_to"
twitter_entities: Object
hashtags: Array
  0: "MeloniPremier"
mentions: Array
  0: "GCerquetti"
  1: "GiorgiaMeloni"
geo: Object
  user_location: "Desenzano del Garda, Lombardia"
  coordinates: Object
  latitude: 45.4694851
  longitude: 10.5389467
metrics: Object
  retweet_count: 0
  reply_count: 0
  like_count: 1
  quote_count: 0
```

## 2.2.2 Elaborazione dei tweets

Una volta che i tweets sono stati salvati all'interno del database, è possibile elaborarli in diversi modi. ITA, infatti, consente di effettuare:

- Sentiment analysis;
- Natural Language Processing;
- Entity Linking;
- Geocoding.

### 2.2.2.1 Sentiment Analysis

ITA integra al suo interno due modelli in grado di effettuare sentiment analysis: SENTIPOLC e Feel-it

#### 2.2.2.1.1 SENTIPOLC

SENTIPOLC [8], utilizzato da ITA sotto il nome di Sent-it, è un modello realizzato dall'università di Bari, in grado di effettuare sentiment analysis.

Esso è in grado di classificare il testo che riceve in input in tre polarità diverse, che sono: positivo, negativo e neutrale.

Per poter utilizzare questo strumento, è necessario collegarsi al server in cui il modello si trova mediante l'apposito url: "<http://193.204.187.210:9009/sentipolc/v1/classify>".

## Uso di SENTIPOLC all'interno di ITA:

```
def __sent_it_analyze_sentiment(self, tweet_text: str, tweet: dict) -> Tuple[int, dict, dict]:\n\n    # build the request to send to ths server\n    data = "{\"texts\": [{\"id\": \"1\", \"text\": \"\"}]}\"\n    # remove special charachters from the teof the tweet before\n        send it\n    data += tweet_text.replace("\n", "").replace("\\", "\\\")\n        .replace("\r", "") + "\"]}"\n    # set the url\n    url = "http://193.204.187.210:9009/sentipolc/v1/classify"\n    # send the request and save the response in json\n    try:\n        json_response = requests.post(url, data=data.encode('utf-\n            8')).json()\n        # if there is a result in the response\n        if 'results' in json_response:\n            # save the result (for consistence the result is\n                normalized)\n            if json_response['results'][0]['polarity'] == "neg":\n                polarity = "negative"\n            elif json_response['results'][0]['polarity'] == "pos":\n                polarity = "positive"\n            else:\n                polarity = "neutral"\n            sentiment_analyses = {'subjectivity':\n                json_response['results'][0]['subjectivity'],\n                'sentiment': polarity}\n            # return the id of the process (2), the result of the\n                analyses, and the original tweet\n            return 2, sentiment_analyses, tweet\n        else:\n            return 2, {}, tweet\n    except requests.exceptions.ConnectionError or\n        urllib3.exceptions.MaxRetryError \\ \n        or urllib3.exceptions.NewConnectionError or\n        ConnectionRefusedError as e:\n        self.log.warning(\n            "\nSENT-IT PHASE:IMPOSSIBLE TO ESTABLISH CONNECTION\nWITH SENT-IT SERVICE. WAITING AND RETRYING.")\n        time.sleep(2)\n        return\n    self.__sent_it_analyze_sentiment(tweet_text=tweet_text,\n        tweet=tweet)
```

Dopo che l'etichetta viene generata dal modello, questa viene salvata nel database, andando quindi ad aggiungere un nuovo campo, in questo caso “sent-it”.

```
_id: "1585169257935306754"
raw_text: "Che eleganza Donna Giorgia 😊 #fiducia #Conte
#MeloniPremier #Meloni ht.."
author_id: "1142782223193653249"
author_name: "⭐️❤️⭐️⭐️⭐️ Anny ⭐️⭐️❤️⭐️"
author_username: "leonessa1864"
created_at: "2022-10-26T07:19:36.000Z"
lang: "it"
possibly_sensitive: false
twitter_entities: Object
metrics: Object
processed: true
tags: Object
sentiment: Object
    sent-it: Object
    sentiment: "positive"
spacy: Object
```

#### 2.2.2.1.2 Feel-it

Feel-it [9] è un modello realizzato da Federico Bianchi, un ricercatore dell'università di Standford, ed è in grado classificare il testo nelle quattro emozioni principali, che sono: rabbia, paura, gioia e tristezza. Inoltre, questo modello è anche in grado di effettuare sentiment analysis; dunque, anch'esso è in grado di indicare se un testo risulta essere positivo, negativo o neutrale.

Nella seguente tabella, si riportano alcuni esempi di classificazione effettuati con Feel-it:

Text	Emotion
Pagliacci ammaestrati dal Grillo parlante di Pinocchio	anger
<i>They are buffoons controlled by Pinocchio's Jiminy Cricket</i>	
Non ci sto dormendo la notte. #22Agosto #COVID19	fear
<i>This does not make me sleep at night. #22August #COVID19</i>	
Adoro questa canzone, è una delle mie preferite STREAM	joy
ICARUS FALLS	
<i>I love this song, it's one of my favourite STREAM ICARUS FALLS</i>	
I brividi. Come si può spegnere una vita con così tanta facilità?	sadness
Non ho parole....	
<i>I got chills. How can you kill someone so easily? I do not know what to say....</i>	

Per poter utilizzare Feel-it, dato che questo è stato caricato su Pypi, è necessario solamente importare la sua corrispettiva libreria: "feel-t".

### Uso di Feel-it all'interno di ITA:

```

        self.sentiment_classifier.predict(hold_list)[0]}
    except:
        # if the model is busy wait and retry
        time.sleep(0.01)
        return self.__feel_it_analyze_sentiment(tweet)
    # return the process id (3), the result and the tweet
    return 3, sentiment_analyses, tweet

```

Le etichette assegnate al testo del tweet vengono successivamente salvate all'interno del database:

```

_id: "1585169257935306754"
raw_text: "Che eleganza Donna Giorgia кнопк #fiducia #Conte
#MeloniPremier #Meloni ht..."
author_id: "1142782223193653249"
author_name: "O•♥•✿•✿•✿ Anny✿•✿•✿"
author_username: "leonessa1864"
created_at: "2022-10-26T07:19:36.000Z"
lang: "it"
possibly_sensitive: false
twitter_entities: Object
metrics: Object
processed: true
tags: Object
sentiment: Object
    feel-it: Object
    emotion: "joy"
    sentiment: "positive"
spacy: Object

```

### 2.2.2.2 Natural language processing

Oltre alla sentiment analysis, ITA consente anche di effettuare l'elaborazione del linguaggio naturale tramite l'utilizzo della libreria SpaCy [10]. Quest'ultima è una libreria di python open source utilizzata nell'elaborazione avanzata del linguaggio naturale e nell'apprendimento automatico.

Le funzionalità di SpaCy che vengono utilizzate sono:

- Part-of-speech Tagging: processo di marcatura di una parola nel testo a una particolare parte del discorso in base al contesto e alla definizione. In altre parole, il tagging POS è il processo di identificazione di una parola come nomi, pronomi, verbi, aggettivi, ecc.
- Naming entities: in un testo possono essere presenti diversi tipi di entità, esempi di entità sono: una persona, un paese. SpaCy consente di riconoscere diversi tipi di entità in un testo effettuando dunque una predizione.

Tuttavia, prima di elaborare il testo è necessario effettuare delle operazioni utili a semplificarlo. È necessario dunque effettuare una fase di preprocessing, in questo modo i modelli che lavorano sul testo saranno agevolati in quanto lavoreranno con del testo “pulito”. I passi che vengono effettuati nella fase di preprocessing sono:

- Tokenization: processo che consiste nel suddividere un testo in piccole unità chiamate token. Un token può essere una parola, parte di una parola o solo caratteri come la punteggiatura;
- Lemmatizzazione: processo che consiste nella riduzione di una forma flessa di una parola alla sua forma canonica, detta lemma;
- Rimozione di stopword: processo che consiste la rimozione di tutti quei termini privi di significato semantico dal testo, ad esempio gli articoli.

## Utilizzo di SpaCy in ITA:

```
def __process_text_with_spacy(self, text_tweet: str, tweet: dict) -> Tuple[int, dict, dict]:  
    """  
        This method perform the natural language processing on the tweet text using spacy.  
    :param text_tweet: the text of the tweet  
    :type text_tweet: str  
    :param tweet: a dict representing the tweet  
    :type tweet: dict  
    :return: the id of the process: 4; a dictionary that contains the result of the analysis; the tweet analyzed  
    :rtype: Tuple[int, dict, dict]  
    """  
  
    # perform the analysis of the text with spacy  
    doc = self.nlp_module(text_tweet)  
    customize_stop_words = [  
        'no',  
        'non',  
        'Non',  
        'No'  
    ]  
    # customize the stopwords  
    for w in customize_stop_words:  
        self.nlp_module.vocab[w].is_stop = False  
  
    lemmas_with_postag = []  
    filtered_sentence = []  
    # remove space, punct and stopwords from the tweet text  
    for word in doc:  
        lexeme = self.nlp_module.vocab[word.lemma_]  
        if not lexeme.is_stop and not lexeme.is_space and not lexeme.is_punct:  
            filtered_sentence.append(word)  
  
    # for each token remained save the lemma, the pos information  
    # adn the morphology  
    for token in filtered_sentence:  
        lemmas_with_postag.append(  
            token.lemma_ + " POS : " + token.pos_ + " ; MORPH : "  
            + token.morph.__str__().replace("|", "-"))  
  
    entities = []  
    # retrieve the entities recognized from spacy  
    for ent in doc.ents:  
        entities.append(ent.text + " : " + ent.label_)  
    return 4, {'processed_text': lemmas_with_postag, 'entities': entities}, tweet
```

I dati ottenuti dall'elaborazione del testo vengono salvati all'interno del database sotto l'apposito campo "spacy". Si riporta il testo presente in un tweet ed i dati generati in seguito alla loro elaborazione.

"@GCerquetti Vero. Ha provato a rassicurare su tutto, ma sulla Resistenza non ce l'ha fatta. Pensa i suoi tantissimi elettori neofascionaziskin che le hanno sentito dire: \"... mai provato simpatia per i regimi antidemocratici, fascismo compreso\"? Apriticielo!\n#MeloniPremier @GiorgiaMeloni"

```
spacy: Object
  processed_text: Array
    0: "@chiadegli POS : ADP ; MORPH : Definite=Def-
      Number=Sing-PronType=Art"
    1: "@fiammafrancesca POS : PROPN ; MORPH : "
    2: "accordo POS : NOUN ; MORPH : Gender=Masc-
      Number=Sing"
    3: "serracchiani POS : PROPN ; MORPH : "
    4: "discorso POS : NOUN ; MORPH : Gender=Masc-
      Number=Sing"
    5: "riguardare POS : VERB ; MORPH : Mood=Ind-
      Number=Sing-Person=3-Tense=Im..."
    6: "politica POS : NOUN ; MORPH : Gender=Fem-
      Number=Plur"
    7: "donna POS : NOUN ; MORPH : Gender=Fem-Number=Plur"
    8: "società POS : NOUN ; MORPH : Gender=Fem"
    9: "risposta POS : NOUN ; MORPH : Gender=Fem-
      Number=Sing"
    10: "meloni POS : PROPN ; MORPH : "
    11: "riguardare POS : VERB ; MORPH : Gender=Masc-
      Number=Sing-Tense=Past-Ver..."
    12: "singolo POS : ADJ ; MORPH : Gender=Fem-
      Number=Sing"
    13: "vicenda POS : NOUN ; MORPH : Gender=Fem-
      Number=Sing"
    14: "maschio POS : NOUN ; MORPH : Gender=Masc-
      Number=Sing"
    15: "vincente POS : ADJ ; MORPH : Number=Sing"
    16: "bisogno POS : NOUN ; MORPH : Gender=Masc-
      Number=Sing"
    17: "uscita POS : NOUN ; MORPH : Gender=Fem-
      Number=Plur"
```

```
18: "collettivo POS : ADJ ; MORPH : Gender=Fem-
Number=Plur"
19: "condizionamento POS : NOUN ; MORPH : Gender=Masc-
Number=Plur"
20: "non POS : ADV ; MORPH : PronType=Neg"
21: "assolo POS : NOUN ; MORPH : Gender=Masc-
Number=Plur"
22: "bullismo POS : NOUN ; MORPH : Gender=Masc-
Number=Sing"
23: "piacere POS : VERB ; MORPH : Mood=Ind-Number=Sing-
Person=3-Tense=Pres-..."
entities: Array
0: "Serracchiani : PER"
1: "Meloni : PER"
```

### 2.2.2.3 Entity Linking

ITA consente anche di effettuare entity linking. Con entity linking si intende quella funzionalità che permette di assegnare un'identità univoca a entità (come personaggi famosi, luoghi o aziende) menzionate in un testo. Ad esempio, data la frase “Washington è la capitale degli stati Uniti”, l’idea è determinare che “Washington” si riferisce alla città di Washington e non a George Washington o qualsiasi altra entità che potrebbe essere indicata come “Washington”.

Per effettuare entity linking, ITA utilizza la libreria TagMe [11]. Questa API consente dunque di ritrovare le entità che sono presenti all’interno dei tweet ed inoltre le collega alla loro corrispettiva pagina di Wikipedia.

## Utilizzo di TagMe all'interno di ITA:

```
def __process_text_with_spacy(self, text_tweet: str, tweet: dict) -> Tuple[int, dict, dict]:  
    """  
        This method perform the natural language processing on the  
        tweet text using spacy.  
    :param text_tweet: the text of the tweet  
    :type text_tweet: str  
    :param tweet: a dict representing the tweet  
    :type tweet: dict  
    :return: the id of the process: 4; a dictionary that contains  
            the result of the analysis; the tweet analyzed  
    :rtype: Tuple[int, dict, dict]  
    """  
  
    # perform the analysis of the text with spacy  
    doc = self.nlp_module(text_tweet)  
    customize_stop_words = [  
        'no',  
        'non',  
        'Non',  
        'No'  
    ]  
    # customize the stopwords  
    for w in customize_stop_words:  
        self.nlp_module.vocab[w].is_stop = False  
  
    lemmas_with_postag = []  
    filtered_sentence = []  
    # remove space, punct and stopwords from the tweet text  
    for word in doc:  
        lexeme = self.nlp_module.vocab[word.lemma_]  
        if not lexeme.is_stop and not lexeme.is_space and not  
            lexeme.is_punct:  
            filtered_sentence.append(word)  
  
    # for each token remained save the lemma, the pos information  
    # and the morphology  
    for token in filtered_sentence:  
        lemmas_with_postag.append(  
            token.lemma_ + " POS : " + token.pos_ + " ; MORPH : "  
            + token.morph.__str__().replace("|", "-"))  
  
    entities = []  
    # retrieve the entities recognized from spacy  
    for ent in doc.ents:  
        entities.append(ent.text + " : " + ent.label_)  
    return 4, {'processed_text': lemmas_with_postag, 'entities': entities}, tweet
```

Come esempio, si riporta il testo di un tweet e il corrispettivo risultato ottenuto con TagMe:

"@GCerquetti Vero. Ha provato a rassicurare su tutto, ma sulla Resistenza non ce l'ha fatta. Pensa i suoi tantissimi elettori neofascionaziskin che le hanno sentito dire: \"... mai provato simpatia per i regimi antidemocratici, fascismo compreso\"? Apriticielo!\n#MeloniPremier @GiorgiaMeloni"

```
tags: Object
tag_me: Array
0: "fascismo:2860249:Fascismo:
https://it.wikipedia.org/wiki/Fascismo"
```

#### 2.2.2.4 Geocoding

ITA consente di effettuare la geolocalizzazione dei tweet presenti all'interno del database, utilizzando il servizio Open Street Map. Se i tweet salvati contengono nel payload anche la località di origine, allora tramite OSM è possibile ricavare le coordinate del luogo, ovvero longitudine e latitudine.

Open Street Map (OSM) [12] è il più grande database geografico libero e modificabile di tutto il mondo, costruito dal lavoro di volontari e rilasciato con una licenza libera. Si tratta di un gigantesco progetto collaborativo, con milioni di utenti registrati in tutto il mondo, il cui scopo è creare e fornire dati geografici liberi a chiunque li voglia utilizzare.

Se i tweet salvati contengono nel payload anche la località di origine, allora tramite OSM è possibile ricavare le coordinate del luogo, ovvero longitudine e latitudine. Per poter utilizzare OSM all'interno di ITA, si è utilizzata la libreria geocoder [13].

```

def __get_osm_coordinates(self, tweet: dict, user_location: Optional[str], city: Optional[str], country: Optional[str]) \
    -> Tuple[int, bool, dict, dict]:
    """
    This method use the Open Street Map service to find the coordinates of a specific location.

    :param tweet: the tweet analyzed
    :type tweet: dict
    :param user_location: the user_location if the tweet contains it
    :type user_location: str, optional
    :param city: the city from where the tweet has been posted if the tweet contains it
    :type city: str, optional
    :param country: the country from where the tweet has been posted if the tweet contains it
    :type country: str, optional
    :return: the id of the process: 5; a dictionary that contains the result of the analysis; the tweet analyzed
    :rtype: Tuple[int, bool, dict, dict]
    """

    g = None
    # build the dict to send as request to the osm service with the information given
    try:
        if user_location is not None:
            g = geocoder.osm(user_location)
        elif city is not None and country is not None:
            g = geocoder.osm(city + "," + country)
    except ValueError:
        return 5, False, {}, tweet
    except requests.exceptions.ConnectionError or \
        requests.exceptions.ReadTimeout or \
        requests.exceptions.Timeout:
        time.sleep(0.5)
        self.log.warning("GEO PHASE: ERROR DURING THE CONNECTION.\nRETRYING.")
        return self.__get_osm_coordinates(tweet, user_location, city, country)
    # return the coordinates if the result of the request is ok
    if g.ok:
        return 5, True, {'latitude': g.osm['y'], 'longitude': g.osm['x']}, tweet
    # return an empty dict if the result is not ok
    else:
        return 5, False, {}, tweet

```

Un tweet elaborato con il servizio OSM presenterà il campo “geo” e al suo interno le coordinate.

```
geo: Object
  user_location: "Desenzano del Garda, Lombardia"
  coordinates: Object
    latitude: 45.4694851
    longitude: 10.5389467
```

## 2.2.2.5 File di configurazione per l’elaborazione dei tweets

Su ITA è possibile decidere quali tra i modelli per l’elaborazione dei dati utilizzare sui tweets raccolti, andando a modificare il file di configurazione “process\_tweets.config”. In questo file si trovano dei campi da avvalorare, la cui modifica comporta quali modelli verranno eseguiti durante l’elaborazione. I campi sono i seguenti:

- TagMe: campo che comprende altri sottocampi, con i quali si settano le impostazioni per utilizzare il servizio TagMe.
  - Enabled: abilita o disabilita il modello. I possibili valori sono True o False;
  - Token: in questa campo si inserisce l’indirizzo a cui inviare la richiesta per poter utilizzare TagMe;
  - Is\_tweet: indica al servizio TagMe se il testo che gli si sta dando in input è un tweet o meno. I possibili valori sono True o False;
  - Rho\_value: parametro che indica una stima nella confidenza dell’annotazione. I possibili valori sono numeri che vanno da 0 a 1.

- **Sentiment\_analyze**: campo nel quale l’utente può decidere quali tipi di modelli utilizzare per la sentiment analysis. I suoi sottocampi sono:
  - **Sent-it**: True/False;
  - **Feel-it**: True/False;
- **Geocoding**: campo che permette di abilitare o meno la fase di geocoding. I possibili valori sono True o False;
- **Nlp**: campo che abilita o disattiva il servizio SpaCy;
- **Analyze\_all\_tweets**: con questo campo l’utente può decidere se elaborare solo i tweet processati con NLP (False), oppure se elaborarli tutti (True).

Inoltre, in questo file è possibile indicare la collezione di tweets presente nel database su cui effettuare l’elaborazione. Un esempio di configurazione di tale file è il seguente:

```
mongodb:
  url: mongodb://localhost:27017/
  database: meloni
  collection: Message
analyzes:
  nlp: False
  tagme:
    enabled: False
    token: 7f5391f2-142e-4fd5-9cc9-56e91c4c9add-843339462
    is_tweet: False
    rho_value: 0.15
  sentiment_analyze:
    sent_it: False
    feel_it: True
  geocoding: False
  analyze_all_tweets: False
```

### 2.2.3 Ritrovamento di informazioni sugli utenti

ITA consente di ritrovare gli id degli utenti i cui tweets sono già stati salvati all'interno del database. Verrà quindi creata una nuova collezione dati in cui saranno presenti gli id degli utenti. Tramite il file di configurazione “search\_users.config”, è possibile impostare da quale database prelevare i tweets e in quale database salvare la lista degli id.

Si riporta un esempio di configurazione del file:

```
mongodb_tweets:
  url: mongodb://localhost:27017/
  database: htm_tweets
  collection: tweets
mongodb_users:
  url: mongodb://localhost:27017/
  database: htm_users
  collection: users

twitter:
  configuration:
    bearer_token:
      AAAAaaaaaaaaaaaaAPtPgAAAAAAoVlZ4I0szkcu4dL%2Bhqif%2F%2BF450o%
      3DJbvSo773bskLu1GexDv9Dq1HjuSjfSwfxgLdDXEdlPO5mKyE6G
    end_point: https://api.twitter.com/2/users
```

Effettuando questo tipo di elaborazione sui tweets pubblicati da Giorgia Meloni, si ottiene il seguente risultato:

```
_id: "130537001"
name: "Giorgia Meloni"
username: "GiorgiaMeloni"
public_metrics: Object
  followers_count: 1736691
  following_count: 240
  tweet_count: 19992
  listed_count: 3164
location: "Italia"
```

## 2.2.4 Gestione dei tweets

Un ulteriore funzionalità di ITA è quella che consente all'utente di estrarre o eliminare i tweets presenti all'interno del database.

Modificando il file di configurazione “manage\_tweets.config”, è possibile indicare i tweets che si vogliono estrarre o eliminare, impostando i seguenti criteri:

- Sentiment: è possibile indicare quali tweets gestire in base alla polarità del sentiment. I possibili valori sono: positive, neutral or negative.
- Keywords: impostando questo campo, è possibile ritrovare i tweets che contengono parole specifiche. È possibile effettuare l'elaborazione con parole chiavi in due modi:
  - Indicando una lista di parole nel sottocampo “words”, separate da virgole;
  - Utilizzando un file in cui sono scritte le parole, in questo caso si utilizzerà il sottocampo “path”, dove si dovrà indicare il percorso al file di testo.
- Postag: impostando questo campo è possibile ritrovare i tweets che contengono parole con uno specifico POS tag.
- Morph: impostando questo campo è possibile ritrovare i tweets che contengono parole con una specifica morfologia.
- Raw\_query: in questo campo, è possibile inserire una query per poter ritrovare i tweet presenti nel database. La query deve essere scritta rispettando la sintassi di MongoDB.
- Logical\_operator: se più di un criterio elencato viene avvalorato, è necessario definire come questi sono logicamente collegati. Con questo campo è quindi possibile impostarlo; i possibili valori sono OR o AND.

## 2.3 CrowdPulse

CrowdPulse è una piattaforma realizzata con lo scopo di visualizzare i tweet raccolti ed elaborati mediante ITA. La piattaforma è suddivisa in due blocchi principali:

- Il blocco “frontend” che contiene tutte le componenti, con relative dipendenze, che si occupano della gestione delle interfacce e gestione della parametrizzazione delle query.
- Il blocco “backend”, che presenta tutte le componenti, che si occupano di gestire sia le routes della piattaforma che la connessione con il database MongoDB.

### 2.3.1 Architettura della piattaforma

CrowdPulse è stato realizzato seguendo l’architettura detta “Mern Stack”, dove il termine “Mern” deriva dagli strumenti che vengono utilizzati per realizzare il software, che sono:

- MongoDB: database utilizzato dall’applicazione;
- Express(.js): framework web Node.js;
- React(.js): un framework client-side di Javascript;
- Node(.js): web server di Javascript.

# MERN Stack Architecture

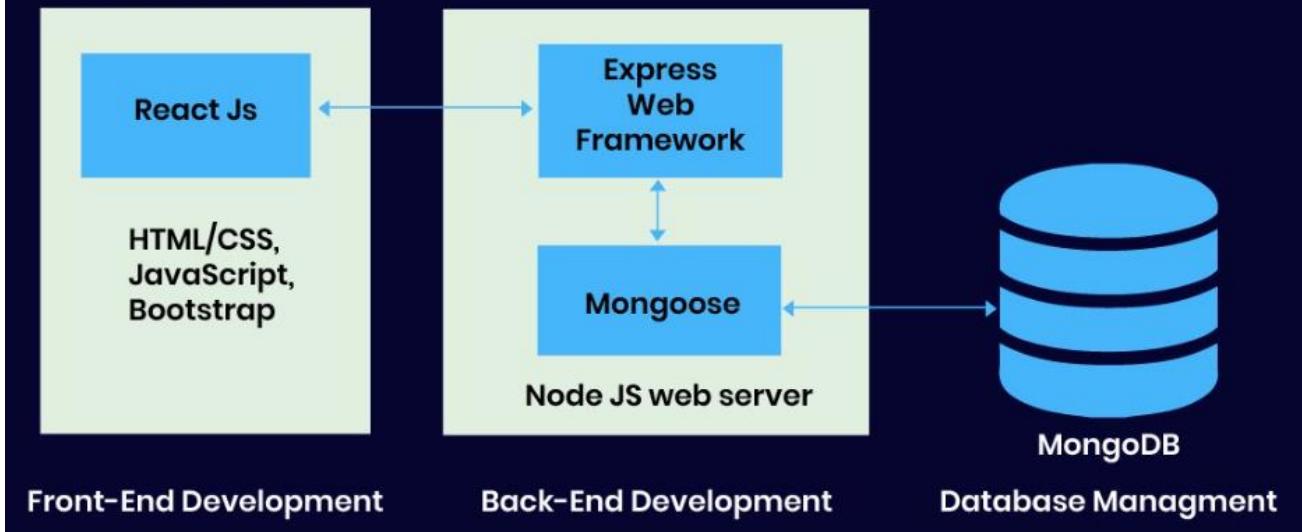


Figura 7 - Architettura MERN

### 2.3.1.1 React.js front end

React.js [14] è un framework di Javascript per la creazione di applicazioni dinamiche lato client in HTML. Esso consente di creare interfacce complesse attraverso componenti semplici, collegarle ai dati sul server di back-end e renderle in forma HTML.

### 2.3.1.2 Express.js e Node.js server

Nel lato backend dell'applicazione si utilizza Express.js [15], il quale viene eseguito tramite Node.js [16]. Express.js è stato scelto per la sua capacità di gestire velocemente le richieste e risposte HTTP.

### 2.3.1.3 MongoDB

MongoDB [17] è un Sistema di gestione di database non relazionale e orientato ai documenti. Esso è considerato un database non standard, infatti viene interpretato come NoSql per la sua struttura. In particolare, MongoDB differisce dalle sue controparti Sql allontanandosi da una tradizionale struttura basata su tabelle a favore di un'implementazione di documenti in formato Json.

### 2.3.2 Backend

L'architettura “Backend” della piattaforma è gestita principalmente da tre componenti:

- Server.js: script principale per la gestione del server che si occupa sia di reindirizzare le richieste http per le varie routes.
- Routes: componente che si occupa sia di connettere la piattaforma al database MongoDb, utilizzando le credenziali presenti nel file “.env”. Inoltre, si occupa di effettuare varie operazioni in base alla richiesta arrivata dal “server.js”. In particolare, durante la fase di “PreLoad” dei dati verranno effettuate cinque richieste “GET” ovvero:
  - “/tweet/getDataSortByDate”: richiesta che permette di estrarre tutte i tweet analizzati. Questi si presenteranno ordinati in maniera crescente;
  - “/tweet/getHashtags”: richiesta che si occupa di estrarre tutti gli hashtags dei tweets presenti nel database selezionato;

- “/tweet/getText”: richiesta che si occupa di estrarre tutte le parole presenti nella base di dati selezionata;
- “/tweet/getTags”: richiesta che si occupa di estrarre tutti i tags presenti nei tweets del database selezionato;
- “/tweet/getUsers”: richiesta che si occupa di estrarre tutti gli username presenti nella base di dati selezionata.

Le operazioni effettuate dalla parte di backend possono essere riassunte in quattro fasi principali:

- Avvio del server effettuato tramite lo script “Server.js”, il quale si occupa di mettere in ascolto il server sulla porta 4000;
- Inizializzazione delle Routes, che viene effettuata dal file Server.js che importa il file routes.js all’interno del software;
- Estrazione dei database: dopo aver importato il file Routes.js, verranno inizializzate le connessioni al database MongoDB. La prima connessione viene effettuata come “admin” per estrarre i DB presenti, mentre la seconda come User e viene utilizzata per l’estrazione dei dati. Le chiavi d’accesso sono rese private grazie all’utilizzo della libreria “dotenv” e in particolare del file “.env”
- Preloading dei dati: dopo che l’utente seleziona il database desiderato, verranno estratti tutti i tweets necessari.

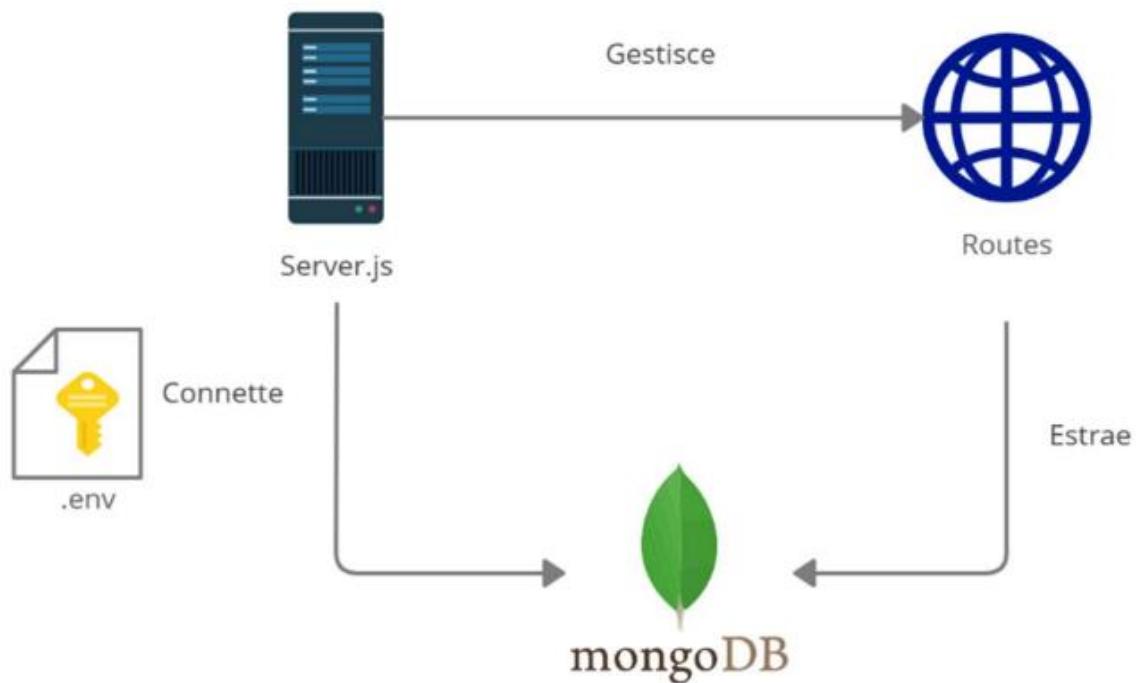
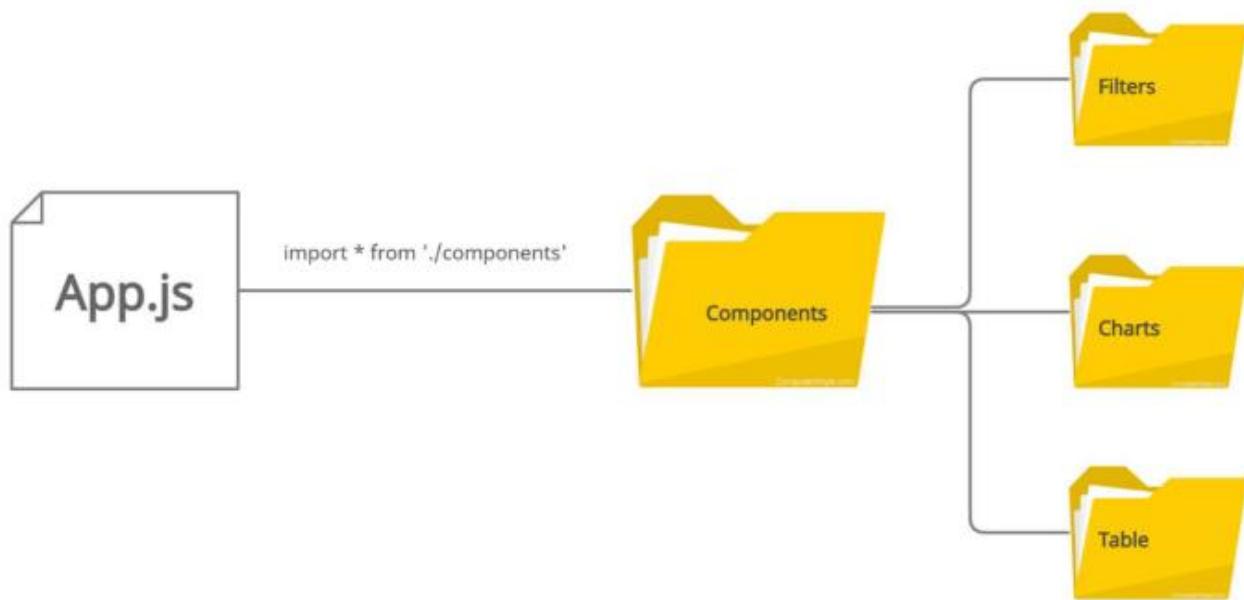


Figura 8 - Logica lato Backend

### 2.3.3 Frontend

L'architettura frontend prevede l'utilizzo di una componente principale, ovvero **App.js** nella quale vengono integrati i vari moduli per la gestione delle diverse funzionalità. Questo modo di gestire gli script è detto “core-plugin”, in quanto è possibile gestire facilmente tutte le future aggiunte o rimozioni di plugin, ovvero i moduli.



*Figura 9 - Struttura della parte frontend di CrowdPulse*

Dalla figura si evince che lo script App.js utilizza i contenuti della cartella components, che sono: filters, charts e table.

### 2.3.3.1 Filters

Il package “Filters” contiene tutte le componenti che si occupano sia di fornire le opzioni di filtraggio che di gestire e aggiornare i dati. In particolare, la cartella conterrà sette componenti diverse:

- Filters.js: componente generica che si occupa di gestire le funzionalità di filtraggio delle sezioni “Maps” e “TweetList”. I filtri presenti in questo script permettono all’utente di selezionare i tweets in base a: hashtags, tags, data di pubblicazione e username. Inoltre, consente di estrarre i tweet anche in base all’elaborazione effettuata su di essi; dunque, si potrà effettuare l’estrazione dei dati andando a selezionare un algoritmo di sentiment analysis e il suo corrispettivo label;

- SentimentFilters.js: estensione della più generica “Filters.js”. I filtri sviluppati in questo script vengono utilizzati nella sezione “Sentiment”;
- TimeLinesFilters.js: estensione della più generica “Filters.js”. I filtri sviluppati in questo script vengono utilizzati nella sezione “TimeLines”;
- WordChartFilters.js: consiste in un'estensione generale dello script “Filters.js” e si occupa di gestire la sezione “WordCloud”;
- SearchFilters.js: componente che permette di gestire l'inserimento e auto completamento dei tags inseriti dall'utente nel filtro dedicati a questi ultimi;
- SearchHashtag.js: componente che permette di gestire l'inserimento e autocompletamento degli hashtags inseriti dall'utente nel filtro dedicati a questi ultimi;
- SearchUser.js: componente che si occupa di gestire l'inserimento e l'auto completamento degli utenti nel filtro dedicato a quest'ultimi;
- SearchText.js: componente che si occupa di gestire l'inserimento del testo nel filtro “processed text”.

### 2.3.3.2 Charts

Il package “Charts” contiene tutti i moduli che si occupano della gestione e visualizzazione dei vari grafici. Tutti i componenti presenti in questa cartella sono stati creati utilizzando la libreria “Chart.js”. I componenti sono:

- BarChart.js: componente che gestisce il grafico a barre della sezione “SentimentCharts”;
- TimelineChart.js: componente che gestisce il grafico temporale presente nella sezione “TimeLines”;

- MultiTimeLineChart.js: componente che gestisce i grafici temporali multilinea presenti nella sezione “SentimentCharts”;
- PieChart.js: componente che si occupa di gestire e visualizzare il grafico a torta presente nella sezione “SentimentCharts”;
- Radar.js: ovvero la componente che si occupa di gestire e visualizzare il grafico a radar presente nella sezione “SentimentCharts”.

### 2.3.3.3 Table

Il package “Table” contiene tutti i moduli necessari per gestire e visualizzare tutti i tweet in formato tabulare. In particolare, la funzionalità della lista dei tweets è gestita dalla componente “TweetsTable.js”. Nella cartella in questione, sono presenti due file:

- Index.jsx: file che stampa tutte le righe della tabella combinando le varie proprietà della componente “TweetsTable.js”;
- Footer/Index.jsx: file che si occupa di effettuare il render della tabella e di gestirne delle funzionalità.

### 2.3.4 Funzionalità principali di CrowdPulse

Nella schermata principale dell’applicazione, realizzata nello script App.js, è possibile selezionare diverse funzionalità. Prima di poter selezionare una funzionalità, è necessario che l’utente selezioni un database.

In ogni schermata, sono presenti dei filtri che consentono all’utente di gestire quali tweets considerare. I filtri in questione sono i seguenti:

- Algorithm: filtro dove è possibile selezionare un algoritmo di sentiment analysis;
- Sentiment: filtro dove è possibile selezionare un label dell'algoritmo selezionato;
- From/To: filtri dove è possibile indicare un range temporale. Verranno ritrovati quei tweets che sono stati pubblicati nel range indicato dall'utente;
- Tags: filtro che permette di ricercare i tweets tramite tags;
- Processed Text: filtro che permette di ricercare i tweets attraverso parole chiave: verranno ritrovati quei tweets che le contengono;
- Hashtags: filtro che permette di ricercare i tweets tramite hashtags;
- Username: filtro in cui è possibile inserire l'username di un utente Twitter, verranno ritrovati tutti quei tweets presenti nel database che sono stati pubblicati dall'user indicato.

Algorithm Feel-it	Genre All	From gg/mm/aaaa	To gg/mm/aaaa	Total Tweets <b>27163</b>
Tags Add new tag	Processed Text Add new Text	Hashtags Add new Hashtag	Username Add new Username	

*Figura 10 - filtri presenti nelle schermate principali*

### 2.3.4.1 Sentiment

Schermata in cui sono presenti i filtri che consentono all'utente di selezionare i dati da visualizzare. In base agli algoritmi scelti, verranno mostrati i corrispettivi grafici.

*Figura 11 - Grafico a barre di Feel-it*

### 2.3.4.2 Word

Schermata in cui viene mostrata la word cloud generata dai tweet presi in considerazione mediante filtri.



*Figura 12- Word cloud generata da tweet inerenti a Giorgia Meloni*

### 2.3.4.3 TimeLines

Schermata in cui viene mostrato l'arco temporale della pubblicazione dei tweet ritrovati.

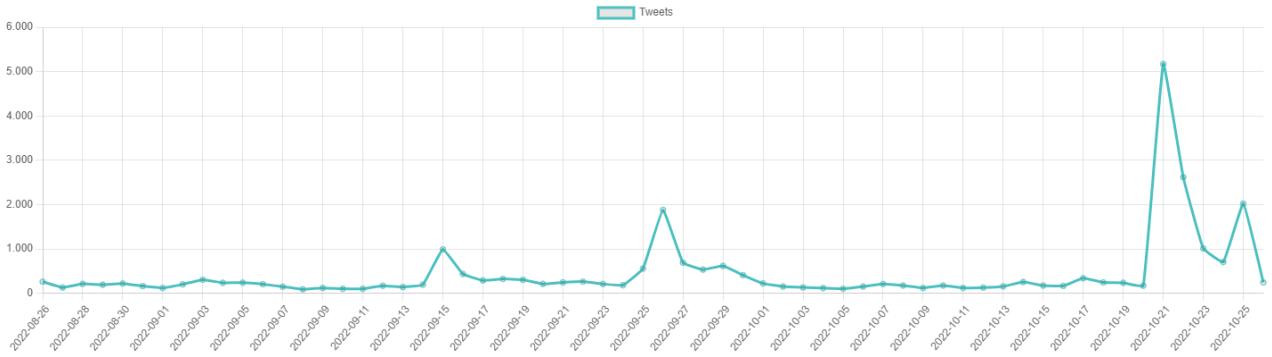


Figura 13- Grafico presente nella sezione Timelines

### 2.3.4.4 TweetList

Schermata in cui viene mostrata una tabella di tutti i tweet ritrovati mediante i filtri, per ogni tweet viene mostrato: l'username di chi ha pubblicato il tweet, il testo ed eventuali tags.

USERNAME	TEXT	TAGS
viviLaGatta	#salariominimo #diritti #clima Che sollievo ascoltare parole di sinistra e leggere proposte d i sinistra in mezzo a questa zuppa di centristi di tutte le sfumature #civatiètomato ❤️ http://t.co/PvyqfQArnv	sinistra centristi
Fede_Valcauda	Come @Radicali sono anni che segnaliamo il pericolo culturale di avere Meloni al governo, e la questione #aborto, che per loro è questione ideologica e di forma della società, è l'emblema della pericolosità della restrizione della libertà delle donne.	aborto ideologica società
massimartinelli	Influencer contro/ Donne vs donne, il nuovo duello nell'era Meloni https://t.co/iKDnYGze3B @ilmessaggero @MariaLatella	
LucillaMasini	Fratelli d'Italia. Eugenia Roccella: «L'aborto non è un diritto, sono femminista». Sì, certo, e la Meloni è comunista. #Meloni #aborto #fascismo #Roccella #Fratelliitalia #comunismo #elezioni #aborto #femminismo #26agosto	Fratelli d'Italia Italia Eugenia Roccella aborto femminista fascismo comunismo aborto femminismo
tonyo1969	Qui sopra ormai non si fa altro che parlare della Meloni, mi sarei anche stufato....non è un a persona all'altezza di guidare alcunché...è fascista, è misogina, è razzista, è omofoba, è contro l'aborto, contro i diritti lgbtq+, è contro l'euro e l'Europa, ha una concezione patriarciale	razzista omofoba aborto diritti lgbtq Europa

Figura 14 - Tabella dei tweets presente nella sezione TweetsList

## 2.3.4.5 Maps

Schermata in cui i tweet ritrovati vengono indicati sulla mappa italiana. Le zone in cui i colori sono più caldi, indicano che in quel punto sono stati pubblicati molti tweet del tema ricercato.

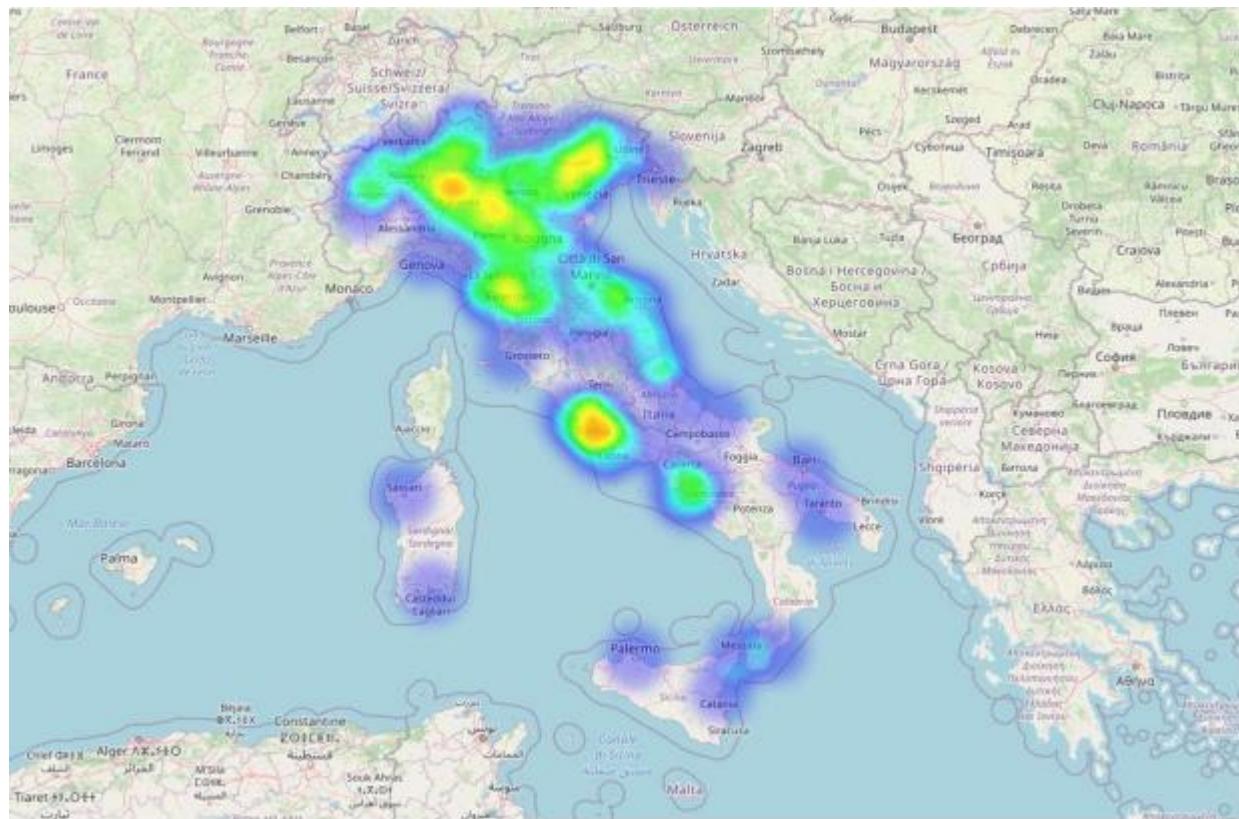


Figura 15 – Rappresentazione su mappa dell'intensità dei tweets pubblicati su Giorgia Meloni

## Capitolo3: Integrazione di nuove funzionalità

Su Italian Tweets Analyzer sono stati introdotti nuovi modelli in grado di effettuare diversi tipi di classificazioni sui tweets ritrovati, e non solo.

Tali modelli sono stati individuati dalla piattaforma HuggingFace.

### 3.1 HuggingFace

HuggingFace [18] è una piattaforma online open source dove vengono pubblicati modelli per il machine learning. Gli utenti di questa piattaforma possono collaborare o chiedere aiuto per la definizione di modelli di machine learning.

Per poter utilizzare i modelli presenti sulla piattaforma, HuggingFace mette a disposizione la libreria Transformers, grazie alla quale è possibile scaricare ed utilizzare modelli già addestrati su un determinato elemento che può essere un testo, un'immagine o un video.

I modelli presenti su HuggingFace, infatti, possono essere usati su:

- Testo: si possono effettuare operazioni di classificazione, estrazione delle informazioni, riassunti, traduzioni e generazione del testo in diversi tipi di lingue.
- Immagini: si possono effettuare operazioni di classificazione, riconoscimento di oggetti e segmentazione.
- Audio: per operazioni di riconoscimento dell'audio e classificazione.

## 3.2 Modifiche effettuate su ITA

### 3.2.1 Hate speech it

Hate\_speech\_it [19] è un modello in grado di effettuare la classificazione dell'hate speech dei contenuti presenti sui social media in lingua italiana, infatti è stato addestrato utilizzando 120.000 commenti presenti su YouTube.

Con l'espressione “hate speech” si intendono tutti quei comportamenti verbali soprattutto violenti, minatori, poco rispettosi nei confronti di un'altra persona e che creano un clima di ostilità e un ambiente poco favorevole alle minoranze.

Dunque, hate\_speech\_it è un modello in grado di classificare il livello di ostilità presente all'interno di un testo, assegnando ad esso le seguenti etichette.

- LABEL\_0 – acceptable
- LABEL\_1 – inappropriate
- LABEL\_2 - offensive
- LABEL\_3 – violent

Per poterlo utilizzare, dato che esso è presente sulla piattaforma HuggingFace, è stato necessario importare la libreria transformers. Per integrare il modello, sono state effettuate modifiche nel file di configurazione “process\_tweets.config” e lo script “TweetProcessor.py”.

La modifica nel file di configurazione consiste nell'aggiunta del nuovo campo hate\_speech\_it sotto la voce sentiment, in questo modo l'utente, nel momento in cui effettua l'elaborazione dei dati, può decidere se effettuare o meno la classificazione dell'hate speech sul testo.

Se il valore del nuovo campo viene impostato a True, allora nel momento in cui l'elaborazione dei tweets viene avviata, si invocherà il metodo:

- Def \_\_hate\_speech\_it\_analyze\_sentiment(self, tweet\_text:str,tweet:dict) -> Tuple[int,dict,dict]

in cui si passerà il testo presente nei tweets al modello di hate speech e quest'ultimo restituirà come risultato l'etichetta.

```
def __hate_speech_it_analyze_sentiment(self, tweet_text: str, tweet: dict) -> Tuple[int, dict, dict]:  
    classifier = pipeline("sentiment-analysis",  
model=model_path_hate_speech_it,  
  
tokenizer=model_path_hate_speech_it)  
    hold_list = [tweet_text]  
    temp_hate_level = " "  
    temp_score = " "  
    try:  
        temp_string = classifier(hold_list)[0]  
        temp_string = str(temp_string)  
        if "LABEL_0" in temp_string:  
            temp_hate_level = "acceptable"  
            temp_string = temp_string.replace("LABEL_0", "")  
        elif "LABEL_1" in temp_string:  
            temp_hate_level = "inappropriate"  
            temp_string = temp_string.replace("LABEL_1", "")  
        elif "LABEL_2" in temp_string:  
            temp_hate_level = "offensive"  
            temp_string = temp_string.replace("LABEL_2", "")  
        elif "LABEL_3" in temp_string:  
            temp_hate_level = "violent"  
            temp_string = temp_string.replace("LABEL_3", "")  
  
        # regex that removes everything from string but  
        # the score  
        temp_score = re.findall(r"[-+]?(?:\d*\.\d+)",  
                               temp_string)  
        temp_score = temp_score[0]  
        temp_score = float(temp_score)  
  
        sentiment_analyses: dict[str][float] =  
        {'hate_speech_level': temp_hate_level, 'score':temp_score}
```

In seguito all'elaborazione, il risultato viene salvato nel database. Si riporta un esempio di testo di un tweet elaborato con il relativo risultato:

"**GR** #Forrest 😊 oggi #8marzo Giorgia #Meloni dice: in via straordinaria, mi potete chiamare \"la presidente\"\\nCon @mariannaaprilee@bravimabasta ↗\\n<https://t.co/S0CYrzHjW>,

```
sentiment: Object
hate_speech_it: Object
hate_speech_level: "acceptable"
score: 0.7939720749855042
```

### 3.2.2 Genre Classification: Roberta

Su ITA è stato inoltre integrato un modello in grado di classificare il testo indicandone il contesto, ad esempio può determinare se un testo rappresenta una notizia, un documento, un documento legale, etc.

Il modello in questione si chiama Roberta [20] e nella tabella sottostante si riportano i labels che il modello può assegnare ad un testo, con la loro relativa descrizione.

Label	Descrizione
Information/ Explanation	Testo che ha lo scopo di descrivere un evento, una persona, una cosa o un concetto.
Instruction	Testo che rappresenta istruzioni tecniche in merito all'utilizzo di un oggetto/procedura.
Legal	Testo che presenta termini ed informazioni legali.
News	Informazioni su eventi avvenuti o che avverranno.
Opinion/ Argumentation	Testo che rappresenta opinioni o argomentazioni.
Promotion	Testo che ha lo scopo di sponsorizzare eventi o prodotti.
Forum	Messaggi scritti da utenti su un forum.

Prose/Lyrical	Testo che può rappresentare una poesia o testo di una canzone.
Other	Nessuna delle categorie precedenti.

Anche questo modello è presente sulla piattaforma HuggingFace, dunque per utilizzare Roberta è stato sufficiente utilizzare la libreria Transformers.

Per integrarlo, sono stati modificati il file di configurazione “process\_tweets.config” e lo script “TweetProcessor.py”.

La modifica nel file di configurazione consiste nell’aggiunta del nuovo campo “genre\_classification: roberta”, il quale può essere impostato a True dall’utente nel caso in cui voglia effettuare la classificazione del testo; False altrimenti.

Nel caso in cui l’elaborazione dei tweets viene avviata e il valore di tale campo è True, allora viene invocato il metodo:

- def \_\_classification\_with\_roberta(self, tweet\_text: str, tweet: dict) -> Tuple[int, dict, dict]

il quale riceve il testo dei tweets da elaborare e richiama Roberta. Una volta che il modello ha restituito i risultati, questi vengono salvati all’interno del database.

## Utilizzo di roberta all'interno di ITA:

```
def __classification_with_roberta(self, tweet_text: str, tweet: dict) -> Tuple[int, dict, dict]:
    classifier = pipeline("text-classification",
model=model_path_roberta_classifier,
                                tokenizer=model_path_roberta_classifier)
    hold_list = [tweet_text]
    temp_result = " "
    temp_score = " "
    try:
        temp_string = classifier(hold_list)[0]
        temp_string = str(temp_string)
        if "Information/Explanation" in temp_string:
            temp_result = "Information/Explanation"
        elif "Instruction" in temp_string:
            temp_result = "Instruction"
        elif "Legal" in temp_string:
            temp_result = "Legal"
        elif "News" in temp_string:
            temp_result = "News"
        elif "Opinion/Argumentation" in temp_string:
            temp_result = "Opinion/Argumentation"
        elif "Promotion" in temp_string:
            temp_result = "Promotion"
        elif "Forum" in temp_string:
            temp_result = "Forum"
        elif "Prose/Lyrical" in temp_string:
            temp_result = "Prose/Lyrical"
        elif "Other" in temp_string:
            temp_result = "Other"

        temp_score = re.findall(r"[-+]?(\d*\.*\d+)", temp_string) # regex that removes everything from string but the score
        #since findall returns a list and we want to save just the value inside of the database as a number, we pick just the first value of the list.
        temp_score = temp_score[0]
        temp_score = float(temp_score)
        temp_result = str(temp_result)
        roberta_classification : dict[str][float] = {'genre': temp_result, 'score': temp_score}

    except:
        # if the model is busy wait and retry
        time.sleep(0.01)
        return self.__classification_with_roberta(tweet)
    # return the process id (7), the result and the tweet
    return 7, roberta_classification, tweet
```

Si riporta come esempio il testo di un tweet, con il relativo risultato ottenuto da Roberta:

"Abbiamo votato per cambiare.\nAdesso al governo c'è la destra di Giorgia Meloni ma il \"sistema\" non cambia.\nIl marocchino arrestato a febbraio, irregolare in Italia da sei mesi, ha chiesto la protezione e, pertanto, non può essere espulso.\nChe c@xxo abbiamo votato a fare? <https://t.co/7BH9u9JH4k>",

```
genre_classification: Object
  roberta: Object
    genre: "Opinion/Argumentation"
    score: 0.9900059103965759
```

### 3.2.3 Image to text: vit-gpt2

Su ITA è stata integrata una nuova funzionalità in grado di effettuare image to text, un'operazione che consiste nel ricavare in forma di testo la semantica di un'immagine. Il modello che consente di effettuare questo tipo di elaborazione è Vit-gpt2-image-captioning [21], anch'esso presente sulla piattaforma HuggingFace.

Il funzionamento del modello per image captioning è il seguente: riceve in input l'url di un'immagine e fornisce in output una stringa di testo che rappresenta la descrizione dell'immagine.

Vit-gpt2-image-captioning è stato addestrato utilizzando il dataset COCO (Common Objects in Context) [22], un vasto dataset che contiene foto rappresentanti oggetti comuni nel loro ambiente naturale.

Per poter utilizzare il modello, è stato necessario modificare il payload dei tweet ricevuti, in modo tale da ottenere, per ogni tweet, i campi relativi ai media.

Come esempio si riporta il payload della sezione media di un tweet:

```
{'media': [{width: 3832, media_key: '3_1612948953351192576', type: 'photo', height: 2555, url: 'https://pbs.twimg.com/media/FmJYhv1WAAKmtB.jpg%27%7D'}, {width: 720, media_key: '3_1612948971860725760', type: 'photo', height: 1279, url: 'https://pbs.twimg.com/media/FmJYi0yXEAAi88w.jpg%27%7D'}, {'width: 3832, media_key: '3_1612930927776104448', type: 'photo', height: 2555, url: 'https://pbs.twimg.com/media/FmJIhVWAAA\_Gd7.jpg%27%7D'}, {'width: 676, media_key: '3_1612909821782523905', type: 'photo', height: 1200, url: 'https://pbs.twimg.com/media/Fml07\_YWAAEVjMf.jpg%27%7D'}], 'users': [{"id: 1603345696760946693, name: 'Davide Volpicella', username: 'DavideVolpicel9'}]}
```

Con questa modifica, nel momento in cui si effettua la ricerca dei tweets, nel database verranno salvate delle nuove informazioni, ovvero quelle relative ai media. I nuovi dati che si avranno sono i seguenti:

- mediakeys: array in cui sono contenuti dei codici che servono per identificare in maniera univoca i media all'interno di Twitter;
- Media\_urls: array che contiene gli url delle immagini presenti all'interno del tweet;
- Media\_types: array che contiene il tipo di media che il tweet contiene, questi possono essere: foto, video e/o gif.

Si riporta un esempio di tweet ritrovato contenente dei media:



```
_id: "1627631270376210435"
raw_text: https://t.co/4xd3Zi3SGI
author_id: "1603345696760946693"
author_name: "Davide Volpicella"
author_username: "DavideVolpicel9"
attachments: Object
media_keys: Array
0: "3_1627631225622917121"
1: "3_1627631230916206599"
2: "3_1627631237769711617"
media_urls: Array
0: "https://pbs.twimg.com/media/FpaB-eSWcAE_4iw.jpg"
1: "https://pbs.twimg.com/media/FpaB-yAXoAcSaMt.jpg"
2: "https://pbs.twimg.com/media/FpaB_LiX0AEbAWr.jpg"
media_types: Array
0: "photo"
1: "photo"
2: "photo"
created_at: "2023-02-20T11:28:29.000Z"
```

Per integrarlo, sono stati modificati il file di configurazione “process\_tweets.config” e lo script “TweetProcessor.py”.

La modifica nel file di configurazione consiste nell’aggiunta del nuovo campo “image\_to\_text:image\_captioning”, il quale può essere impostato a True dall’utente nel caso in cui voglia effettuare l’operazione di image to text sulle immagini presenti nei tweets; False altrimenti.

Nel caso in cui l’elaborazione dei tweets viene avviata e il valore di tale campo è True, allora viene invocato il metodo:

- def \_\_image\_captioning(self, urls: list, tweet: dict) -> Tuple[int, dict, dict]

il quale riceve in input gli url delle immagini presenti nei tweets (se presenti) ed invoca Vit-gpt2-image-captioning. I risultati ottenuti vengono successivamente salvati all’interno del database.

```
def __image_captioning(self, urls: list, tweet: dict) ->
    Tuple[int, dict, dict]:
    image_captioning = pipeline("image-to-text",
model="nlpconnect/vit-gpt2-image-captioning")
    try:
        captions = []
        for url in urls:
            captions.append(image_captioning(url))
    except:
        # if the model is busy wait and retry
        time.sleep(0.01)
        return self.__image_captioning(tweet)
    return 8, captions, tweet
```

Come esempio, si riportano i risultati ottenuti dall’elaborazione del tweet mostrato in precedenza:

```
image_to_text: Object
image_captioning: Array
0: "a brown bear standing in a field of grass "
1: "a car with a black seat and a black steering wheel "
2: "a soccer player is running towards the ball "
```

### 3.3 Modifiche effettuate su CrowdPulse

#### 3.3.1 Aggiunta dell'algoritmo per hate speech

Su CrowdPulse sono state effettuate delle modifiche per poter visualizzare i dati ottenuti dal modello di hate speech. Le modifiche hanno riguardato l'aggiunta di una nuova voce nel filtro “Algorithm”, ovvero “hate\_speech\_it” e la realizzazione di nuovi grafici per la visualizzazione di dati.

Con queste modifiche, nel momento in cui l'utente selezionerà la nuova voce, i tweet presenti all'interno del database che contengono il campo relativo all'hate speech verranno estratti. Per far ciò, sono state effettuate modifiche allo script SentimentFilters.js:

```
while(i<data.length){
    if(data[i].sentiment!==undefined){
        if(data[i].sentiment['hate_speech_it']!==undefined){
            if(data[i].sentiment['hate_speech_it'].hate_speec
                h_level === 'acceptable'){
                    acceptable++;
                }elseif(data[i].sentiment['hate_speech_it'].hate_speech_level==
                    ='inappropriate'){
                        inappropriate++;
                    }elseif(data[i].sentiment['hate_speech_it'].hate_speech_level==
                    ='offensive'){
                        offensive++;
                    }elseif(data[i].sentiment['hate_speech_it'].hate_speech_level==
                    ='violent'){
                        violent++;
                    }
                }
            }
        i++;
    }
```

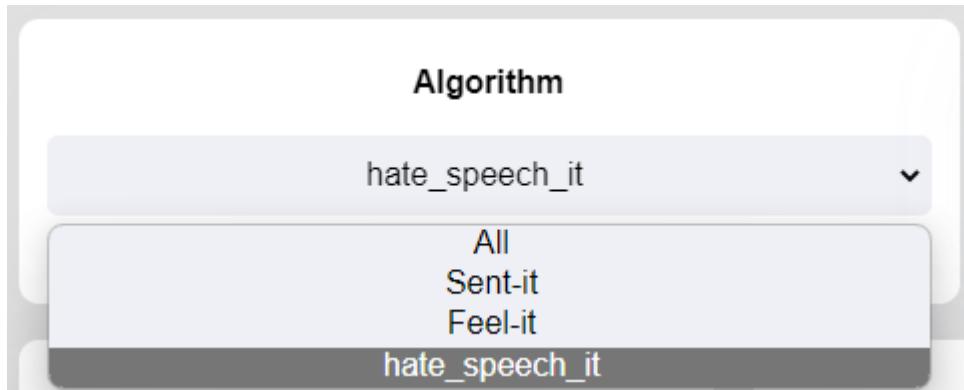


Figura 16- filtro algorithm con hate speech integrato

I grafici realizzati per la visualizzazione dei dati ottenuti da hate\_speech\_it sono due, il primo è un grafico a barre mentre il secondo è di tipo radar. Per la loro realizzazione sono stati realizzati due nuovi script: HatespeechBarChart.js e HatespeechRadarChart.js.

### HatespeechBarChart:

```
import React from "react";
import {Bar} from 'react-chartjs-2';

const HatespeechBarChart = (props) =>{
    var acceptable = props.hatespeechCounter.acceptable;
    var inappropriate = props.hatespeechCounter.inappropriate;
    var offensive = props.hatespeechCounter.offensive;
    var violent = props.hatespeechCounter.violent;
    return(
        <div>
            <Bar
data= {{
    labels: ['acceptable','inappropriate', 'offensive','violent'],
    datasets: [{

        label: 'hate_speech_level',
        data: [acceptable, inappropriate, offensive, violent],
        backgroundColor: [
            'rgba(255, 99, 132, 0.2)',
            'rgba(75, 192, 192, 0.2)',
            'rgba(255, 206, 86, 0.2)',
            'rgba(255, 206, 86, 0.2'

        ],
        borderColor: [

```

```

        'rgba(255, 99, 132, 1)',
        'rgba(75, 192, 192, 1)',
        'rgba(255, 206, 86, 1)',
        'rgba(255, 206, 86, 1)'
    ],
    borderWidth: 1
}]
})
width={100}
height={400}
options={{ maintainAspectRatio: false }}
/>
</div>
)
}
export default HatespeechBarChart

```

## HatespeechRadarChart:

```

import React from "react";
import {Radar} from 'react-chartjs-2';

const HatespeechRadarChart = (props) =>{
  var acceptable = props.hatespeechCounter.acceptable;
  var inappropriate = props.hatespeechCounter.inappropriate;
  var offensive = props.hatespeechCounter.offensive;
  var violent = props.hatespeechCounter.violent;
  return(
    <div>
      <Radar
    data= {{
      labels: [
        'acceptable',
        'inappropriate',
        'offensive',
        'violent',
      ],
      datasets: [{
        label: 'hate_speech_level',
        data: [acceptable,inappropriate, offensive, violent],
        fill: true,
        backgroundColor: 'rgba(255, 99, 132, 0.2)',
        borderColor: 'rgb(255, 99, 132)',
        pointBackgroundColor: 'rgb(255, 99, 132)',
        pointBorderColor: '#fff',
        pointHoverBackgroundColor: '#fff',
        pointHoverBorderColor: 'rgb(255, 99, 132)'
      }]
    }}}

```

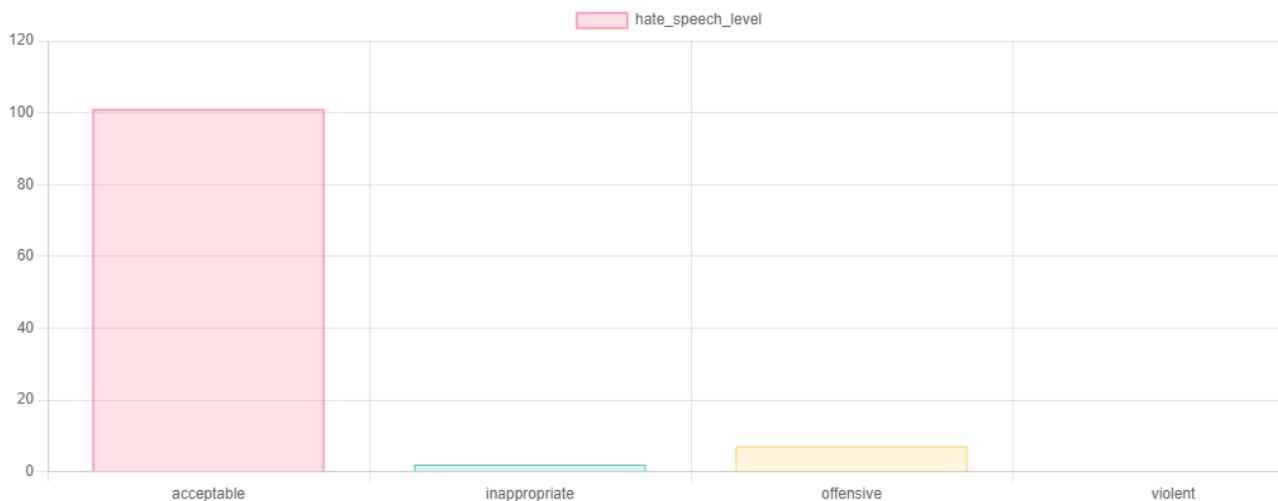
```

        width={100}
        height={400}
        options={{ maintainAspectRatio: false }}
            />
        </div>
    )
}

export default HatespeechRadarChart

```

I due script sono stati importati su SentimentCharts.js, dove vi è un controllo che modifica i grafici che vengono visualizzati in base all'algoritmo scelto. Dunque, se l'utente seleziona l'algoritmo `hate_speech_it`, appariranno i seguenti grafici:



*Figura 17- Grafico a barre di hate\_speech\_it su tweets inerenti a Giorgia Meloni*

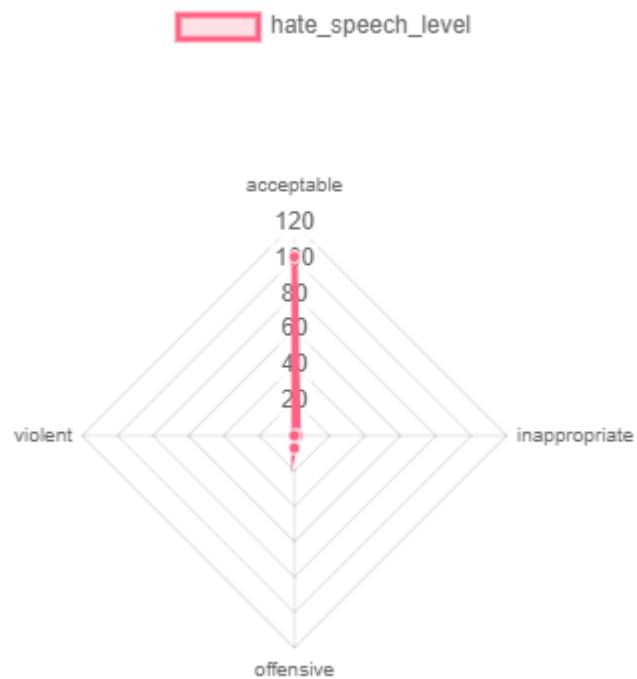


Figura 18 Grafico radar di hate\_speech\_it su tweets inerenti Giorgia Meloni

### 3.3.2 Integrazione di un nuovo filtro per text categorization

Su CrowdPulse è stato aggiunto un nuovo filtro che permette all’utente di selezionare i tweet in base ai labels ottenuti da Roberta, un modello in grado di effettuare text categorization. Per far ciò, nello script Filters.js sono state realizzati nuove funzioni che consentono di estrarre i tweets dal database in base alla loro categoria.

```
filterByGenre = () => {
    var i=0;
    var k=0;
    var temp=[];
    if(this.state.flagGenre==0 || this.state.flagGenre=='0'){
        while(i<this.state.data.length){
            if(this.state.data[i].genre_classification!=undefined){
                if(this.state.data[i].genre_classification['roberta']!=undefined){
                    temp[k] = this.state.data[i];
                    k++;
                }
            }
            i++;
        }
    }
}
```

```

        }
    }else if(this.state.flagGenre==1 || this.state.flagGenre=='1'){
        while(i<this.state.data.length){
            if(this.state.data[i].genre_classification!=undefined){
                if(this.state.data[i].genre_classification['roberta']!=undefined){
                    if(this.state.data[i].genre_classification['roberta'].genre=='Information/Explanation'){
                        temp[k] = this.state.data[i];
                        k++;
                    }
                }
            }
            i++;
        }
    }else if(this.state.flagGenre==2 || this.state.flagGenre=='2'){
        while(i<this.state.data.length){
            if(this.state.data[i].genre_classification!=undefined){
                if(this.state.data[i].genre_classification['roberta']!=undefined){
                    if(this.state.data[i].genre_classification['roberta'].genre=='Instruction'){
                        temp[k] = this.state.data[i];
                        k++;
                    }
                }
            }
            i++;
        }
    }else if(this.state.flagGenre==3 || this.state.flagGenre=='3'){
        while(i<this.state.data.length){
            if(this.state.data[i].genre_classification!=undefined){
                if(this.state.data[i].genre_classification['roberta']!=undefined){
                    if(this.state.data[i].genre_classification['roberta'].genre=='Legal'){
                        temp[k] = this.state.data[i];
                        k++;
                    }
                }
            }
            i++;
        }
    }else if(this.state.flagGenre==4 || this.state.flagGenre=='4'){
        while(i<this.state.data.length){
            if(this.state.data[i].genre_classification!=undefined){
                if(this.state.data[i].genre_classification['roberta']!=undefined){
                    if(this.state.data[i].genre_classification['roberta'].genre=='News'){
                        temp[k] = this.state.data[i];
                        k++;
                    }
                }
            }
        }
    }
}

```

```

        }
        i++;
    }
}else if(this.state.flagGenre==5 || this.state.flagGenre=='5'){
    while(i<this.state.data.length){
        if(this.state.data[i].genre_classification!=undefined){
            if(this.state.data[i].genre_classification['roberta']!=undefined){
                if(this.state.data[i].genre_classification['roberta'].genre=='Opinion/Argumentation'){
                    temp[k] = this.state.data[i];
                    k++;
                }
            }
        }
        i++;
    }
}else if(this.state.flagGenre==6 || this.state.flagGenre=='6'){
    while(i<this.state.data.length){
        if(this.state.data[i].genre_classification!=undefined){
            if(this.state.data[i].genre_classification['roberta']!=undefined){
                if(this.state.data[i].genre_classification['roberta'].genre=='Promotion'){
                    temp[k] = this.state.data[i];
                    k++;
                }
            }
        }
        i++;
    }
}else if(this.state.flagGenre==7 || this.state.flagGenre=='7'){
    while(i<this.state.data.length){
        if(this.state.data[i].genre_classification!=undefined){
            if(this.state.data[i].genre_classification['roberta']!=undefined){
                if(this.state.data[i].genre_classification['roberta'].genre=='Forum'){
                    temp[k] = this.state.data[i];
                    k++;
                }
            }
        }
        i++;
    }
}else if(this.state.flagGenre==8 || this.state.flagGenre=='8'){
    while(i<this.state.data.length){
        if(this.state.data[i].genre_classification!=undefined){
            if(this.state.data[i].genre_classification['roberta']!=undefined){
                if(this.state.data[i].genre_classification['roberta'].genre=='Prospective/Lyrical'){
                    temp[k] = this.state.data[i];
                    k++;
                }
            }
        }
    }
}

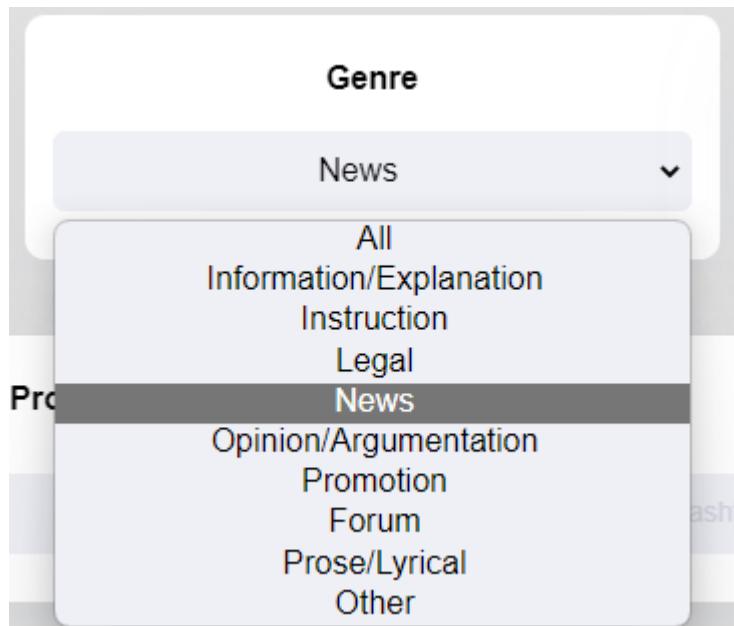
```

```

        }
    }
    i++;
}
}else if(this.state.flagGenre==9 || this.state.flagGenre=='9'){
    while(i<this.state.data.length){
        if(this.state.data[i].genre_classification!=undefined){
            if(this.state.data[i].genre_classification['roberta']!=undefined){
                if(this.state.data[i].genre_classification['roberta'].genre=='Other'){
                    temp[k] = this.state.data[i];
                    k++;
                }
            }
        }
        i++;
    }
}
this.state.data = temp
this.handleQuery();
}

```

Il filtro appare sull'applicazione nel seguente modo:



*Figura 19 - Nuovo filtro per genre classification*

### 3.3.3 Modifica dei filtri Algorithm e Sentiment

Nella versione precedente di CrowdPulse, i filtri Algorithm e Sentiment sono un indipendente dall'altro. Data l'aggiunta di nuovi algoritmi e di nuovi labels, si è riscontrata la necessità di rendere i due filtri dipendenti. Ovvero, far apparire nel filtro Sentiment quei labels che corrispondono all'algoritmo selezionato. Ad esempio, se si seleziona l'algoritmo Feel-it, nel filtro sentiment, dovranno apparire i seguenti labels: positive, negative, neutral, joy, sadness, anger, fear.

Per rendere i due filtri dipendenti, è stata realizzata la funzione configureDropDownLists, la quale viene invocata ogni volta che si seleziona un algoritmo diverso e questa consente, in base alla selezione dell'algoritmo, di far apparire i corrispettivi label.

```
configureDropDownLists = (ddl1, ddl2) =>{

    var all = ["All", "Positive", "Neutral", "Negative",
              "Joy", "Sadness", "Anger", "Fear",
              "Acceptable", "Inappropriate",
              "Offensive", "Violent"];
    var sent_it = ["All", "Positive", "Neutral", "Negative"];
    var feel_it = ["All", "Positive", "Neutral", "Negative",
                  "Joy", "Sadness", "Anger", "Fear"];
    var hate_speech_it = ["All", "Acceptable",
                          "Inappropriate", "Offensive", "Violent"];
    var i = 0;
    switch (ddl1.value) {
        case '0':
            ddl2.options.length = 0;
            for (i = 0; i < all.length; i++) {
                this.createOption(ddl2, all[i], all[i]);
            }
            break;
        case '1':
            ddl2.options.length = 0;
            for (i = 0; i < sent_it.length; i++) {
                this.createOption(ddl2, sent_it[i], sent_it[i]);
            }
            break;
        case '2':
            ddl2.options.length = 0;
            for (i = 0; i < feel_it.length; i++) {
                this.createOption(ddl2, feel_it[i], feel_it[i]);
            }
            break;
        case '3':
            ddl2.options.length = 0;
            for (i = 0; i < hate_speech_it.length; i++) {
                this.createOption(ddl2, hate_speech_it[i], hate_speech_it[i]);
            }
            break;
    }
}
```

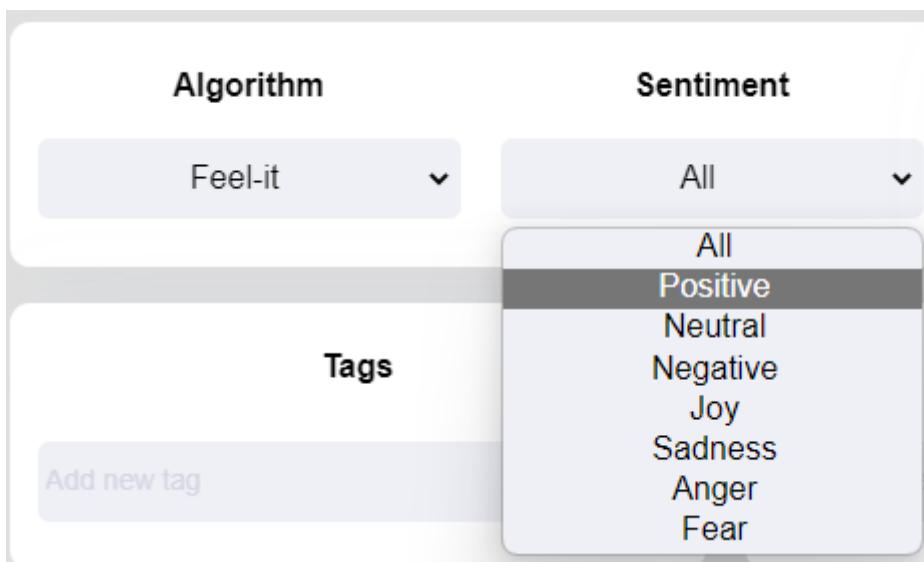
```

        ddl2.options.length = 0;
        for (i = 0; i < feel_it.length; i++) {
            this.createOption(ddl2, feel_it[i], feel_it[i]);
        }
        break;
    case '3':
        ddl2.options.length = 0;
        for (i = 0; i < hate_speech_it.length; i++) {
            this.createOption(ddl2, hate_speech_it[i],
                hate_speech_it[i]);
        }
        break;

    default:
        ddl2.options.length = 0;
        break;
    }
    this.resetCategory2(ddl1);
    this.handleCategory2(ddl1);
}

```

Si riportano alcuni esempi:



*Figura 20 - Voci del filtro sentiment dell'algoritmo Feel-it*

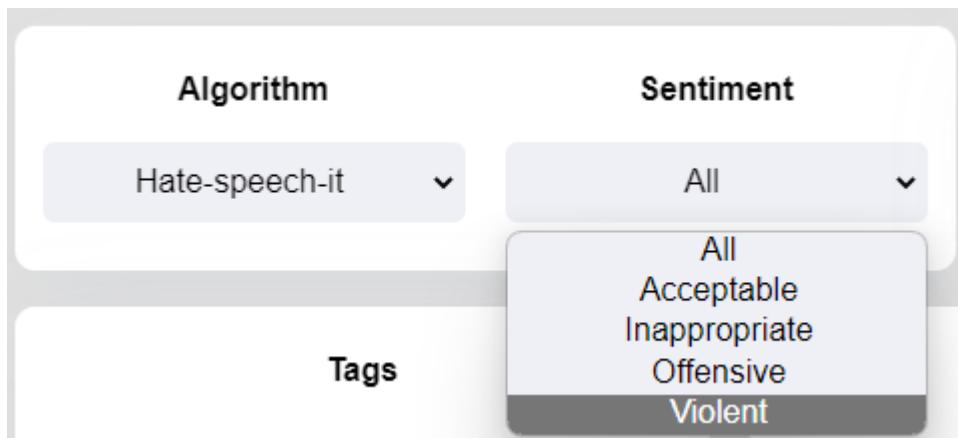


Figura 21- Voci del filtro sentiment dell'algoritmo Hate\_speech\_it

### 3.3.4 Realizzazione di una nuova sezione per le captions

Nel programma è stata aggiunta una nuova sezione denominata “Captions”, in cui è possibile visualizzare in una tabella le informazioni relative ai media presenti all’interno del database selezionato.

Nella tabella, per ogni tweet, vengono riportati:

- l’username;
- l’url dell’immagine presente all’interno del tweet;
- il testo generato dal modello Vit-gpt2-image-captioning, il quale rappresenta la semantica dell’immagine.

Per la realizzazione della nuova tabella sono stati realizzati dei nuovi script: CaptionsTable.js e index.jsx; presenti entrambi nella cartella Table>Captions.

## Implementazione della sezione “Captions”:

```
import React, { useState } from "react";

import "../table.css";
import Table from ".";
import jsPDF from 'jspdf'
import 'jspdf-autotable'
import XLSX from 'xlsx'

var exportData=[];
const columns = [
  { title: "username", field: "author_username", },
  { title: "image", field: "media_urls", },
  { title: "caption", field: "image_to_text", }]

const setData = (props) =>{

  var i = 0
  while (i<props.data.length) {
    exportData.push({
      author_username:props.data[i].author_username,
      media_urls:printMediaUrls(props.data[i]),
      image_to_text:printImageToText(props.data[i])
    })
    i++
  }
}

const downloadExcel = () => {
  const newData = exportData.map(row => {
    delete row.tableData
    return row
  })
  const workSheet = XLSX.utils.json_to_sheet(newData)
  const workBook = XLSX.utils.book_new()
  XLSX.utils.book_append_sheet(workBook, workSheet, "students")
  //Buffer
  XLSX.write(workBook, { bookType: "xlsx", type: "buffer" })
  //Binary string
  XLSX.write(workBook, { bookType: "xlsx", type: "binary" })
  //Download
  XLSX.writeFile(workBook, "TweetData.xlsx")

}

const printMediaUrls = (data) =>{
  var image
```

```

        if(data.media_urls!==undefined){
            image = data.media_urls
            return image
        }else{
            return("")
        }
    }

const printImageToText = (data) =>{
    var caption

    if(data.image_to_text!==undefined){
        caption = data.image_to_text.image_captioning
        return caption
    }else{
        return("")
    }
}

const downloadPdf = () => {
    const doc = new jsPDF('landscape')

    console.log(exportData)
    doc.text("Tweet Details", 20, 10)
    doc.autoTable({
        theme: "grid",
        columns: columns.map(col => ({ ...col, dataKey: col.field })),
        body: exportData
    })
    doc.save('table.pdf')
}

const DisplayTable = (props) => {
    setData(props)
    return (
        <main className="container_table">
            <button className='button activeButton' onClick={() => downloadPdf()}> Export Table</button>
            <button className='button activeButton' onClick={() => downloadExcel()}> Export Excel</button>
            <br/><br/><br/>
            <div className="wrapper_table">
                <Table data={props.data} rowsPerPage={100} />
            </div>
        </main>
    );
};

export default DisplayTable;

```

```
import React, { useState } from "react";

import useTable from "../../hooks/useTable.js";
import "../table.css";
import TableFooter from "../TableFooter";

const printMediaUrls = (data) =>{
    if(data.media_urls!==undefined){
        return(
            data.media_urls.map(item=>(<ol><li><p><a href={item} target="_blank" className="">{item}</a></p></li></ol>))
        )
    }else{
        return("")
    }
}

const printImageToText = (data) =>{
    if(data.image_to_text!==undefined){
        return(
            data.image_to_text.image_captioning.map(item=>(<ol><li><p><a href={item} target="_blank" className="">{item}</a></p></li></ol>))
        )
    }else{
        return("")
    }
}

const Table = ({ data, rowsPerPage }) => {
    const [page, setPage] = useState(1);
    const { slice, range } = useTable(data, page, rowsPerPage);

    return (
        <>
            <table className="table" id="tabella">
                <thead className="tableRowHeader">
                    <tr>
                        <th className="tableHeader">Username</th>
                        <th className="tableHeader">Image</th>
                        <th className="tableHeader">Caption</th>
                    </tr>
                </thead>
                <tbody>
```

```

{slice.map((data) => (
  <tr className="tableRowItems" >
    <td className="tableCell">{data.author_username}</td>
    <td className="tableCell">{printMediaUrls(data)} </td>
    <td className="tableCell">{printImageToText(data)} </td>
  </tr>
))})
</tbody>
</table>

      <TableFooter range={range} slice={slice} setPage={setPage}
page={page} />
    </>
  );
};

export default Table;

```

La nuova sezione si presenta nel seguente modo:

USERNAME	IMAGE	CAPTION
DavideVolpicel9	<a href="https://pbs.twimg.com/media/FpaBccSXoAEUPUD.jpg">https://pbs.twimg.com/media/FpaBccSXoAEUPUD.jpg</a> <a href="https://pbs.twimg.com/media/FpaBc3OXsAEZUGT.jpg">https://pbs.twimg.com/media/FpaBc3OXsAEZUGT.jpg</a> <a href="https://pbs.twimg.com/media/FpaBdZ3WcAAaAhj.jpg">https://pbs.twimg.com/media/FpaBdZ3WcAAaAhj.jpg</a> <a href="https://pbs.twimg.com/media/FpaBd6fWYAIF5oy.jpg">https://pbs.twimg.com/media/FpaBd6fWYAIF5oy.jpg</a>	a large jetliner flying through a blue sky a dog is laying down in the grass a soccer player is running towards the ball a car with a black seat and a black steering wheel
DavideVolpicel9	<a href="https://pbs.twimg.com/media/FpaB-eSWcAE_4iw.jpg">https://pbs.twimg.com/media/FpaB-eSWcAE_4iw.jpg</a> <a href="https://pbs.twimg.com/media/FpaB-yAXoAcSaMt.jpg">https://pbs.twimg.com/media/FpaB-yAXoAcSaMt.jpg</a> <a href="https://pbs.twimg.com/media/FpaB_LiX0AEbAWr.jpg">https://pbs.twimg.com/media/FpaB_LiX0AEbAWr.jpg</a>	a brown bear standing in a field of grass a car with a black seat and a black steering wheel a soccer player is running towards the ball

*Figura 22 - Tabella presente nella sezione Captions*

### 3.3.5 Generazione di WordCloud tramite captions

Nella sezione “Word” è stata aggiunta la voce captions nel filtro Type. Con questa aggiunta, l’utente può generare una word cloud con il testo generato dal modello Vit-gpt2-image-captioning.

Per far ciò è stata realizzata una nuova funzione queryCaptioning nello script WordCloud.js, la quale estrae le captions salvate all’interno del database e le invia alla classe WordCloud.

```
queryCaptioning = () =>{
    var i = 0
    var j = 0
    var k = 0
    var words=[{
        text:null,
        value:0
    }]
    var temp = null
    var arrayWords = []
    var index = 1;
    while(i<this.state.data.length){
        j=0;
        //check image to text not null
        if(this.state.data[i].image_to_text!==undefined){
            while(j<this.state.data[i].image_to_text.image_captioning.length){
                temp=this.state.data[i].image_to_text.image_captioning[j].
                    split(" ");
                //check word
                for(var w = 0; w<temp.length; w++){
                    if(this.checkWord(temp[w])===false
                    ){
                        //check if the word has already been counted
                        if(arrayWords[temp[w]]==undefined){
                            words.push({
                                text:temp[w],
                                value:1
                            })
                            arrayWords[temp[w]] = index;
                            index++;
                        }else{
                            try {

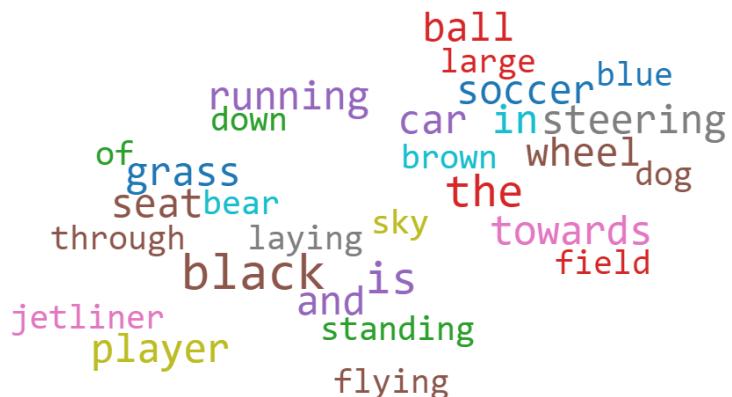
```

```

        words[arrayWords[temp[w]]].value++;
    } catch (error) {
    }
}
}
j++;
}
}
i++;
}
this.state.words=words;
this.setState({words:words});
this.setState({flag:1});
}

```

Si riporta un esempio di word cloud generata da captions:



*Figura 23 - word cloud generata da captions*

## Capitolo 4: Sperimentazione

Italian Tweets Analyzer e CrowdPulse possono essere utilizzati per effettuare analisi in diversi contesti. Tuttavia, il processo di analisi che viene effettuato è sempre lo stesso. Quest'ultimo consiste delle seguenti fasi principali:

- Riconoscimento dei tweets: fase in cui vengono riconosciuti i tweet d'odio attraverso un processo di matching con determinate parole considerate “sensibili”;
- Estrazione: i tweet vengono estratti e salvati localmente. I tweet raccolti sono quelli pubblicati da inizio gennaio a fine marzo 2023;
- Elaborazione dei dati: per ogni tweet si individuano: sentiment, genere, hate speech e si effettua l’operazione di image to text nel caso in cui presentano immagini;
- Visualizzazione: i dati ottenuti dall’elaborazione vengono visualizzati tramite grafici e tavole;
- Analisi dei dati ottenuti: in base ai risultati visti, possono essere effettuate diverse manovre.

Questo processo è stato reso possibile grazie all’integrazione del progetto “Mappa dell’intolleranza Italiana”[23], il quale si occupa di monitorare le forme d’odio presenti in Italia sulla piattaforma Twitter.

### 4.1 Individuazione delle forme d’odio in Italia

Un uso possibile dei due tools potrebbe essere quello di attuare una manovra di prevenzione verso diverse forme d’odio, in modo tale che si possano attuare azioni atte a sensibilizzare le persone verso determinate tematiche.

Dunque, ad esempio, i due tools possono essere utili a amministrazioni locali, scuole ed associazioni che lavorano sul territorio nazionale per poter comprendere in quali zone intervenire e la tipologia di intervento richiesto.

Per dimostrarne il funzionamento, i due programmi sono stati utilizzati per individuare sulla piattaforma Twitter messaggi d'odio delle seguenti categorie:

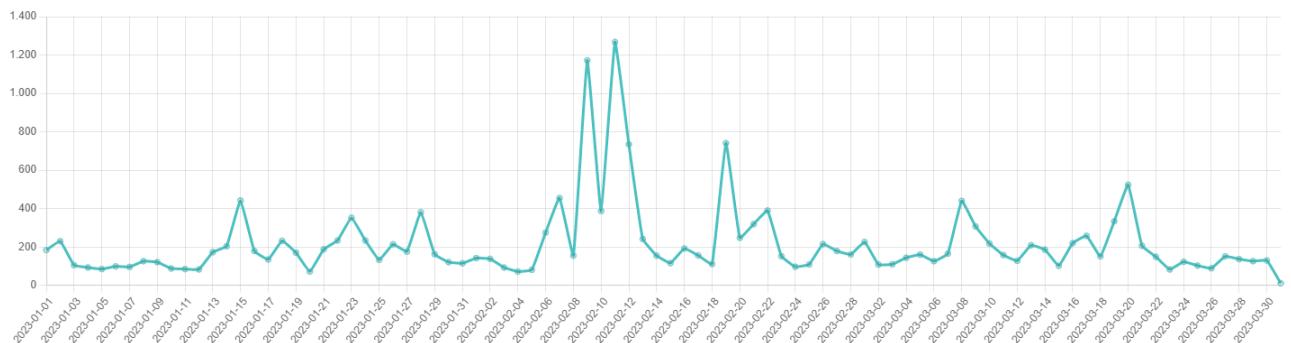
- Misoginia: avversione o repulsione nei confronti delle donne;
- Omofobia: la paura e l'avversione irrazionale nei confronti dell'omosessualità;
- Xenofobia: avversione indiscriminata nei confronti degli stranieri e di tutto ciò che proviene dall'estero.

#### 4.1.1 Misoginia

Da inizio gennaio a fine marzo 2.023 sono stati ritrovati 19.539 tweets che esprimono odio nei confronti delle donne.

#### Misoginia: Timeline

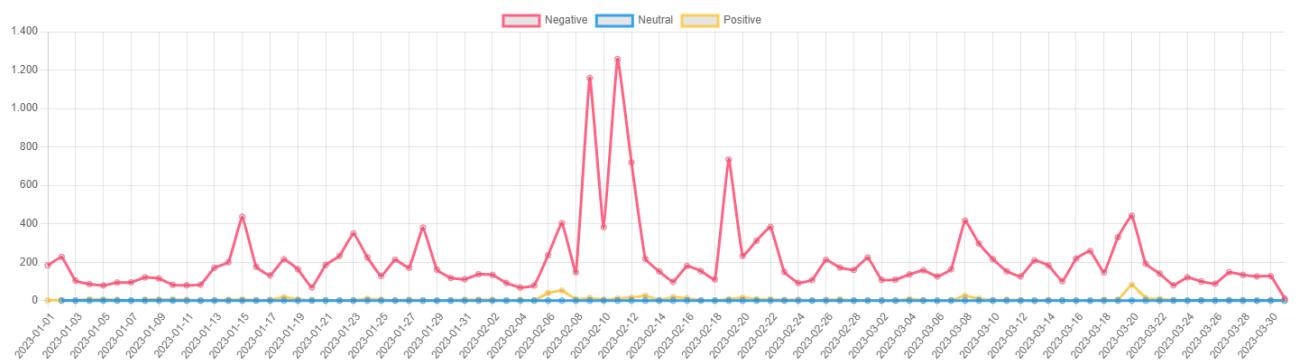
L'analisi della sequenza temporale ha prodotto il seguente risultato:



## Misoginia: Sentiment

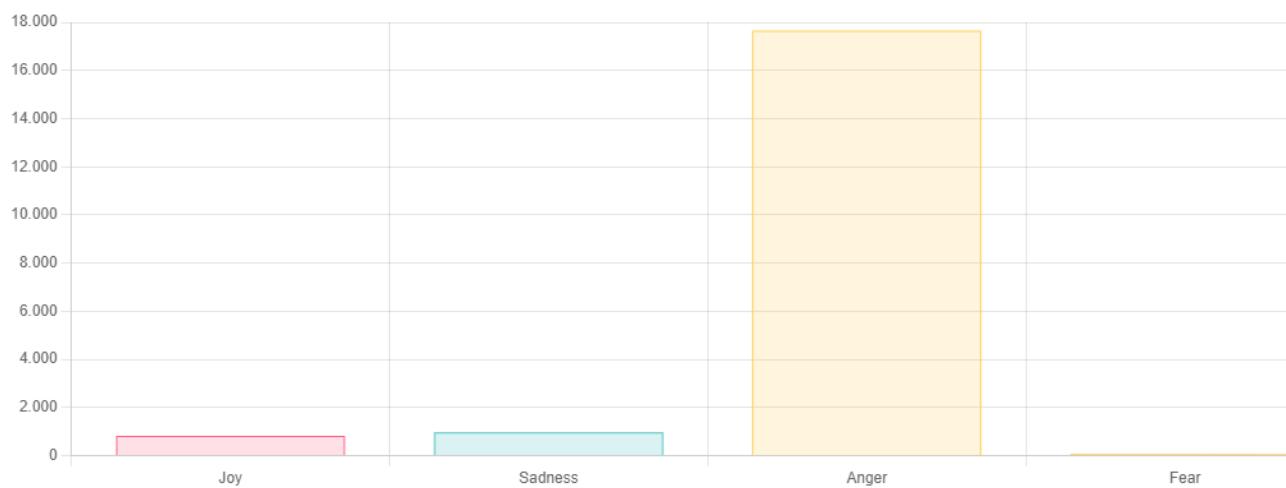
Utilizzando l'algoritmo “Feel-it” i tweet vengono così suddivisi:

- 18.941 (96,94%) tweet considerati negativi
- 0 (0%) tweet considerati neutrali
- 598 (3,06%) tweet considerati positivi



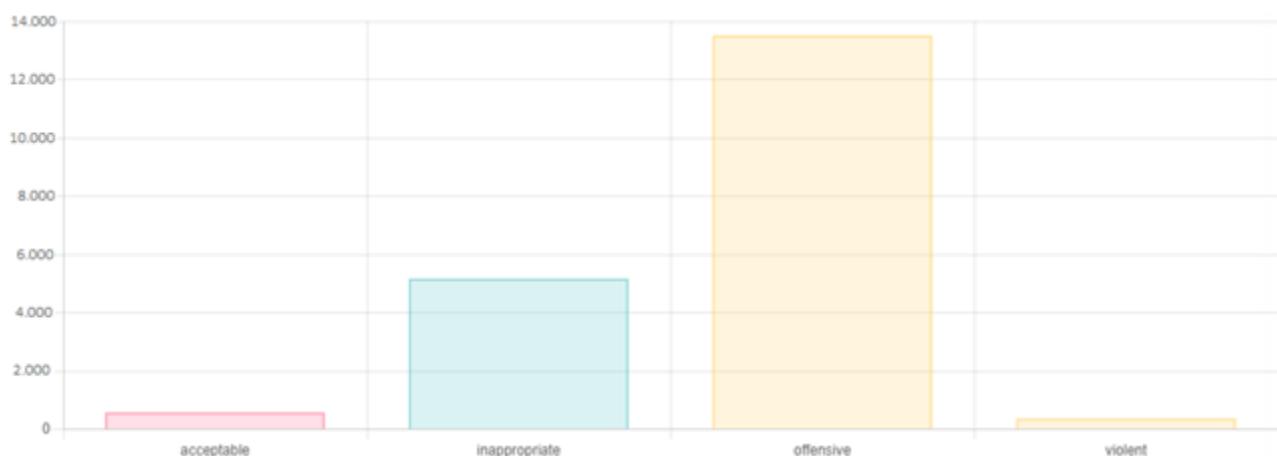
Classificando in base alle emozioni si è ottenuto che:

- 825 tweets esprimono gioia;
- 976 tweets esprimono tristezza;
- 17.662 tweets esprimono rabbia;
- 76 tweets esprimono paura.



Utilizzando l'algoritmo di hate speech, è stato riscontrato che:

- 490 tweets sono accettabili;
- 5.060 tweets sono inappropriati;
- 13.902 tweets sono offensivi;
- 87 tweets sono violenti.



# Misoginia: WordCloud

Word Cloud generata tramite tags presenti nei tweets:

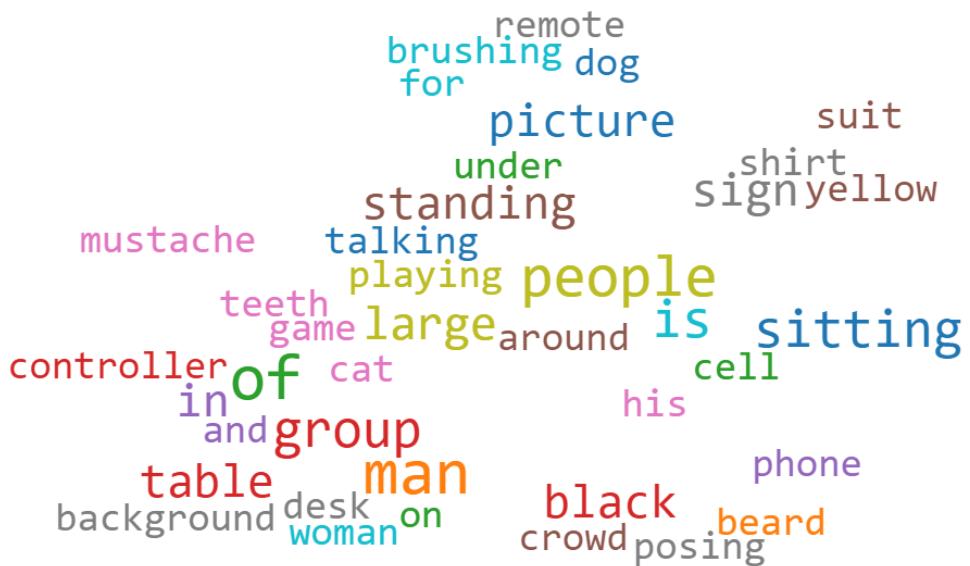


Figura 24 - WordCloud tags (Misoginia)

Word Cloud generata tramite hashtags presenti nei tweets:

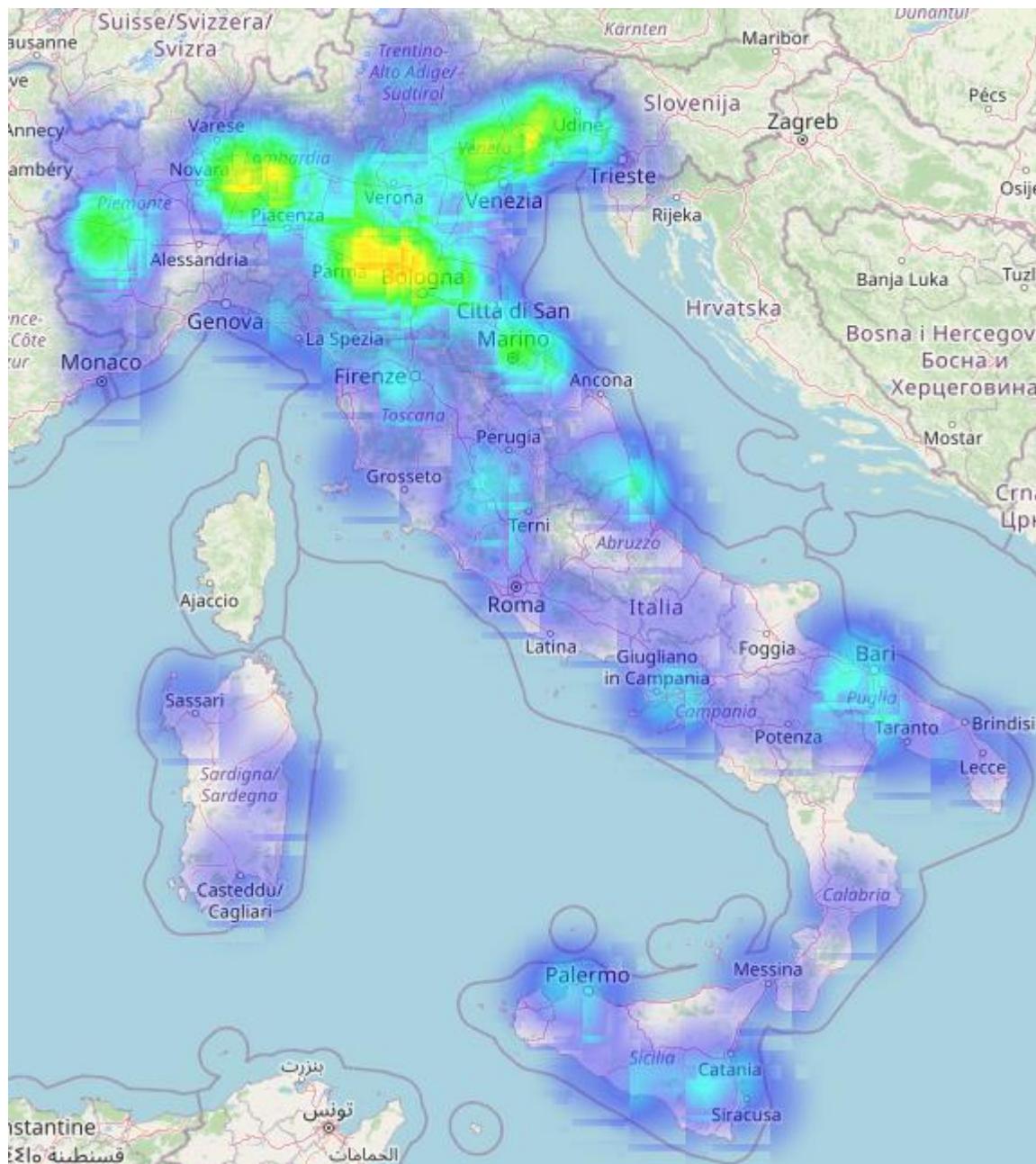


Word Cloud generata tramite captions dei media, ottenute tramite l'operazione di image to text:



## Misoginia: Maps

Dei 19.539 tweets, 8.419 sono stati geolocalizzati.

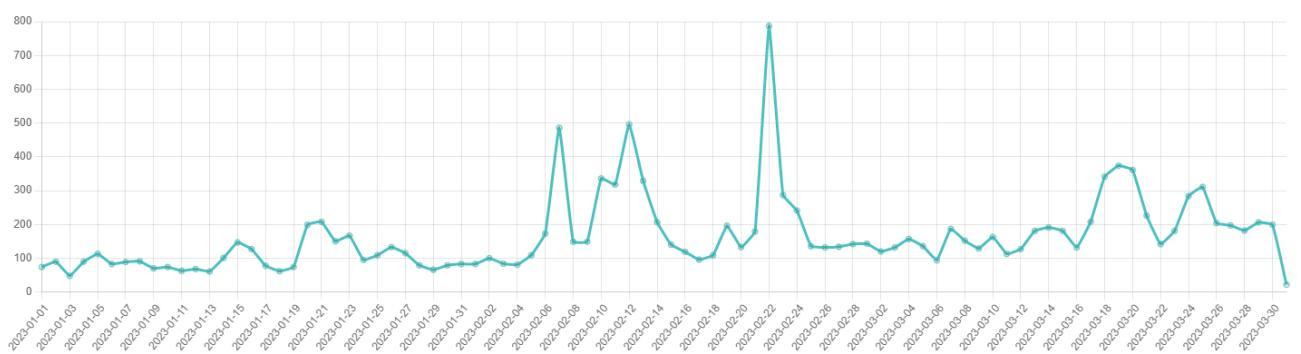


#### 4.1.2 Omofobia

Da inizio gennaio a fine marzo 2023 sono stati ritrovati 19.539 tweets omofobi.

#### Omofobia: Timeline

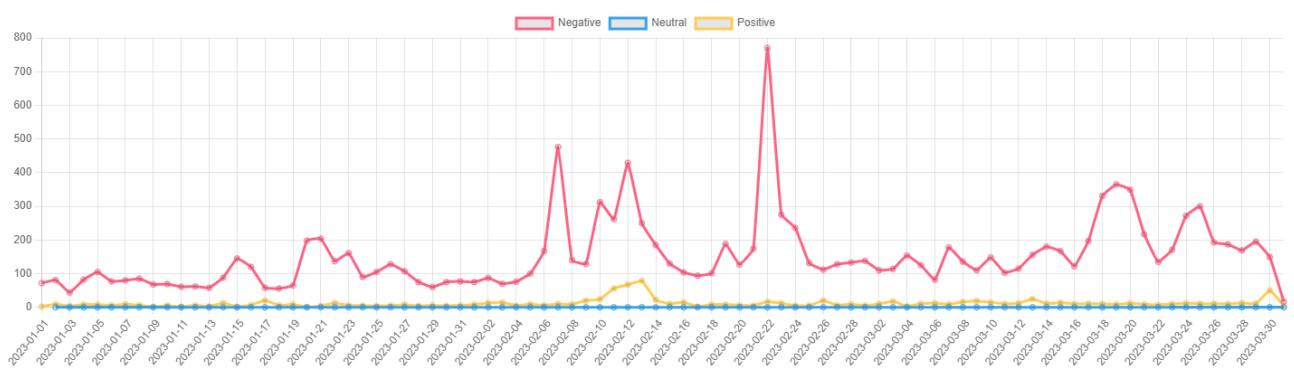
L'analisi della sequenza temporale ha prodotto il seguente risultato.



#### Omofobia: Sentiment

Utilizzando l'algoritmo “Feel-it” i tweet vengono così suddivisi:

- 13.820 (92,86%) tweet considerati negativi
- 0 (0%) tweet considerati neutrali
- 1.062 (7,14%) tweet considerati positivi



Classificando in base alle emozioni si è ottenuto che:

- 1.513 tweets esprimono gioia;
- 321 tweets esprimono tristezza;
- 12.617 tweets esprimono rabbia;
- 431 tweets esprimono paura.

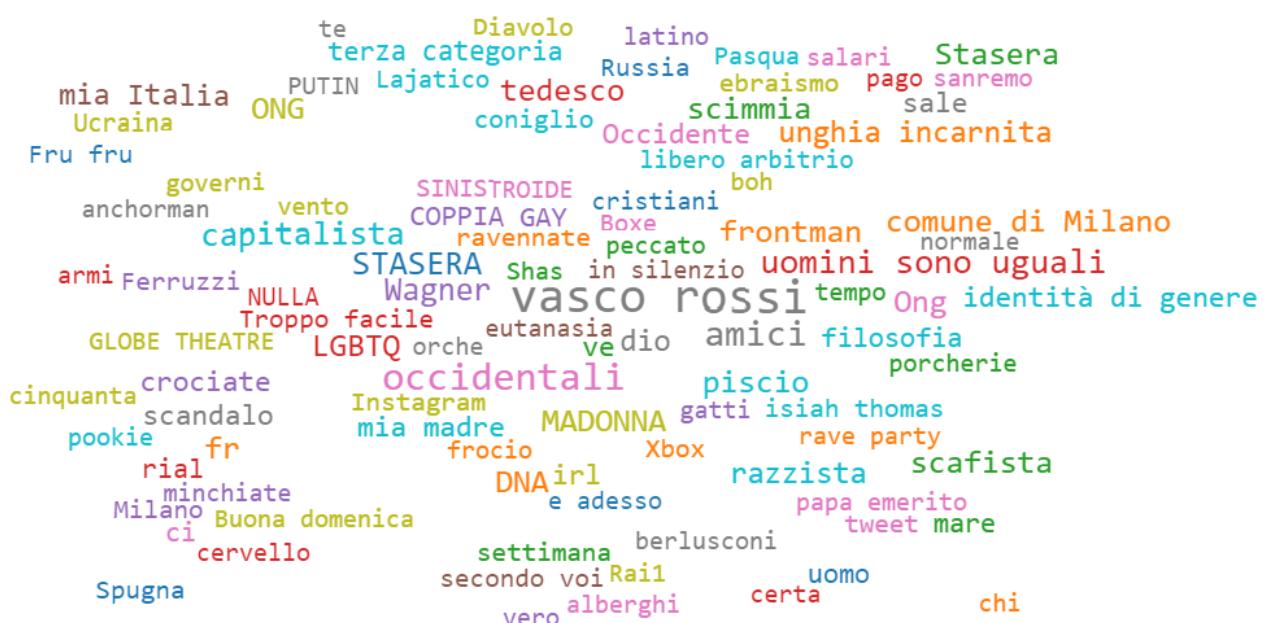


Utilizzando l'algoritmo di hate speech, è stato riscontrato che:

- 1.079 tweets sono accettabili;
  - 698 tweets sono inappropriati;
  - 8.010 tweets sono offensivi;
  - 5.095 tweets sono violenti.

# Omofobia: WordCloud

Word Cloud generata tramite tags presenti nei tweets:



Word Cloud generata tramite hashtags presenti nei tweets:

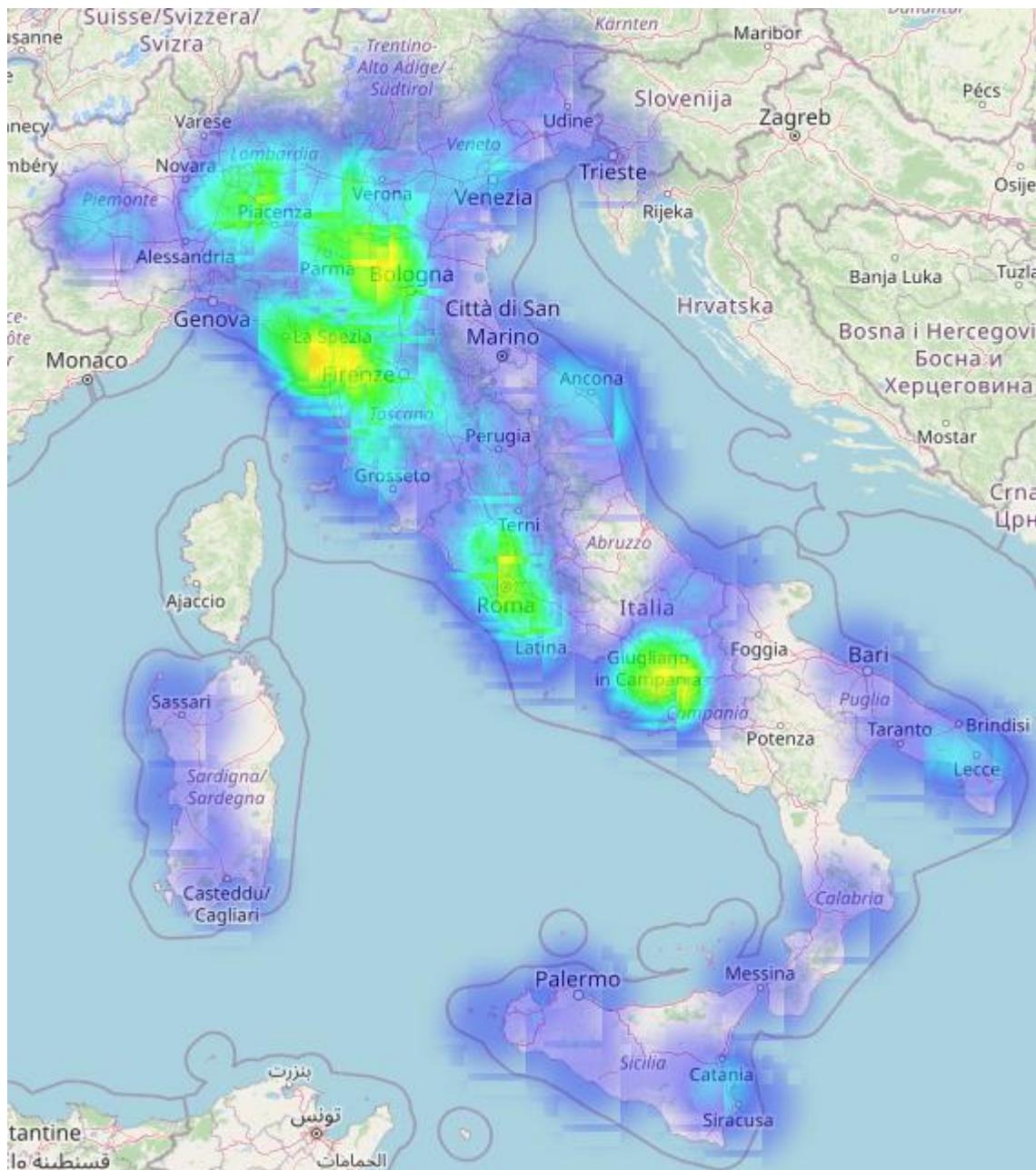


Word Cloud generata tramite captions dei media, ottenute tramite l'operazione di image to text:



## Omofobia: Maps

Dei 14.882 tweets, 6.931 sono stati geolocalizzati.



### 4.1.3 Xenofobia

Da inizio gennaio a fine marzo 2023 sono stati ritrovati 25.281 tweets che esprimono xenofobia.

#### Xenofobia: Timeline

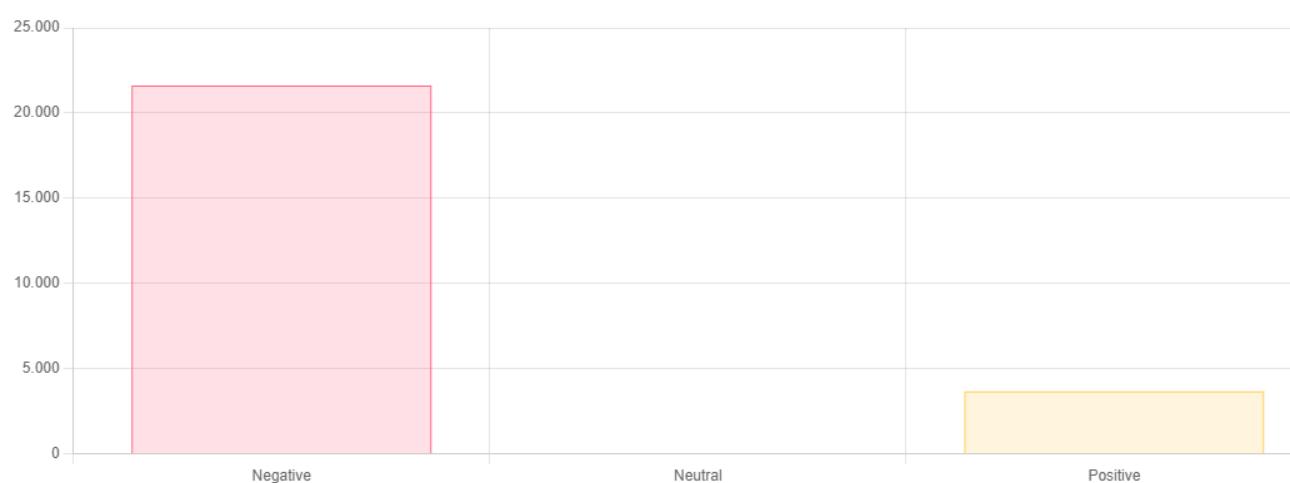
L'analisi della sequenza temporale ha prodotto il seguente risultato:

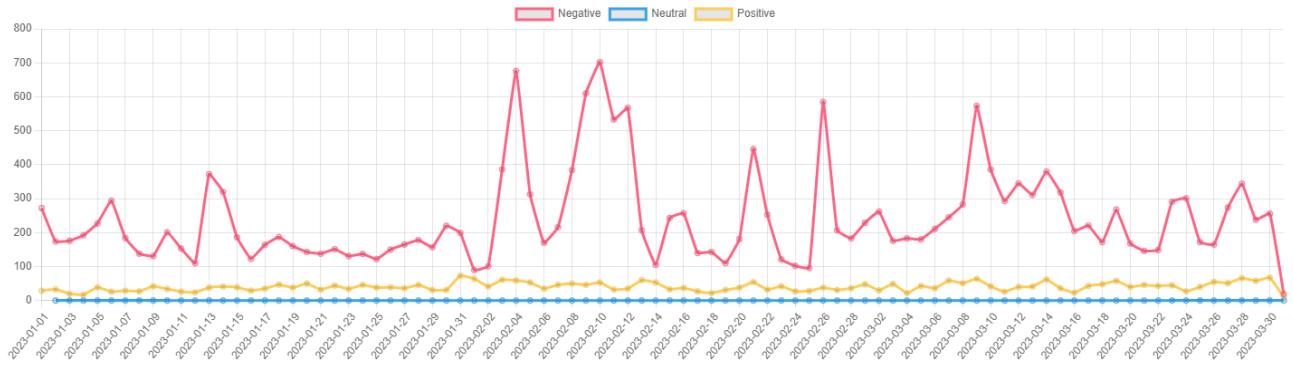


#### Xenofobia: Sentiment

Utilizzando l'algoritmo "Feel-it" i tweet vengono così suddivisi:

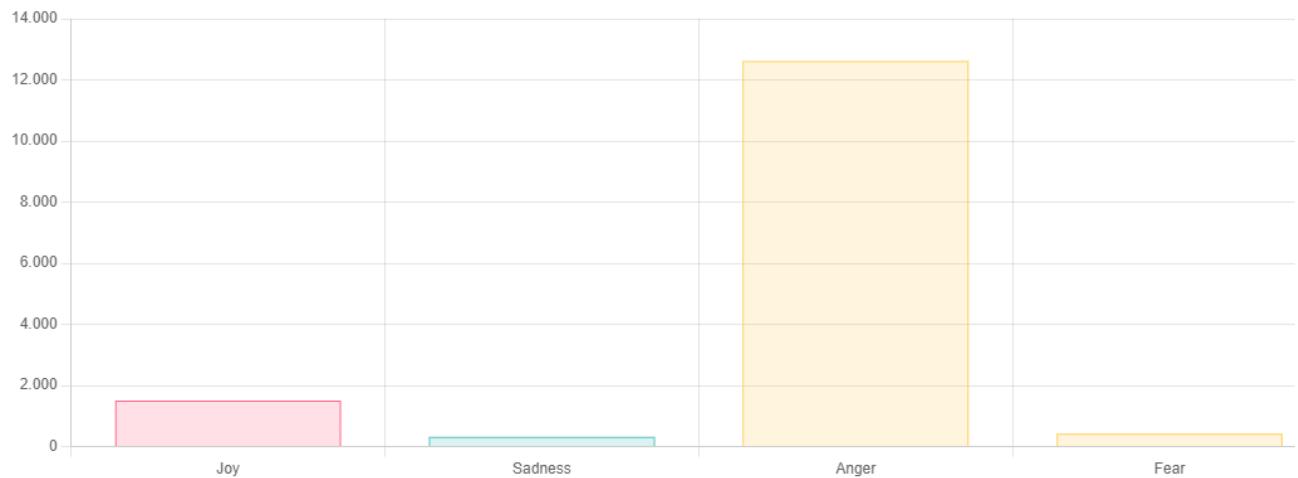
- 21.615 (85,50%) tweet considerati negativi
- 0 (0%) tweet considerati neutrali
- 3.666 (14,50%) tweet considerati positivi





Classificando in base alle emozioni si è ottenuto che:

- 3.401 tweets esprimono gioia;
- 1.430 tweets esprimono tristezza;
- 19.978 tweets esprimono rabbia;
- 472 tweets esprimono paura.



Utilizzando l'algoritmo di hate speech, è stato riscontrato che:

- 13563 tweets sono accettabili;
  - 20 tweets sono inappropriati;
  - 9098 tweets sono offensivi;
  - 2600 tweets sono violenti.

# Xenofobia: WordCloud

Word Cloud generata tramite tags presenti nei tweets:



Word Cloud generata tramite hashtags presenti nei tweets:

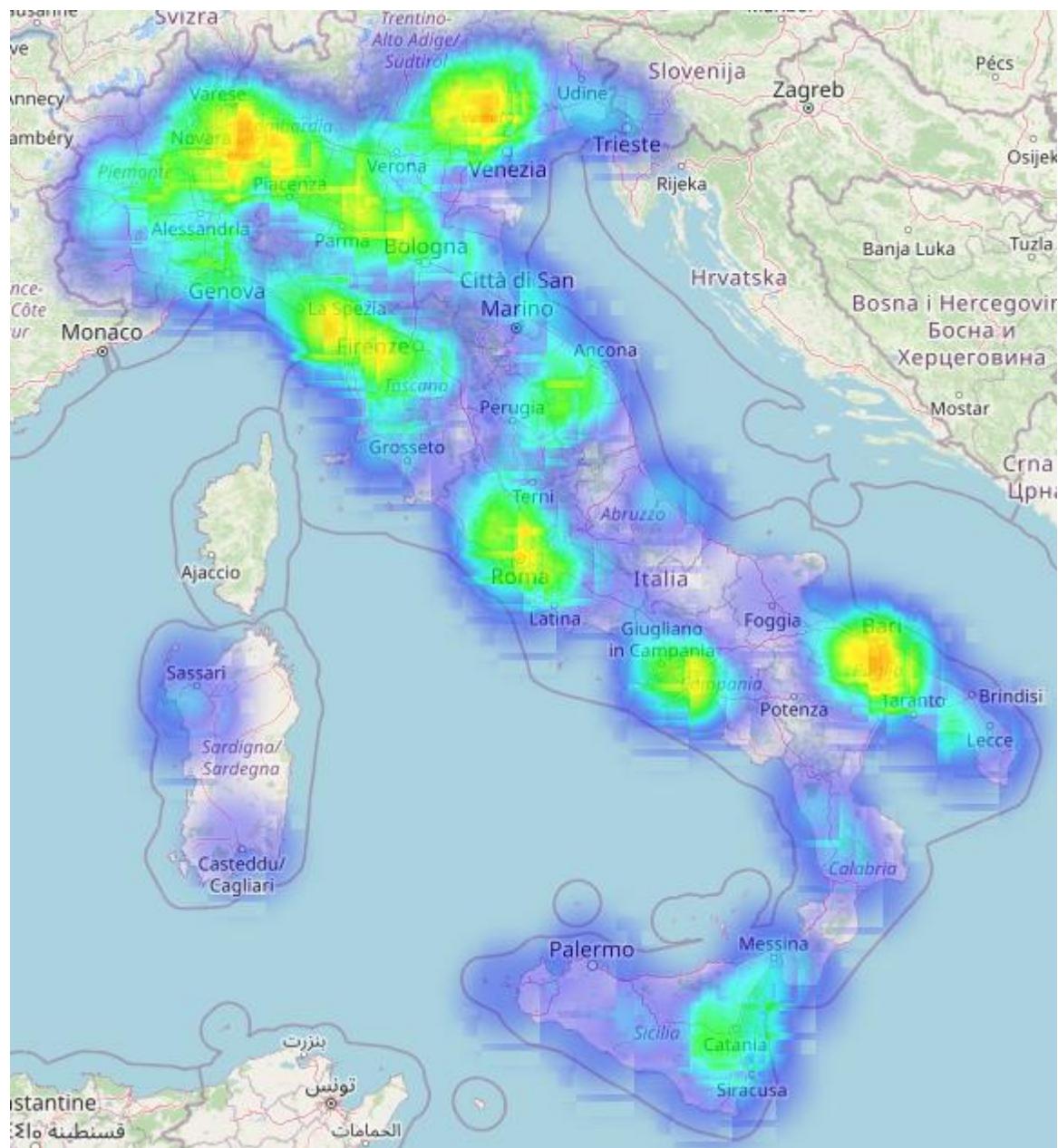


Word Cloud generata tramite captions dei media, ottenute tramite l'operazione di image to text:



## Xenofobia: Maps

Dei 25.281 tweets, 12.497 sono stati geolocalizzati.



## 4.2 Valutazione dell'opinione pubblica in merito a determinati eventi

ITA e CrowdPulse possono anche essere utilizzati per valutare l'opinione pubblica italiana su Twitter in merito a determinati eventi che avvengono nel mondo. In questa fase di sperimentazione, si è scelto di valutare il conflitto Russia-Ucraina. Dunque, attraverso ITA sono state effettuate due ricerche:

- ricerca di tweets d'odio nei confronti della Russia;
- ricerca di tweets d'odio nei confronti dell'Ucraina.

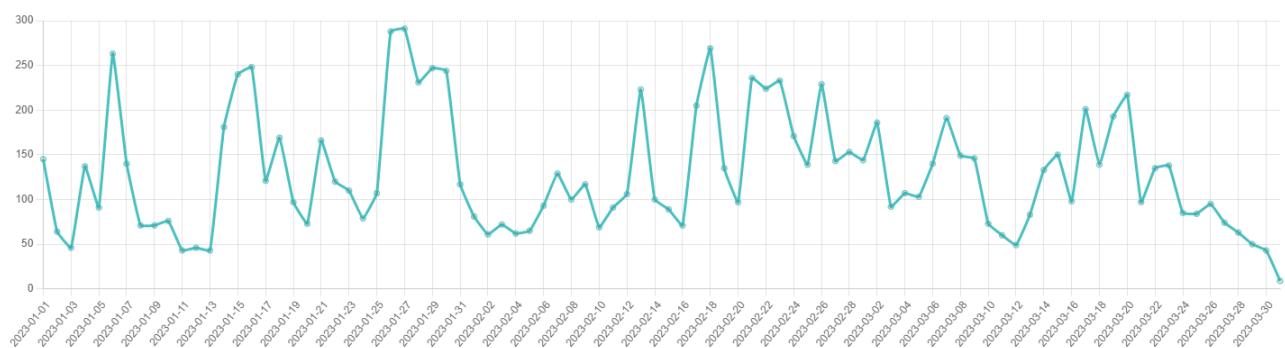
I tweet ricercati ricadono nel range temporale che va' da inizio gennaio a fine marzo.

### 4.2.1 Russia

Da inizio gennaio a fine marzo 2023 sono stati ritrovati 11.744 tweets d'odio nei confronti della Russia.

#### Russia: TimeLines

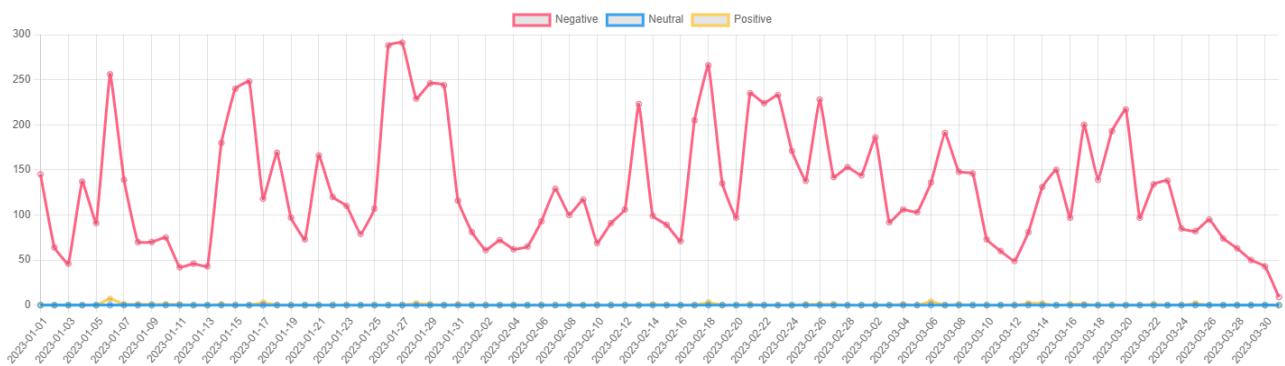
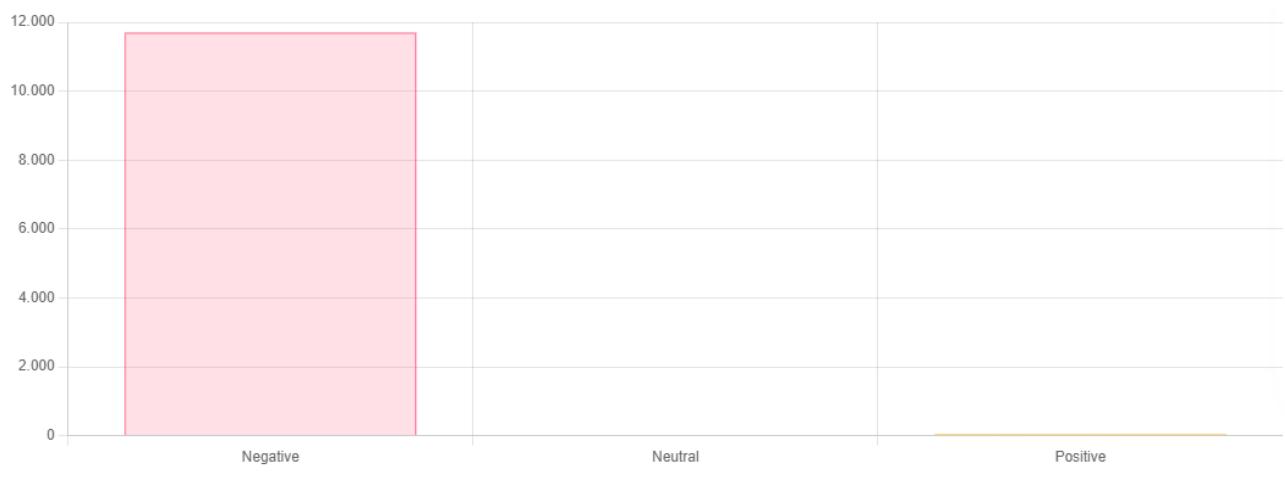
L'analisi della sequenza temporale ha prodotto il seguente risultato:



## Russia: Sentiment

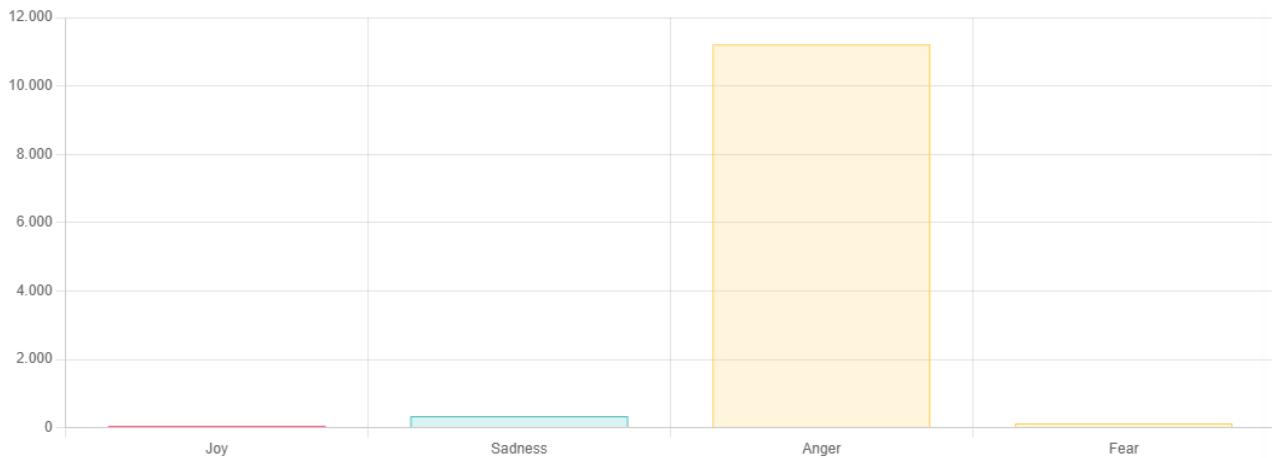
Utilizzando l'algoritmo “Feel-it” i tweet vengono così suddivisi:

- 11.701 (99.63%) tweet considerati negativi
- 0 (0%) tweet considerati neutrali
- 43 (0.37%) tweet considerati positivi



Classificando in base alle emozioni si è ottenuto che:

- 60 tweets esprimono gioia;
- 339 tweets esprimono tristezza;
- 11.281 tweets esprimono rabbia;
- 127 tweets esprimono paura.



## Russia: WordCloud

Word Cloud generata tramite tags presenti nei tweets:

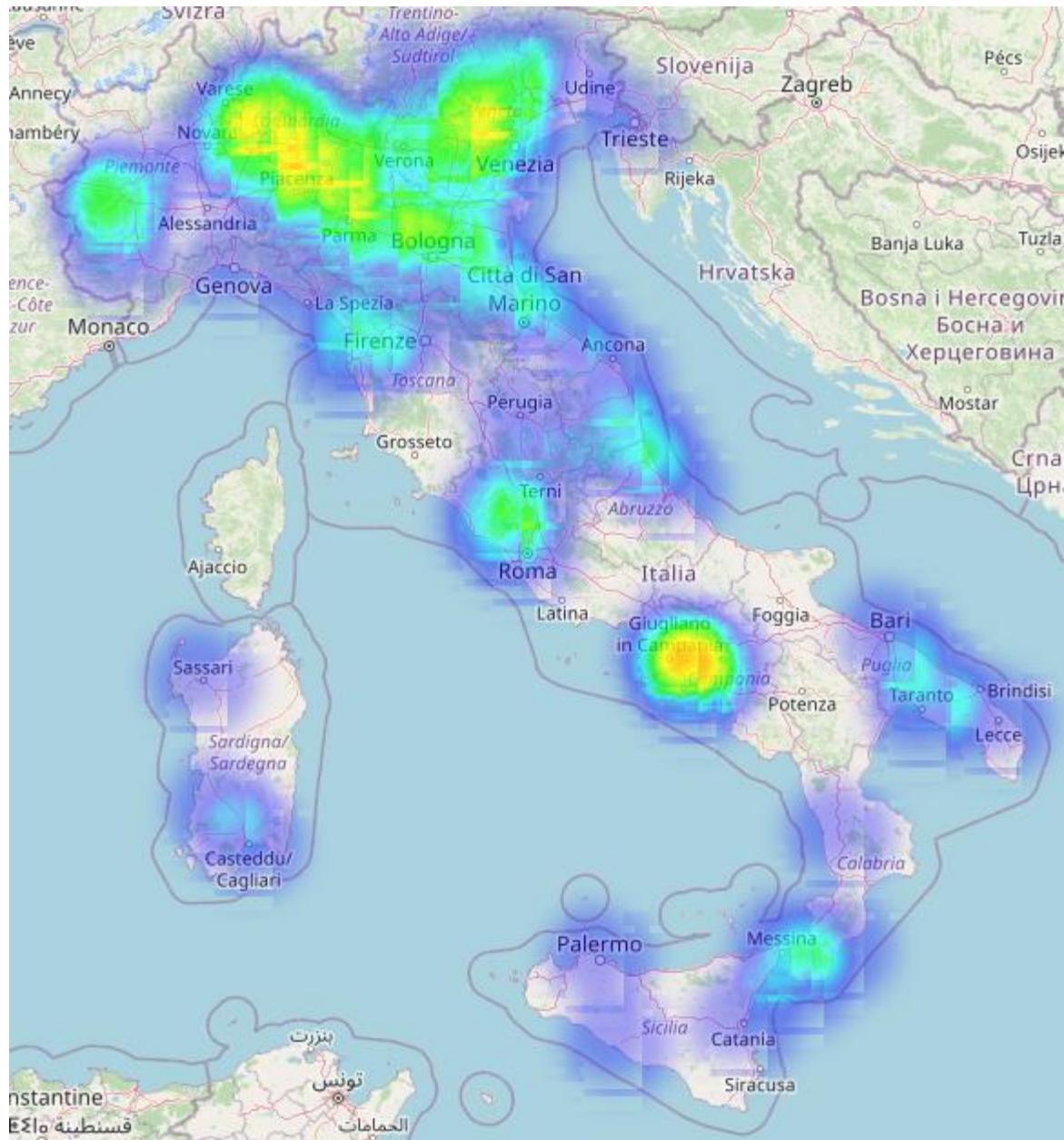
invasione barbarica Leopard diritti dell'uomo  
 Donetsk pacifista estinzione russofoni Olimpiadi di Mosca  
 avesse sovietico guerra BBC popolo ucraino  
 Ukraine esseri umani STORIA amato  
 quelli Dnipro criminale Genocidio occupato  
 ebrei Cinesi ch Russi accise Sorcio intelligenza  
 europei lanciano xenofoba sciacallo assassino  
 Libia arte russa USA governi puttana Slava  
 mattanza dittatori  
 Corte Penale internazionale Poveri slava amico complemento cremlino  
 capitale cattolico Mentana armi Ucraini  
 teatro La Fenice cadaveri sponda sinistra Nazisti  
 industria bellica idiota Beslan popolo russo  
 Hitler dittatore minoranza pandemia democratico macellai  
 sadomasochiste commissione parlamentare di inchiesta calce  
 proprio Paese impero coloniale europeo Putin diritti umani  
 Giulietto Chiesa truppe ucraini demagogo Stepan Bandera bambini  
 Franchi Presidente folle peggio  
 Zar terrorismo oggi gente Mondo! Non popolo Merkel Geograficamente  
 corrotti ucraino Europei mosca possa nazisti

Word Cloud generata tramite hashtags presenti nei tweets:



## Russia: Maps

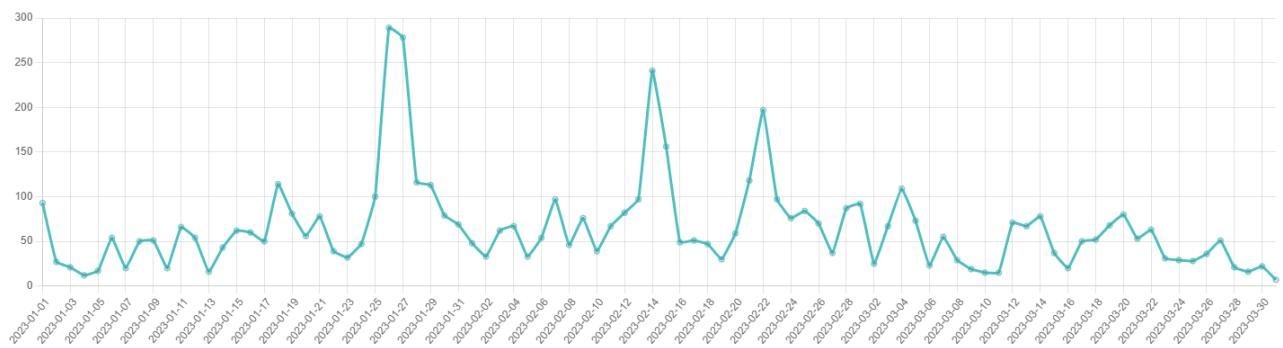
Dei 11.744 tweets ritrovati, 5010 sono stati geolocalizzati.



## 4.2.2 Ucraina

Da inizio gennaio a fine marzo 2023, sono stati ritrovati 5898 tweets d'odio nei confronti dell'ucraina.

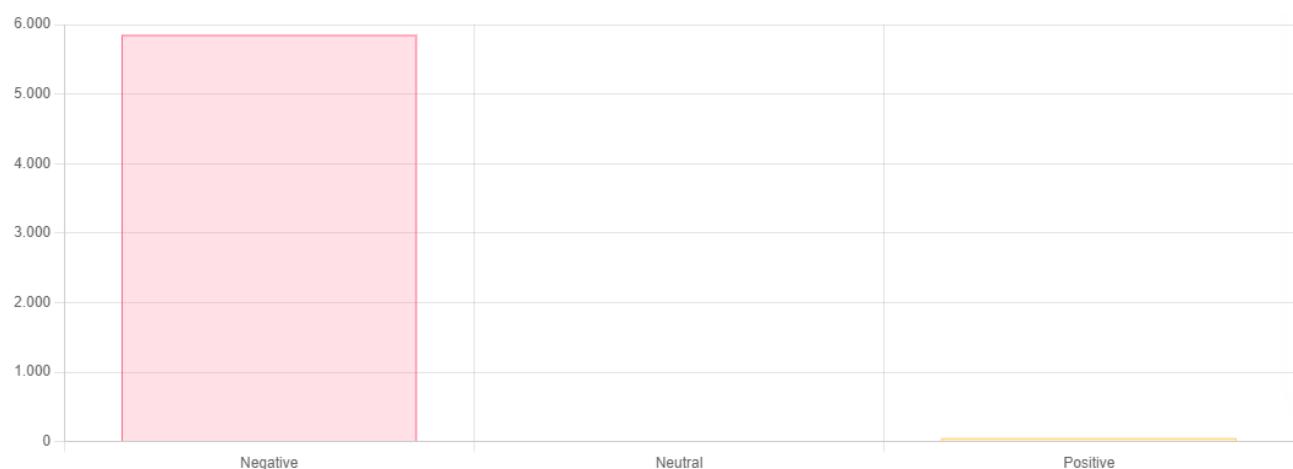
### Ucraina: TimeLine

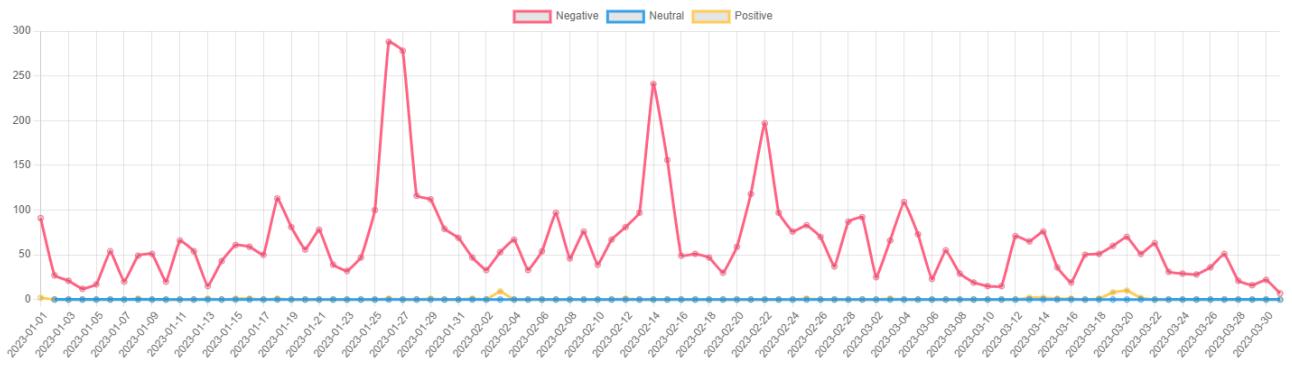


### Ucraina: Sentiment

Utilizzando l'algoritmo “Feel-it” i tweet vengono così suddivisi:

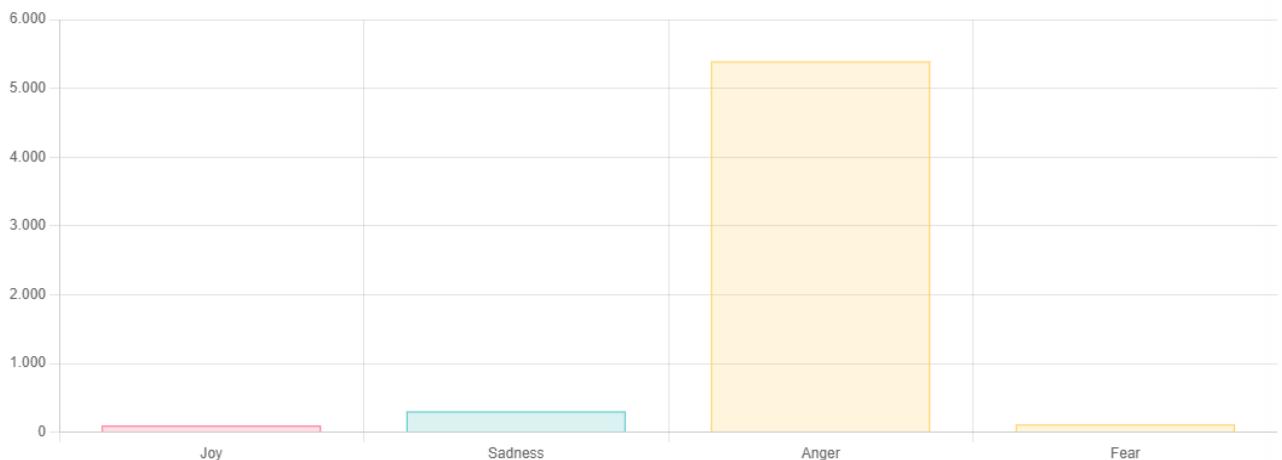
- 5.848 (99,15%) tweet considerati negativi
- 0 (0%) tweet considerati neutrali
- 50 (0,85%) tweet considerati positivi





Classificando in base alle emozioni si è ottenuto che:

- 96 tweets esprimono gioia;
- 300 tweets esprimono tristezza;
- 5.390 tweets esprimono rabbia;
- 112 tweets esprimono paura.



## Ucraina: WordCloud

Word Cloud generata tramite tags presenti nei tweets:

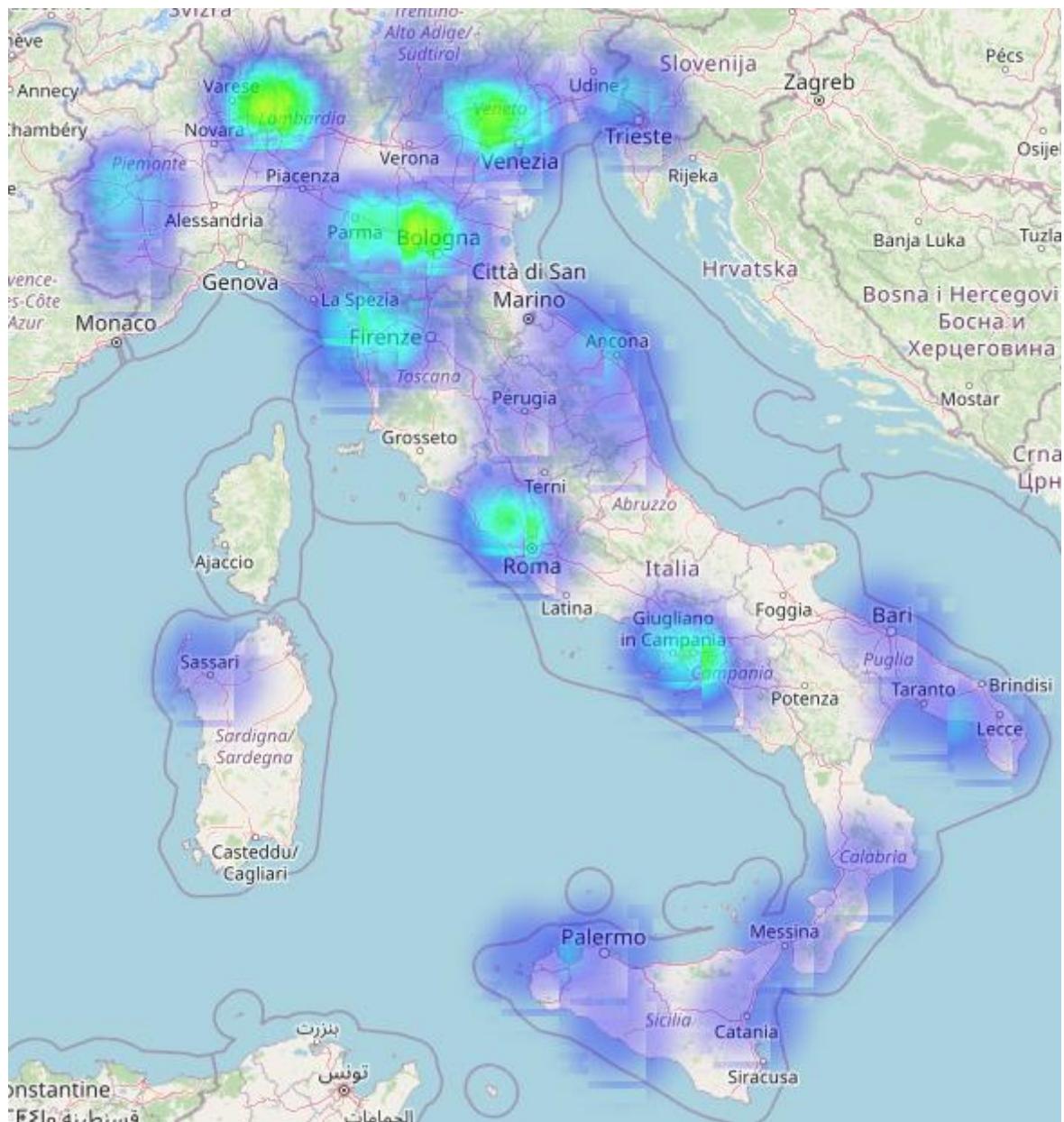


Word Cloud generata tramite hashtags presenti nei tweets:



## Ucraina: Maps

Dei 5.898 tweets ritrovati, 2.415 sono stati geolocalizzati.



## Sviluppi futuri

I possibili sviluppi futuri che possono essere attuati sui due tools sono diversi, ad esempio per quanto riguarda ITA si potrebbe estendere il software anche ad altre piattaforme social come Instagram e Facebook.

Inoltre, attualmente ITA è in grado di effettuare molte operazioni di elaborazione per il testo, ma poche per i media che possono essere presenti nei Tweet. Dunque, si potrebbero integrare modelli in grado di agire su di essi.

Per quanto riguarda CrowdPulse, si potrebbe realizzare una nuova sezione in cui si mettono a confronto i dati di diverse collezioni, in modo tale da velocizzare il processo di analisi.

## Bibliografia

### [1] Digital Report 2023

- <https://datareportal.com/reports/digital-2023-global-overview-report>

### [2] Global Web Index

- <https://www.gwi.com>

### [3] Caso Facebook-Cambridge Analytica

- [https://it.wikipedia.org/wiki/Scandalo\\_Facebook-Cambridge\\_Analytica](https://it.wikipedia.org/wiki/Scandalo_Facebook-Cambridge_Analytica)

### [4] California Consumer Privacy act 2018

- [https://en.wikipedia.org/wiki/California\\_Consumer\\_Privacy\\_Act](https://en.wikipedia.org/wiki/California_Consumer_Privacy_Act)

## [5] Social Media Analytics

- <https://www.digital4.biz/marketing/social-media-analytics-cos-e/>

## [6] Sentiment Analysis

- <https://www.insidemarketing.it/glossario/definizione/sentiment-analysis/>

## [7] Hate Tweet Map

- <https://darioamorosodaragona.gitlab.io/hatemap/index.html>

## [8] SENTIPOLC

- <http://www.di.unito.it/~tutreeb/sentipolc-evalita16/data.html>

## [9] Feel-it

- <https://huggingface.co/MilaNLProc/feel-it-italian-emotion>

## [10] SpaCy

- <https://spacy.io>

## [11] TagMe

- <https://tagme.d4science.org/tagme/>

## [12] Open Street Map

- <https://it.wikipedia.org/wiki/OpenStreetMap>

## [13] Geocoder

- <https://pypi.org/project/geocoder/>

## [14] React.js

- [https://it.wikipedia.org/wiki/React\\_\(web\\_framework\)](https://it.wikipedia.org/wiki/React_(web_framework))

## [15] Express.js

- <https://it.wikipedia.org/wiki/Express.js>

[16] Node.js

- <https://it.wikipedia.org/wiki/Node.js>

[17] MongoDB

- <https://it.wikipedia.org/wiki/MongoDB>

[18] HuggingFace

- <https://huggingface.co>

[19] Hate\_Speech\_it

- [https://huggingface.co/IMSyPP/hate\\_speech\\_it](https://huggingface.co/IMSyPP/hate_speech_it)

[20] Roberta

- [https://huggingface.co/docs/transformers/model\\_doc/roberta](https://huggingface.co/docs/transformers/model_doc/roberta)

[21] Vit-gpt2-image-captioning

- <https://huggingface.co/nlpconnect/vit-gpt2-image-captioning>

[22] COCO (Common Object in Context)

- <https://cocodataset.org>

[23] Mappa dell'intolleranza italiana

- S. Brena, C. Musto, G. Semeraro. Il progetto Mappa Italiana dell'Intolleranza. In AA.VV., La Rete e il fattore C – Competenze, Consapevolezze e Conoscenze. A cura di Sonia Montegiove, Emma Pietrafesa, Flavia Marzano, 431-446, Roma, 2014