

TWITTER SENTIMENT ANALYSIS IN ITALY DURING COVID-19

Progetto di Reti Geografiche: Struttura, Analisi,
Prestazioni

Docente

Delfina Malandrino

Componenti Gruppo

Antonio Pappalardo

Francesco Parisi

Domenico Rossi

A.A 2020/2021

Sommario

1	Studio dell'Arte.....	2
2	Studio della Fattibilità	3
2.1	Limitazioni.....	4
3	Introduzione al Problema	6
3.1	Obiettivi	6
4	Progettazione	7
4.1	Tecnologie ed Ambienti Utilizzati	7
4.1.1	Python	7
4.1.2	MongoDB	8
4.1.3	VueJS.....	9
4.2	Architecture Binding	9
5	Implementazione	10
5.1	Twitter Capture	10
5.2	Translate & Sentiment.....	14
5.3	Web App	17
6	Raccolta e Analisi dei Dati	28
6.1	Eventi Significativi.....	28
6.2	Pulizia dei Dati.....	30
7	Elaborazione dei dati	33
7.1	Visualizzazione Grafica dei Dati.....	33
7.2	Risposte al Problema	37
8	Conclusioni	43
9	Riferimenti.....	44

1 Studio dell'Arte

Dovendo implementare un progetto per l'esame di Reti Geografiche, che riguardasse tematiche inerenti al corso predetto, abbiamo formalizzato varie idee. Dai vari colloqui tenutisi in accordo con la professoressa Malandrino, la realizzazione è ricaduta sulla seguente idea:

Data la situazione attuale caratterizzata dal fenomeno pandemico del Covid-19, si vuole realizzare Sentiment Analysis, mediante Twitter, per capire ciò che la popolazione pensa a tal riguardo, ponendo un focus su come essa ha reagito in seguito alle varie decisioni governative (dpcm, etc.).

Attraverso Twitter, abbiamo deciso di catturare i tweets (con riferimento ai relativi retweets) con determinati hashtags. Per farlo è stato necessario creare degli appositi account su Twitter Developer, necessari per avere a disposizione le API dal social stesso. Successivamente, dopo varie riunioni, è stato deciso di creare una copia permanente dei tweets catturati mediante l'utilizzo di un database.

Per la visualizzazione dei risultati, che rappresentano il sentimento generale della popolazione, vi erano vari possibili sviluppi come:

- Applicativo Java
- Applicativo Python
- Foglio Excel
- Applicazione Web

Dopo un attento studio della fattibilità, la scelta è ricaduta sull'implementazione più articolata ma allo stesso tempo più user-friendly, così da permettere una visione dettagliata dei dati raccolti.

2 Studio della Fattibilità

Questa fase risulta essere particolarmente importante, in quanto ci ha permesso di prendere decisioni significative riguardanti lo sviluppo e le implementazioni future. Gli interrogativi che ci siamo posti sono i seguenti:

➤ *Quale Social media utilizziamo per l'analisi?*

- ✗ Facebook, Instagram: Tali social non mettono a disposizione API gratuite per la raccolta dei dati e statistiche.
- ✓ Twitter: Dispone di API gratuite, mediante Twitter Developer, che permettono di ottenere dati da analizzare per progetti accademici.

➤ *Quale linguaggio di programmazione utilizziamo per la cattura dei tweets?*

- ✓ Il linguaggio scelto è stato Python, in quanto abbiamo pensato fosse particolarmente utile per creare lo script per la cattura; utilizzando apposite librerie open source messe a disposizione, come ad esempio Tweepy, che si prestano perfettamente al contesto di interesse.

➤ *Come gestiamo i dati raccolti?*

- ✓ Per la gestione dei dati, abbiamo optato per un salvataggio persistente degli stessi, mediante l'utilizzo di un database non-relazionale, tale scelta progettuale è data dalla possibilità di avere record strutturalmente diversi tra loro, e che abbiano maggiore manutenibilità, rispetto ad un DB relazionale. Conseguentemente abbiamo creato cinque collezioni, ognuna delle quali rappresenta un intervallo temporale di cattura dei tweets . Inoltre, per una gestione ottimale sono stati aggiunti dei parametri allo schema di ogni record:
 - **Hashtag**: contenente l'hashtag utilizzato per la cattura
 - **Retweet**: contenente un contatore per il numero di retweets
 - **Hashtags**: contenete l'insieme degli hashtags presenti
 - **Sentiment**: contenente il valore associato al sentiment (range 1-5)
 - **Compound**: contenente un valore aggregato (positivo/negativo) a cui è associato il sentiment

➤ *Come andiamo a gestire i vari retweets?*

- ✓ Dopo un confronto, abbiamo ritenuto essere importante considerare i retweets. Tuttavia, per ottimizzare la fase di analisi e per evitare di occupare ulteriore memoria per dati ridondanti, abbiamo tenuto traccia degli stessi mediante un contatore associato ad ogni tweet.

➤ *Quale strumento utilizziamo per il Sentiment?*

- ✗ Inizialmente per poter effettuare il sentiment abbiamo valutato il possibile utilizzo di Azure Text Analytics. Dopo una prima implementazione di testing, abbiamo ritenuto che non fosse idoneo utilizzare tale servizio, in quanto la mole di dati da prendere in considerazione risultava essere particolarmente consistente. A questo si aggiungevano altre importanti limitazioni dettate principalmente dal piano di sottoscrizione del servizio di Microsoft.
- ✓ Una valida alternativa per effettuare il sentiment, non avendo limitazioni in termini di richieste, è *Vader Sentiment*. Questa libreria sfrutta un elenco di caratteristiche lessicali (come parole) in lingua inglese, e ad esse associa una serie di valori.

➤ *Come effettuiamo la traduzione dei tweets catturati?*

- ✗ Come prima ipotesi, per poter tradurre i tweets catturati, abbiamo pensato ad un'apposita libreria di Google Translate, che una volta ricevuto l'input effettua la traduzione. Dato il numero eccessivo di richieste, per poter effettuare la traduzione di tutti i tweets, necessitavamo di un proxy server funzionante.
- ✓ Disponendo di una VPN valida, ciò ha permesso di fixare il problema delle richieste limitate, quindi la scelta dello strumento di traduzione è ricaduta sull'utilizzo di TextBlob.

➤ *Come possiamo elaborare e visualizzare i dati raccolti?*

- ✓ Prese in considerazione le varie tecnologie utilizzabili per la visualizzazione dei risultati, di comune accordo, la scelta è ricaduta sullo sviluppo di un applicativo web. Tale scelta è stata dettata da una predisposizione maggiore alla visualizzazione di dati, utilizzando varie funzionalità e grafici.

2.1 Limitazioni

Nella realizzazione del progetto [1] vi è stata la presenza di svariate limitazioni, che hanno indirizzato i nostri sviluppi in modi dipendenti dalle stesse.

Le limitazioni avute hanno riguardato principalmente:

- **API Twitter Developer:** Le API fornite da Twitter [2] mettono a disposizione diverse funzionalità utili, ma con utilizzi limitati. È il caso, ad esempio della reperibilità dei dati nel passato, infatti nel nostro caso non risulta possibile acquisire tweets antecedenti a sette giorni dalla data corrente. Un'ulteriore limitazione è caratterizzata dal limite di velocità delle richieste al secondo e dall'estrazione mensile di tweets.

Di seguito vengono riportate le seguenti limitazioni:

TWEET LOOKUP		RECENT SEARCH	
Tweet cap ⓘ	500,000 Tweets / month PER PROJECT	Rate limit ⓘ	900 request / 15 min PER USER AUTH
Rate limit ⓘ	450 request / 15 min PER APP AUTH		300 request / 15 min PER APP AUTH
	180 request / 15 min PER USER AUTH		

Figura 1: Limitazione API Twitter

- **Azure Text Analytics:** Tale strumento [3], come detto in precedenza, è stato scartato in quanto presenta limitazioni dovute principalmente al piano di sottoscrizione ad Azure. Essendo un account "For Students" dà la possibilità di testare questo servizio, ma in maniera molto limitata. Nel nostro caso non poteva quindi essere impiegato dato il numero spropositato di tweets catturati, in quanto il limite fissato è di max 10 record per richiesta.
- **HTTP Error 429 - Too Many Requests:** Data la limitazione inerente alle troppe richieste in un determinato periodo di tempo, che genera un errore 429, per poter effettuare il sentiment dei tweets catturati abbiamo utilizzato una VPN. Sfruttando una prova gratuita mensile del software Seed4.me [4], abbiamo avuto a disposizione una serie di indirizzi IP associati a varie nazioni. Per fixare tale errore, dunque ogni volta che veniva superato il numero di richieste, si cambiava manualmente l'indirizzo IP e si rieseguiva nuovamente lo script. Ciò ha permesso di portare a termine tutte le richieste senza dover attendere i tempi stabiliti dalla limitazione.
- **Fattori Imponderabili:** Oltre alle varie limitazioni di tipo tecnico, sono state riscontrate limitazioni di tipo semantico/sintattico delle frasi presenti all'interno dei tweets. Tali limitazioni sono:
 - *Cogliere l'ironia e il sarcasmo*
 - *Tradurre i dialetti*
 - *Errori grammaticali e ortografici*
 - *Parole abbreviate ("xke, cmq, ntn, qst, nn, etc....")*





3 Introduzione al Problema

Come già detto in precedenza, il nostro progetto nasce con l'idea di analizzare e comprendere ciò che la popolazione italiana pensa rispetto alle decisioni governative in merito al tema del Covid-19. L'interesse per l'argomento ci ha spinto verso questa decisione progettuale, in quanto soprattutto in quest'ultimo periodo, si sta vivendo una situazione surreale, dove purtroppo ogni giorno si hanno notizie di contagi e decessi. Inoltre, le varie situazioni di lockdown, coprifuochi e limitazioni in zone, hanno evidentemente avuto un forte impatto sulla quotidianità di ogni individuo.

Per avere dati reali rappresentanti il pensiero della popolazione italiana è stato quindi deciso di fare Sentiment Analysis [5]. Essa consente di effettuare analisi delle interazioni tra utenti, stabilite in un determinato contesto ed in uno spaccato temporale definito. Nello specifico si tratta dell'analisi computazionale di sentimenti e opinioni espressi all'interno di testi generati in rete.

3.1 Obiettivi

Gli obiettivi individuati per la realizzazione progettuale sono:

-  **Individuazione dei principali hashtags di tendenza:** Si è proceduto ad individuare gli hashtags più utilizzati, riguardanti l'argomento Covid-19, valutando varie categorie di interesse.
-  **Cattura e memorizzazione dei tweets:** Si è proceduto settimanalmente a catturare i tweets, salvandoli successivamente in un database.
-  **Sentiment dei dati acquisiti:** Una volta ottenuti i dati, essi sono stati processati ottenendo così i valori associati a ciascun tweet.
-  **Studio ed elaborazione dei dati:** I dati finali sono stati studiati per ottenere risultati e risposte in merito al problema.

4 Progettazione

La metodologia utilizzata prevede di ottenere le opinioni mediante opportuni hashtags, in particolare si è proceduto ad acquisire tutti i tweets che facessero riferimento agli stessi. Sono stati individuati 20 hashtags di tendenza che appartenessero a varie categorie di nostro interesse e che rispettassero determinati parametri di popolarità.

Una volta stabiliti gli hashtags maggiormente diffusi e utilizzati, si è proceduto con la scrittura di uno script in Python che permetta l'acquisizione dei tweets entro un certo periodo di tempo stabilito, ciò è realizzabile utilizzando la libreria Tweepy [6]. Successivamente i dati acquisiti vengono resi persistenti e inseriti in un database non relazionale (MongoDB [7]), utilizzando l'apposita libreria PyMongo [8]. I tweets catturati vengono tradotti dall'italiano all'inglese tramite l'utilizzo di TextBlob [9]. Questa traduzione è eseguita utilizzando una VPN, che permette di effettuare un gran numero di richieste senza limiti. Si passa poi al sentiment analysis attraverso Vader Sentiment [10]. I dati raccolti e processati verranno analizzati e mostrati in una Web app, sviluppata utilizzando Vue.js [11] e Google Charts [12], per la creazione e la visualizzazione di grafici di vario tipo.

4.1 Tecnologie ed Ambienti Utilizzati

Le tecnologie individuate per la realizzazione degli obiettivi preposti sono diverse. Innanzitutto, tale progetto è stato sviluppato prendendo in considerazione tre fattori di interesse: un approccio computazionale, la persistenza dei dati e la rappresentazione delle informazioni raccolte. Ciò ha garantito una gestione agevole e mirata delle componenti, consentendone una realizzazione intuitiva e senza eventuali impedimenti. Di seguito vengono riportate le principali tecnologie utilizzate.

4.1.1 Python

Python è un linguaggio di programmazione dinamico orientato agli oggetti utilizzabile per molti tipi di sviluppo software. Offre un forte supporto all'integrazione con altri linguaggi e programmi, è fornito di una estesa libreria standard e può essere imparato in pochi giorni. Inoltre, Python è open source ed è possibile usarlo e distribuirlo senza restrizioni di copyright. Le caratteristiche principali sono:

- Portabilità
- Facilità d'uso
- Ricchezza di librerie
- Strutturalmente Performante
- Gestione automatica della memoria

- **Tweepy:** Una libreria Open Source che offre un'interfaccia per accedere alle API REST di Twitter, come ricerche e stream. Tweepy comprende una serie di classi e metodi che rappresentano i modelli di Twitter e gli endpoint API e gestisce in modo semplice vari dettagli di implementazione, come: Codifica e decodifica dei dati, Richieste http, Impaginazione dei risultati, Autenticazione OAuth e Flussi.
- **PyMongo:** PyMongo è una distribuzione Python contenente strumenti per lavorare con MongoDB. Essenzialmente questo strumento è stato utile per creare la connessione al DataBase.
- **Vader Sentiment:** Una libreria Open Source che consente l'analisi del sentiment, basandosi su regole lessicali e in sintonia con i sentimenti espressi nei social media. Vader sfrutta un elenco di caratteristiche lessicali (come parole) che sono solitamente etichettate in base al loro orientamento semantico come positive oppure negative. Vader non solo si limita ad individuare il punteggio di positività e negatività, ma verifica anche quanto un sentimento stesso sia negativo o positivo.
- **TextBlob:** TextBlob è una libreria Python per l'elaborazione di dati testuali. Fornisce una semplice API per immergersi in attività comuni di elaborazione del linguaggio naturale (NLP) come la codifica di parti del discorso, l'estrazione di frasi nominali, l'analisi del sentiment, la classificazione, la traduzione e altro ancora. Caratteristiche:
 - Estrazione di frasi sostantive
 - Etichettatura di parti del discorso
 - Analisi del sentiment
 - Classificazione (Naive Bayes, Albero decisionale)
 - Tokenizzazione (suddivisione del testo in parole e frasi)
 - Frequenze di parole e frasi
 - Analisi
 - Inflessione delle parole (pluralizzazione e singolarizzazione) e lemmatizzazione
 - Correzione ortografica
 - Traduzione

4.1.2 MongoDB

<< MongoDB è un database distribuito, document-based, per uso generico destinato agli sviluppatori di applicazioni moderne e al cloud >> (mongodb.com) .

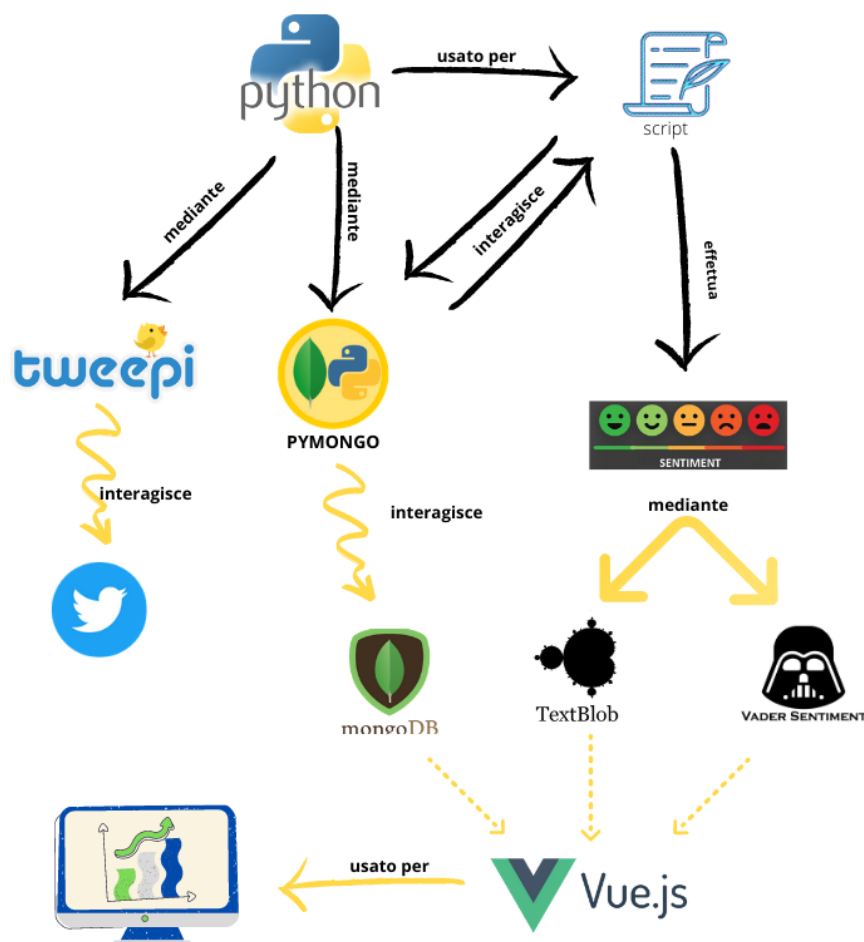
È un DBMS non relazionale, orientato ai documenti. Classificato come un database di tipo NoSQL, la caratteristica che lo rende "particolare" è che si allontana dalla struttura tradizionale basata su tabelle dei database relazionali, in favore di documenti in stile JSON con schema dinamico, rendendo l'integrazione di dati più facile e veloce.

4.1.3 VueJS

VueJS è un framework Javascript; Esso è uno dei principali framework per la creazione di applicazioni a singola pagina (SPA) inoltre è il framework più facile da imparare, dato che il creatore si è posto esattamente questo come obiettivo. Ci permette di realizzare anche applicazioni complesse. Inoltre, Vue è un framework progressivo, il che significa che alcune componenti, come il routing o il rendering server-side non sono inclusi di default. Tuttavia, Vue ha librerie ufficiali che si occupano di aggiungere le suddette ed altre funzionalità corredate da un'ottima documentazione.

4.2 Architecture Binding

Con la seguente mappa concettuale illustriamo come le varie tecnologie e le varie componenti interagiscono tra loro.



5 Implementazione

Questa fase è la più importante, in quanto è dove si implementano effettivamente le varie scelte progettuali, delle quali è stato parlato nei capitoli precedenti. In particolare, verrà spiegato come sono stati sviluppati gli Script e la Web-App.

Gli script sono stati realizzati in Python ed essenzialmente realizzano la parte di cattura dei tweets e la parte di traduzione e sentiment(back-end). Mentre la Web-App è stata realizzata con il framework Vue.js, per la visualizzazione dei dati raccolti (front-end).

5.1 Twitter Capture

Il primo script implementato è stato Twitter Capture. Esso risulta particolarmente importante poiché, attraverso la sua esecuzione, vengono catturati i vari tweets da Twitter e resi persistenti all'interno del database MongoDB. Tuttavia, in fase di cattura è stata innanzitutto svolta una prima pulizia dei tweets. Le stringhe ottenute infatti dovranno successivamente essere processate per le fasi di Translate e Sentiment e quindi con lo scopo di restituire come risultato stringhe quanto più chiare e pulite possibili.

Analizziamo nel dettaglio il comportamento dello script implementato:

```
### Formattazione Tweets
def cleanTweet(string):
    string = string.strip()
    string = string.replace('#', ' ')
    string = string.replace('\n', ' ')
    string = re.sub('(\@(\w)*: )|(\@(\w)*)*', '', string)
    string = re.sub('http://\S+|https://\S+', '', string)
```

La prima funzione di pulizia è `cleanTweet(string)`. Essa è stata implementata per pulire il campo Testo della struttura del record. La funzione prende in input il testo del tweet, sottoforma di stringa e formatta in un modo ben preciso. Dal testo vengono rimosse tutte le componenti non adatte, come il simbolo “#” e il carattere “\n”, questo utilizzando il metodo `replace()`, che permette di sostituire un carattere dato in input con una stringa vuota. Mediante delle espressioni regolari di `re.sub()` vengono poi trovate una serie di caratteri speciali, come ad esempio “@”. In tal caso, grazie a questa soluzione, tutte le stringhe che presentano un carattere iniziale @ e seguito da testo vengono rimossi dal tweet, apportando una pulizia importante per evitare la presenza di eventuali nomi utente e altri tag. Lo stesso metodo è stato applicato anche per la rimozione di link non utili alla cattura, facendo in modo che alla presenza di stringhe con la dicitura “http” o “https” queste vengano rimosse.

```

### Formattazione Hashtags
def cleanHashtags(string):
    string = string.replace('\n', '')
    string = re.sub('(\@(\w)*: )|(\@(\w)*)*', '', string)
    string = re.sub('http://\S+|https://\S+', '', string)

```

Questa seconda funzione , chiamata `cleanHashtags(string)`, serve a pulire il campo Hashtag nella struttura del record. La funzione prende in input nuovamente una stringa, quindi il tweet , ed anche in questo caso con una serie di espressioni regolari si effettua una profonda pulizia. È importante segnalare che in questa fase è stata prevista anche la rimozione delle emoticons, mediante apposita espressione regolare, come mostrato nel codice:

```

emoji_pattern = re.compile("[
    u"\U0001F600-\U0001F64F"
    u"\U0001F300-\U0001F5FF"
    u"\U0001F680-\U0001F6FF"
    u"\U0001F1E0-\U0001F1FF"
    u"\U00002500-\U00002BEF"
    u"\U00002702-\U000027B0"
    u"\U00002702-\U000027B0"
    u"\U000024C2-\U0001F251"
    u"\U0001f926-\U0001f937"
    u"\U00010000-\U0010ffff"
    u"\u2640-\u2642"
    u"\u2600-\u2B55"
    u"\u200d"
    u"\u23cf"
    u"\u23e9"
    u"\u231a"
    u"\ufe0f"
    u"\u3030"
    "]" + "", flags=re.UNICODE)
string = emoji_pattern.sub(r'', string)
return string

```

Lo script corrente, come detto, risulta essere importante in quanto dopo la cattura, deve rendere persistenti i dati sul database. È in questa fase che quindi si crea la connessione al Db, tramite PyMongo:

```

### Database Connection
client=
MongoClient("mongodb+srv://dbUser:dbUser@cluster0.lcwhz.mongodb.net/Pr
ogReti?retryWrites=true&w=majority")
db=client['ProgReti']
collection=db['Week-1']

```

Il costruttore della classe Mongo Client prende in input, come parametro, l'indirizzo del DB. Una volta creata la connessione, si accede al DataBase, con la seguente sintassi : `db = client["nome db"]` in cui l'oggetto client , ottenuto dalla creazione del db, riceve il nome del database. Dunque, la variabile db permetterà di agire sul DataBase specificato. Creata la connessione, si può operare senza alcun problema. Per poter estrapolare i dati da Twitter, tuttavia, sono necessarie delle identificazioni lato utente. In particolare, tramite le API di Twitter Developer, vengono rilasciate delle chiavi univoche, che permettono , in base all' utilizzo richiesto, di poter effettuare specifiche operazioni. Dal seguente codice è possibile verificare le credenziali per l'accesso e la conseguente autenticazione a Twitter:

```
#### Twitter Developer Credentials
consumerKey = '4SUob4rkVJ3udNkdoWhgWB8w6'
consumerSecret = 'gQHBuXo0Amw6fTlt1CwJp8xiIqnQxxgjyftCQnBZz0nBUmqbBi'
accessToken = '1331549356277817344-w4vmuKtf6W6qCcwvaNyqiqdvCjCILh'
accessTokenSecret = 'kc3IqpTz3h0gzSny9MtdVuYVnQgoPXS2dZ0wZo48lYyRO'

auth = tweepy.OAuthHandler(consumerKey, consumerSecret)
auth.set_access_token(accessToken, accessTokenSecret)
api = tweepy.API(auth,wait_on_rate_limit=True)
```

Dopo che tutti gli strumenti necessari sono stati resi utilizzabili in modo corretto, si procede all' effettiva cattura dei Tweets mediante la seguente parte di codice:

```
data = {"documents": []}

# Inserimento Hashtag
inputTerm = input("Enter Keyword/Tag to search about: ")
searchTerm = inputTerm + " -filter:retweets"

print("Start Tweet Capturing..")
for tweet in tweepy.Cursor(api.search,q=searchTerm, lang="it",
since="2020-11-27", until="2020-12-03", result_type="mixed",
tweet_mode="extended").items(1000):
    rt_count = tweet.retweet_count
    print(tweet.created_at, tweet.full_text)
    p2 = ["", tweet.full_text]
    hast = ""
    a = tweet.full_text.count("#")
    if (a >= 0):

        for c in range(a):
            p1 = p2[1].split("#", 1)
            try:
                if (p1[1].find(" ") < 0):
                    hast += "#" + p1[1]
                    hast = hast.replace("\n", "")
                    break
            else:
                p2 = p1[1].split(" ", 1)
```

```

        hast += "#" + p2[0]
        hast = hast.replace("\n", "")
    except IndexError:
        hast += "#" + p1[0]
    ddata=datetime.datetime.strptime(str(tweet.created_at), '%Y-%m-%d
%H:%M:%S')
    tweet.full_text = cleanTweet(tweet.full_text)
    hast = cleanHashtags(hast)
    tt={
        "Id": tweet.id,
        'Hashtag': inputTerm,
        "Retweet": rt_count,
        "Data":str(ddata),
        "Testo":tweet.full_text,
        "Hashtags": hast,
        "Sentiment": 0,
        "Compound": 0
    }
    #data['documents'].append({"language": "it", "id": tweet.id, "text":
tweet.full_text, "Retweet": rt_count})
    collection.insert_one(tt)

```

La porzione di codice appena mostrata permette di inserire da tastiera una parola chiave, corrispondente all'hashtag da utilizzare per la ricerca dei tweets. Utilizzando dunque le API, viene effettuata la funzione di search principalmente per i tweets scritti in italiano che hanno all'interno la parola chiave scelta. Per ovviare al problema dei retweets è stata utilizzata una variabile `searchTerm` al cui interno è stato passato l'hashtag immesso e il filtro `"-filter:retweets"` necessario per evitare che, durante la cattura, vengano salvate più copie di uno stesso tweet. Le API, come detto nel capitolo delle limitazioni, ci impongono di poter catturare solo quei tweets scritti non più di sette giorni prima dalla data corrente. Abbiamo quindi inserito la possibilità di poter specificare l'intervallo temporale per la cattura, utilizzando i parametri `since` e `until`, così da recuperare in una sola cattura i tweets dell'intera settimana, mentre con `items()` si è specificato il numero totale di tweets da catturare. Infine, ci sono una serie di controlli degli errori e di richiami alle funzioni di pulizia, fino ad arrivare a salvare i tweets nella collezione corrispondente, seguendo la struttura specificata.

5.2 Translate & Sentiment

Parte cruciale del progetto è stata quella di ottenere il sentiment dai tweets catturati, in modo da stabilirne l'opinione pubblica riscontrata nel periodo analizzato. Dal momento che la cattura ha visto esclusivamente tweets di nazionalità italiana, ciò ha imposto come vincolo la traduzione degli stessi nella lingua inglese affinché Vader Sentiment ne potesse restituire un risultato valido. Affinché la realizzazione del progetto proseguiva in maniera efficiente e senza inconvenienti, si è pensato di seguire una metodologia che permettesse di automatizzare tutte le azioni necessarie allo sviluppo.

Innanzitutto, l'idea è stata quella di catturare prima i tweets, renderli persistenti sul database di MongoDB, e successivamente eseguire le operazioni di traduzione e sentiment, questo principalmente per evitare eventuali errori di cattura derivanti ad esempio da un inserimento di una data errata oppure per errori di troppe richieste (Error 429). Nello script in Python, dopo aver effettuato il collegamento con MongoDB alla collezione creata, si è proceduto come segue:

```
analyzer = SentimentIntensityAnalyzer()
hashtag = input("Inserisci un Hashtag per avviare la procedura: ")
count = 0
```

In particolare, per prima cosa sono state istanziate diverse variabili: `analyzer`, a cui viene passato l'oggetto `SentimentIntensityAnalyzer()` e necessario per il sentiment; la variabile `hashtag` che permette allo stesso utente di inserire da terminale l'hashtag per l'avvio delle operazioni; la variabile `count` che fa da contatore, ritornando il numero totale dei tweets elaborati.

Successivamente è stato utilizzato un ciclo `for` per poter controllare ciascun tweet della collezione. Questa procedura non è globale ma funziona per il solo l'hashtag immesso, quindi per evitare un numero eccessivo di richieste, tale operazione viene svolta per il solo hashtag passato. Tale vincolo è presente nel seguente codice:

```
for x in collection.find({ "Hashtag":hashtag, "Sentiment":0}):
    tweet = ("Testo:", x["Testo"])
    translate = translatorTweet(tweet)
```

Dal momento che viene preso in considerazione un hashtag per volta, nel ciclo `for` è stato utilizzato il metodo `find()`, fondamentale per ricercare un determinato parametro di un record della collezione. La ricerca dei tweets è stata quindi eseguita in riferimento all'hashtag inserito da terminale, effettuando un controllo che verificasse il valore di sentiment, questo per far sì che un tweet già analizzato non sia preso nuovamente in considerazione. Una volta ottenuto il testo del tweet e passato alla variabile `tweet`, si procede con l'operazione di traduzione. Per essa è stata utilizzata la libreria `TextBlob` che, a differenza di Google Translate, ci ha permesso di effettuare la traduzione dei tweets anche con un alto numero di richieste, seppur con limitazioni. Pur avendo la possibilità di effettuare un gran numero di richieste al secondo, fondamentale è stato l'utilizzo di una

VPN, in particolare Seed4.me, che ci ha consentito di evitare in tal caso problemi relativi alla velocità e numero delle richieste, semplicemente cambiando indirizzo IP e passando da una nazione all'altra evitando ulteriori rallentamenti ed errori.

Prima di poter tradurre ciascun tweet, per facilitarne soprattutto la comprensione del codice, è stata implementata una funzione apposita:

```
### Tweet Translator
def translatorTweet(tweet):
    blob = TextBlob(str(tweet))
    translate = blob.translate(from_lang='it',to='en')
    return translate
```

Con la funzione `translatorTweet(tweet)` si è fatto in modo che, passando come parametro il campo di testo del tweet, venga inizializzata una variabile `blob` in cui al suo interno vi è l'oggetto `TextBlob` e il parametro `tweet`. Fatto questo si sceglie la lingua che viene data in input a `textblob` e quella di output. Il valore di ritorno della funzione è `translate`, ovvero la stringa di testo tradotta.

```
for x in collection.find({"Hashtag":hashtag,"Sentiment":0}):
    tweet = ("Testo:", x["Testo"])
    translate = translatorTweet(tweet)
    sentiment = analyzer.polarity_scores(str(translate))
    compound = sentiment["compound"]
    sentimentAnalysis = Sentiment(compound)
    query = {
        "Id":x["Id"],
        "Sentiment":sentimentAnalysis,
        "Compound": compound
    }
    collection.update_one({"_id":x["_id"]},{
        "$set":{
            "Sentiment":sentimentAnalysis,
            "Compound":compound
        }
    })
    count += 1

print("Total Tweets with {} Updated Success:{}".format(hashtag,count))
```

Una volta eseguita la traduzione, nello stesso script viene avviata la procedura di sentiment. Esso è ottenuto utilizzando l'apposita variabile `sentiment`, a cui viene passato `polarity_scores(str(translate))`. Tale funzione, richiamata dalla libreria di Vader, permette di fornire come output i punteggi relativi a quattro classi di sentiment:

- neg: Negativo
- neu: Neutro
- pos: Positivo
- compound: Composto

polarity_scores() permette quindi di confrontare il testo dato in input con un dizionario che mappa le caratteristiche lessicali con le intensità delle emozioni, dove il punteggio è ottenuto sommando l'intensità di ogni parola nel testo. Una volta ricavati i punteggi delle classi di sentiment, abbiamo preso in considerazione la classe compound. Il motivo di tale scelta è dovuto al fatto che compound è un punteggio composto di tutte le classi di sentiment e che calcola la somma di tutte le valutazioni lessicali, assumendo un valore che va da -1 a 1. Come da codice mostrato in precedenza, è stata implementata anche una funzione specifica per il sentiment:

```
def Sentiment(compound):
    if(compound <= -0.5501 and compound >= -1):
        sentimentAnalysis = 1
    if(compound <= -0.2001 and compound >= -0.5500):
        sentimentAnalysis = 2
    if(compound >= -0.2000 and compound <= 0.2000):
        sentimentAnalysis = 3
    if(compound >= 0.2001 and compound <= 0.5500):
        sentimentAnalysis = 4
    if(compound >= 0.5501 and compound <= 1):
        sentimentAnalysis = 5
    return sentimentAnalysis
```

Generalmente in questa funzione viene fatta una suddivisione del compound in diverse categorie, in modo da avere ulteriori classi di sentimento oltre quelle di base (positive, neutral, negative). La classificazione è stata implementata dopo aver fatto un certo numero di test, perfezionandone di volta in volta il range, e quindi consapevoli del risultato generato. I primi test sono stati effettuati su una singola parola, per iniziare a capire l'incidenza del valore associato alla variabile compound. I successivi test sono stati caratterizzati da brevi frasi per capire, in questo caso, quanto l'output desse i risultati voluti, basandoci su parole appartenenti inequivocabilmente a una classe di sentiment specifica (negativo: *"assolutamente no!"*; positivo: *"è stupendo"*). Aumentando poi la complessità della frase, abbiamo potuto constatare la correttezza dell'output dato, rispetto ai limiti delle varie categorie. In totale quindi sono state individuate cinque classi di sentimento:

- Negativo
- Tendente negativo
- Neutro
- Tendente positivo
- Positivo

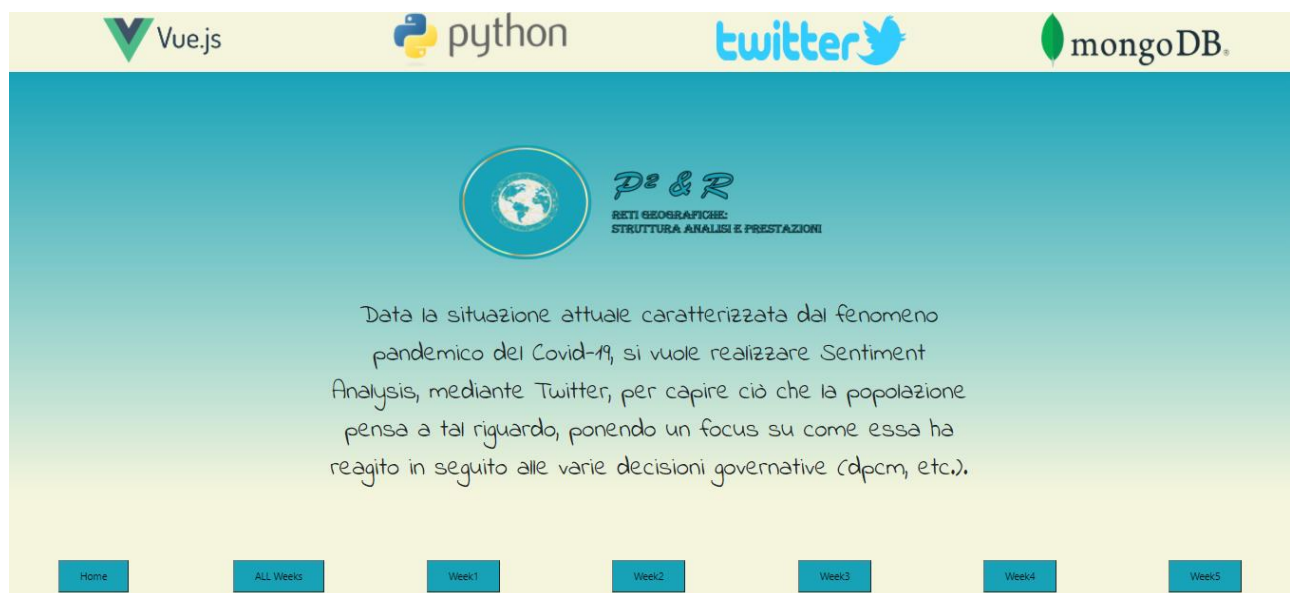
Una volta passato il compound alla funzione `Sentiment()` viene verificato se tale valore corrisponde ad una delle precedenti classi di sentimento, ritornando un valore intero `sentimentAnalysis` che va da 1 per un tweet negativo fino a 5 per uno positivo. La scelta di utilizzare un valore di tipo intero è dovuta principalmente al fatto che è più pratico ed efficiente manipolare interi piuttosto che stringhe.

Successivamente, una volta chiamata la funzione `Sentiment(compound)` e ottenuto il valore intero, si è proceduto con l'aggiornamento dei campi `Sentiment` e `Compound` di ciascun record nel database. Questa operazione è stata svolta grazie a `collection.update_one()`, che passando come valore chiave l'id del tweet, permette di andare ad aggiornare i singoli campi per ogni tweet. Come feedback finale, da terminale viene mostrato all'utente il numero totale dei tweets processati.

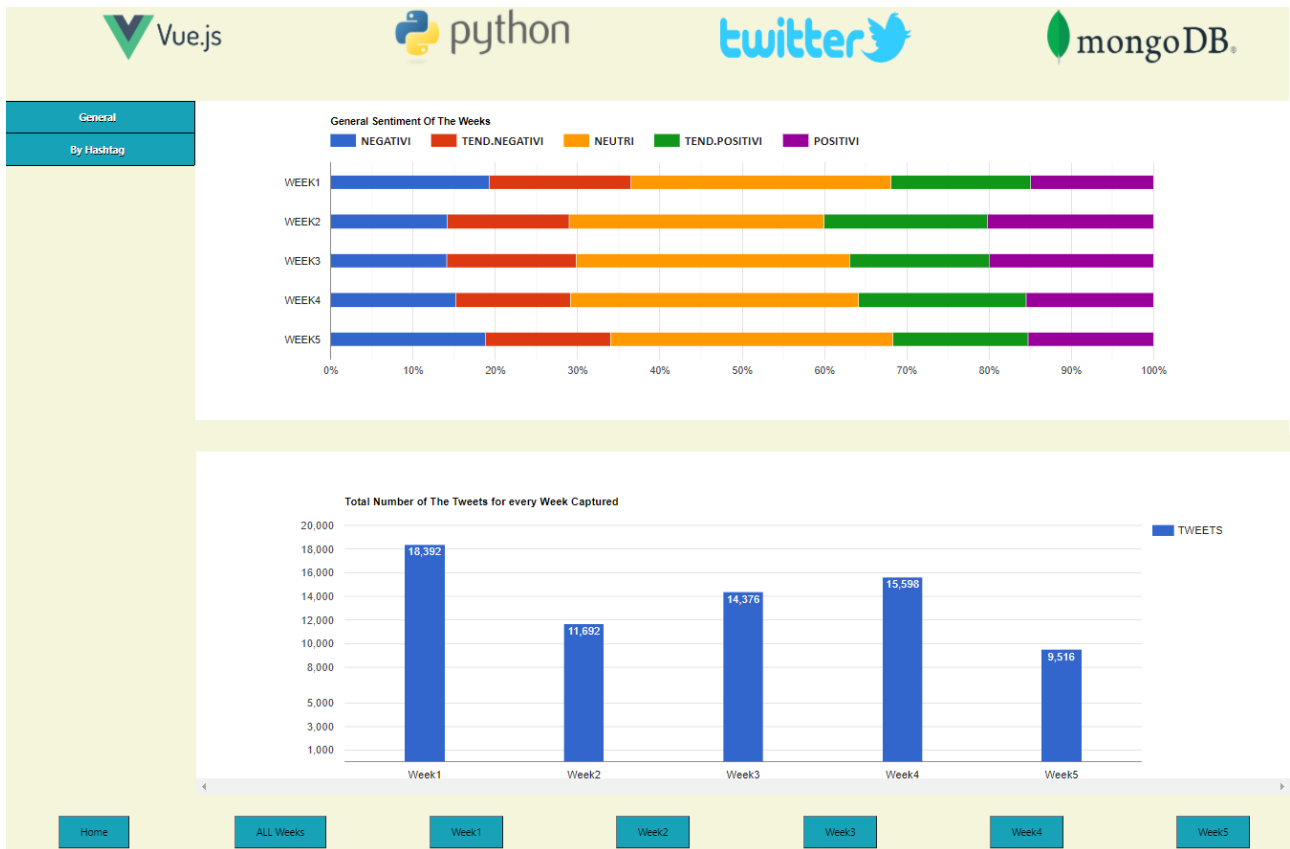
5.3 Web App

Per la visualizzazione dei dati, mediante Web-App, abbiamo utilizzato il framework `Vue.js`. Di conseguenza, è stata creata un'applicazione a singola pagina. Inoltre, per una visualizzazione più ordinata abbiamo utilizzato `Sass`.

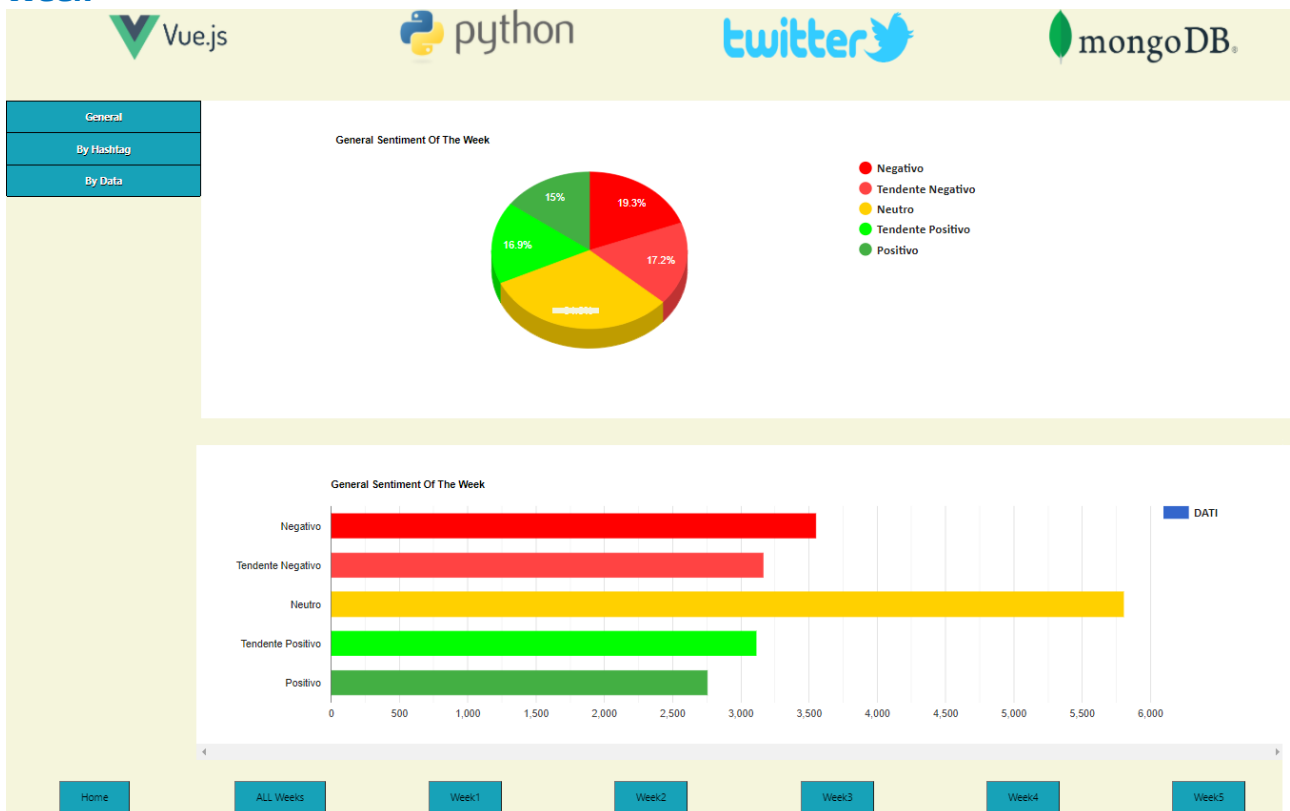
Home



All Weeks



Week



App.vue

Pagina iniziale della Web App; composta da un componente header, da un body e da un footer.

il body è composto da un div che contiene il tag <router-view> che permette di utilizzare il router, strumento base per app a singola pagina.

```
<template>

  <div id="app">

    <div class="header">
      <Header/> //Componente vue
    </div>
    <div class="view">
      <router-view/> //Utilizzo di vue-router
    </div>
    <div class="footer">
      <div class="selection"> //links alla visualizzazione della componente
        <button class="hvr-outline-in"><router-link to="/" class="item">Home</router-link></button>
        <button class="hvr-outline-in"><router-link to="/All" class="item">ALL Weeks</router-link></button>
        <button class="hvr-outline-in"><router-link to="/week1" class="item">Week1</router-link></button>
        <button class="hvr-outline-in"><router-link to="/week2" class="item">Week2</router-link></button>
        <button class="hvr-outline-in"><router-link to="/week3" class="item">Week3</router-link></button>
        <button class="hvr-outline-in"><router-link to="/week4" class="item">Week4</router-link></button>
        <button class="hvr-outline-in"><router-link to="/week5" class="item">Week5</router-link></button>
      </div>
    </div>
  </div>
</template>

<script>
import Header from './components/Header'
import { store } from './store/index' //utilizzo dello store
export default {
  name: 'App',
  store,
  components: {
    Header
  }
}
```

Store.JS

File javascript che rappresenta lo store dell'applicazione, ovvero si occupa della gestione dei dati. All'interno sono state implementate varie funzioni per la corretta visualizzazione dei dati.

Per prima cosa sono stati importati i file json estrapolati dal db;

```
//import dei file json dove sono salvati i dati
import Week1 from '../JSON/Week-1.json'
import Week2 from '../JSON/Week-2.json'
import Week3 from '../JSON/Week-3.json'
import Week4 from '../JSON/Week-4.json'
import Week5 from '../JSON/Week-5.json'
```

Prima di aver creato lo state, campo chiave dello store, che rappresenta i dati , abbiamo creato una costante Hashtag che contenesse tutti gli hashtag di studio.

```
//Costante che tiene traccia degli hashtag utilizzati
const Hashtag=["#andratuttobene", "#natale2020", "#immuni", "#iorestoacasa", "#dpcm", "#zonarossa", "#zonaarancione",
               "#zonagiulla", "#giuseppeconte", "#nolockdown", "#LItaliaSiRibella", "#lockdownitalia", "#coronavirus",
               "#congiuntifuriregione", "#vaccinocovid", "#sanità", "#secondaondata", "#covidioti", "#dad", "#mascherine"]
export const store = new Vuex.Store({ //creazione dello store
  state: { //lo state rappresenta i dati
    Week1,
    Week2,
    Week3,
    Week4,
    Week5,
    Hashtag
  },
```

Lo store gestisce varie funzioni, in particolar modo si suddividono in getters, mutation e action;

In questo specifico caso, non dovendo modificare nessun dato, sono state utilizzate solo getters. Sono state previste 7 getters, per poter visualizzare correttamente i grafici.

Avendo tenuto traccia dei retweets mediante un contatore associato ad ogni tweet, data la presenza di fattori imponderabili (ironia, sarcasmo, etc.) e non potendo analizzare un eventuale confutazione del tweet, abbiamo così deciso di utilizzare un fattore di popolarità che bilancerà in base alla tendenza di una data settimana il numero dei retweets, tramite la funzione getCategory(). Inoltre, attraverso la funzione getRetweetMultiplier(), gli hashtag sono stati suddivisi in quattro sottogruppi in base alla popolarità settimanale. Ogni hashtag appartenente al rispettivo sottogruppo è stato moltiplicato per una costante. In dettaglio, sono state previste quattro costanti (0.25, 0.50, 0.75, 1.0). Gli hashtag appartenenti al gruppo con una popolarità più alta sono stati moltiplicati per la costante minore; al decrescere della popolarità la costante aumenta. In sintesi, si bilancia il rapporto tra retweets privi di commenti e retweets commentati. Questa scelta implementativa deriva dal fatto che, analizzando e studiando i dati, abbiamo notato che al crescere della tendenza vi erano più retweets con fattori imponderabili.

```
getters:{
```

```
//FUNZIONE PER GENERARE L'ORDINE DEGLI HASHTAG + UTILIZZATI CHE PRENDE IN INPUT LA SETTIMANA
```

```

getcategory:(state)=>Week=>{
},

// FUNZIONE CHE PRENDE IN INPUT LA SETTIMANA E LA RACCOLTA DEI TWEETS,
// E PRODUCE IL NUMERO DEI RETWEET
getRetweetMultiplier:(state, getters)=>(tweets, week)=>{
    let st= tweets
    let multiplier=getters.getcategory(week); //CHIAMATA ALLA FUNZIONE PER LA POPOLARITA'
//ORA SI FILTRA LA LISTA DEI TWEET PER IL SENTIMENT
    let Negative= st.filter(a=> a.Sentiment===1)
    let TendNegative= st.filter(a=> a.Sentiment===2)
    let Neutre= st.filter(a=> a.Sentiment===3)
    let TendPositive= st.filter(a=> a.Sentiment===4)
    let Positive= st.filter(a=> a.Sentiment===5)

    let i=0;
    let neg=0;
    let tendneg=0;
    let neu=0;
    let tendpos=0;
    let pos=0;

    //HASHTAG PIU' POPOLARI, DI CONSEGUENZA FATTORE MOLTIPLICATIVO=0,25
    for(i=0;i<5;i++){
        neg=neg+(Negative.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.25)
        tendneg=tendneg+(TendNegative.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.25)
        neu=neu+(Neutre.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.25)
        tendpos=tendpos+(TendPositive.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.25)
        pos=pos+(Positive.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.25)
    }
    //FATTORE MOLTIPLICATIVO=0,5
    for(i=5;i<10;i++){
        neg=neg+(Negative.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.5)
        tendneg=tendneg+(TendNegative.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.5)
        neu=neu+(Neutre.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.5)
        tendpos=tendpos+(TendPositive.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.5)
        pos=pos+(Positive.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.5)
    }
    //FATTORE MOLTIPLICATIVO=0,75
    for(i=10;i<15;i++){
        neg=neg+(Negative.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.75)
        tendneg=tendneg+(TendNegative.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.75)
        neu=neu+(Neutre.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.75)
        tendpos=tendpos+(TendPositive.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.75)
        pos=pos+(Positive.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) *0.75)
    }
    //FATTORE MOLTIPLICATIVO=1
    for(i=15;i<20;i++){

```

```

    neg=neg+(Negative.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) )
    tendneg=tendneg+(TendNegative.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0))
    neu=neu+(Neutre.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0) )
    tendpos=tendpos+(TendPositive.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0))
    pos=pos+(Positive.filter(a=> a.Hashtag===multiplier[i]).map(a=> a.Retweet).reduce((a,b)=>a+b,0))

  }
  return[Math.floor(neg),Math.floor(tendneg),Math.floor(neu),Math.floor(tendpos),Math.floor(pos)]
},

```

Funzioni per la visualizzazione dei dati globali:

Le funzioni previste per la visualizzazione globale sono 2. La prima, ovvero la funzione `getCompareChart()` filtra le 5 collezioni per gli hashtag dati in input.

Per prima cosa crea un array composto da espressioni regolari, in base agli hashtag che ignorano il case; poi si filtra ciascuna collezione per gli hashtag;

```

getCompareChart:(state, getters)=>(hashtags)=>{
  let re=[]
  for(i=0;i<hashtags.length;i++){
    re[i]= new RegExp(hashtags[i], "i")
  }
  let st1=state.Week1.filter(tweet=>{
    for (var i=0; i<re.length; i++)
      if(tweet.Hashtags.match(re[i])) return true;
    return false;
  })

  let st2=state.Week2.filter(tweet=>{
    for (var i=0; i<re.length; i++)
      if(tweet.Hashtags.match(re[i])) return true;
    return false;
  })

  let st3=state.Week3.filter(tweet=>{
    for (var i=0; i<re.length; i++)
      if(tweet.Hashtags.match(re[i])) return true;
    return false;
  })

  let st4=state.Week4.filter(tweet=>{
    for (var i=0; i<re.length; i++)
      if(tweet.Hashtags.match(re[i])) return true;
    return false;
  })

  let st5=state.Week5.filter(tweet=>{
    for (var i=0; i<re.length; i++)
      if(tweet.Hashtags.match(re[i])) return true;
    return false;
  })

```

```
}}
```

Dopo aver filtrato, viene eseguito il calcolo dei retweets per ogni collezione e si invia in output un array contenente due array: uno per la visualizzazione del sentiment di ogni settimana(stweeks) ed uno per il sentiment generale(stweek).

```
let retweet1=getters.getRetweetMultiplier(st1,1)
let retweet2=getters.getRetweetMultiplier(st2,2)
let retweet3=getters.getRetweetMultiplier(st3,3)
let retweet4=getters.getRetweetMultiplier(st4,4)
let retweet5=getters.getRetweetMultiplier(st5,5)

let stweeks=[
  ["settimana", "NEGATIVI", "TEND.NEGATIVI", "NEUTRI", "TEND.POSITIVI", "POSITIVI",{role:'annotation'}],
  ["WEEK1",st1.filter(a=>a.Sentiment===1).length+retweet1[0],
  st1.filter(a=>a.Sentiment===2).length+retweet1[1],
  st1.filter(a=>a.Sentiment===3).length+retweet1[2],
  st1.filter(a=>a.Sentiment===4).length+retweet1[3],
  st1.filter(a=>a.Sentiment===5).length+retweet1[4],
  ''],
  ["WEEK2",st2.filter(a=>a.Sentiment===1).length+retweet2[0],
  st2.filter(a=>a.Sentiment===2).length+retweet2[1],
  st2.filter(a=>a.Sentiment===3).length+retweet2[2],
  st2.filter(a=>a.Sentiment===4).length+retweet2[3],
  st2.filter(a=>a.Sentiment===5).length+retweet2[4],
  ''],
  ["WEEK3",st3.filter(a=>a.Sentiment===1).length+retweet3[0],
  st3.filter(a=>a.Sentiment===2).length+retweet3[1],
  st3.filter(a=>a.Sentiment===3).length+retweet3[2],
  st3.filter(a=>a.Sentiment===4).length+retweet3[3],
  st3.filter(a=>a.Sentiment===5).length+retweet3[4], ''],
  ["WEEK4",st4.filter(a=>a.Sentiment===1).length+retweet4[0],
  st4.filter(a=>a.Sentiment===2).length+retweet4[1],
  st4.filter(a=>a.Sentiment===3).length+retweet4[2],
  st4.filter(a=>a.Sentiment===4).length+retweet4[3],
  st4.filter(a=>a.Sentiment===5).length+retweet4[4], ''],
  ["WEEK5",st5.filter(a=>a.Sentiment===1).length+retweet5[0],
  st5.filter(a=>a.Sentiment===2).length+retweet5[1],
  st5.filter(a=>a.Sentiment===3).length+retweet5[2],
  st5.filter(a=>a.Sentiment===4).length+retweet5[3],
  st5.filter(a=>a.Sentiment===5).length+retweet5[4], ''],
];

let totneg=      (st1.filter(a=>a.Sentiment===1).length+retweet1[0])+
                  (st2.filter(a=>a.Sentiment===1).length+retweet2[0])+
                  (st3.filter(a=>a.Sentiment===1).length+retweet3[0])+
                  (st4.filter(a=>a.Sentiment===1).length+retweet4[0])+
                  (st5.filter(a=>a.Sentiment===1).length+retweet5[0]);
```



```

let tottendneg=    (st1.filter(a=>a.Sentiment===2).length+retweet1[1])+
                  (st2.filter(a=>a.Sentiment===2).length+retweet2[1])+
                  (st3.filter(a=>a.Sentiment===2).length+retweet3[1])+
                  (st4.filter(a=>a.Sentiment===2).length+retweet4[1])+
                  (st5.filter(a=>a.Sentiment===2).length+retweet5[1]);

let totneu=       (st1.filter(a=>a.Sentiment===3).length+retweet1[2])+
                  (st2.filter(a=>a.Sentiment===3).length+retweet2[2])+
                  (st3.filter(a=>a.Sentiment===3).length+retweet3[2])+
                  (st4.filter(a=>a.Sentiment===3).length+retweet4[2])+
                  (st5.filter(a=>a.Sentiment===3).length+retweet5[2]);

let tottendpos=   (st1.filter(a=>a.Sentiment===4).length+retweet1[3])+
                  (st2.filter(a=>a.Sentiment===4).length+retweet2[3])+
                  (st3.filter(a=>a.Sentiment===4).length+retweet3[3])+
                  (st4.filter(a=>a.Sentiment===4).length+retweet4[3])+
                  (st5.filter(a=>a.Sentiment===4).length+retweet5[3]);

let totpos=       (st1.filter(a=>a.Sentiment===5).length+retweet1[4])+
                  (st2.filter(a=>a.Sentiment===5).length+retweet2[4])+
                  (st3.filter(a=>a.Sentiment===5).length+retweet3[4])+
                  (st4.filter(a=>a.Sentiment===5).length+retweet4[4])+
                  (st5.filter(a=>a.Sentiment===5).length+retweet5[4]);

let stweek=[
  ["settimana", "NEGATIVI", "TEND.NEGATIVI", "NEUTRI", "TEND.POSITIVI", "POSITIVI", {role: 'annotation'}],
  ["WEEKS", totneg, tottendneg, totneu, tottendpos, totpos, ""]
]
return [stweeks, stweek]
},

```

La seconda funzione è `getSentimentAll()`. Tale funzione restituisce un array contenente il sentiment generale per ogni settimana e un array che contiene il numero totale di tweets raccolti ogni settimana. La funzione consta di 4 azioni fondamentali, ovvero:

1. Viene calcolato il numero di retweets per ogni settimana
2. Viene creato l'array per la visualizzazione dei dati suddivisi per ogni settimana
3. Viene creato l'array per la visualizzazione del numero totale di tweets di ogni settimana
4. Vengono restituiti i due array

```

//FUNZIONE CHE CREA UN ARRAY PER LA VISUALIZZAZIONE DEI DATI
//DI OGNI SETTIMANA, SUDDIVIDENDO OGNI BARRA PER I VARI SENTIMENT
//E RESTITUISCE ANCHE IL NUMERO TOTALE DI TWEET DI OGNI SETTIMANA
getSentimentAll: (state, getters) => {
  let retweet1 = getters.getRetweetMultiplier(state.Week1, 1)

```

```

let sumret1=retweet1.reduce((a,b)=> a+b,0);
let retweet2=getters.getRetweetMultiplier(state.Week2,2)
let sumret2=retweet2.reduce((a,b)=> a+b,0);
let retweet3=getters.getRetweetMultiplier(state.Week3,3)
let sumret3=retweet3.reduce((a,b)=> a+b,0);
let retweet4=getters.getRetweetMultiplier(state.Week4,4)
let sumret4=retweet4.reduce((a,b)=> a+b,0);
let retweet5=getters.getRetweetMultiplier(state.Week5,5)
let sumret5=retweet5.reduce((a,b)=> a+b,0);

let stweeks=[
  ["settimana","NEGATIVI","TEND.NEGATIVI","NEUTRI","TEND.POSITIVI","POSITIVI",{role:'annotation'}],
  ["WEEK1",state.Week1.filter(a=>a.Sentiment===1).length+retweet1[0],
    state.Week1.filter(a=>a.Sentiment===2).length+retweet1[1],
    state.Week1.filter(a=>a.Sentiment===3).length+retweet1[2],
    state.Week1.filter(a=>a.Sentiment===4).length+retweet1[3],
    state.Week1.filter(a=>a.Sentiment===5).length+retweet1[4],
    ''],
  ["WEEK2",state.Week2.filter(a=>a.Sentiment===1).length+retweet2[0],
    state.Week2.filter(a=>a.Sentiment===2).length+retweet2[1],
    state.Week2.filter(a=>a.Sentiment===3).length+retweet2[2],
    state.Week2.filter(a=>a.Sentiment===4).length+retweet2[3],
    state.Week2.filter(a=>a.Sentiment===5).length+retweet2[4],
    ''],
  ["WEEK3",state.Week3.filter(a=>a.Sentiment===1).length+retweet3[0],
    state.Week3.filter(a=>a.Sentiment===2).length+retweet3[1],
    state.Week3.filter(a=>a.Sentiment===3).length+retweet3[2],
    state.Week3.filter(a=>a.Sentiment===4).length+retweet3[3],
    state.Week3.filter(a=>a.Sentiment===5).length+retweet3[4],''],
  ["WEEK4",state.Week4.filter(a=>a.Sentiment===1).length+retweet4[0],
    state.Week4.filter(a=>a.Sentiment===2).length+retweet4[1],
    state.Week4.filter(a=>a.Sentiment===3).length+retweet4[2],
    state.Week4.filter(a=>a.Sentiment===4).length+retweet4[3],
    state.Week4.filter(a=>a.Sentiment===5).length+retweet4[4],''],
  ["WEEK5",state.Week5.filter(a=>a.Sentiment===1).length+retweet5[0],
    state.Week5.filter(a=>a.Sentiment===2).length+retweet5[1],
    state.Week5.filter(a=>a.Sentiment===3).length+retweet5[2],
    state.Week5.filter(a=>a.Sentiment===4).length+retweet5[3],
    state.Week5.filter(a=>a.Sentiment===5).length+retweet5[4],''],
]

let weeks=[
  ["WEEKS","TWEETS",{role:'annotation'}],
  ["Week1",state.Week1.length+sumret1,state.Week1.length+sumret1],
  ["Week2",state.Week2.length+sumret2,state.Week2.length+sumret2],
  ["Week3",state.Week3.length+sumret3,state.Week3.length+sumret3],
  ["Week4",state.Week4.length+sumret4,state.Week4.length+sumret4],
  ["Week5",state.Week5.length+sumret5,state.Week5.length+sumret5]

```

```

    ]
    return [stweeks,weeks]
  },

```

Funzioni per le singole settimane

Per ogni settimana abbiamo previsto di visualizzare un grafico generale, un grafico in base ad un hashtag scelto e un grafico in base alla data scelta; di conseguenza sono state implementate 3 funzioni.

La funzione GetSentimentWeek(), riceve in input una settimana ed estrapola un array per la visualizzazione del sentiment relativo.

Come primo passo controlla la settimana data; poi crea 5 variabili per i 5 sentiment che avranno la collezione filtrata; effettua il calcolo dei retweets ed infine restituisce l'array per la visualizzazione.

```

//FUNZIONE CHE PRENDE IN INPUT IL NUMERO DELLA SETTIMANA
//E RITORNA UN OGGETTO CHE CONTIENE LA VISUALIZZAZIONE DEI DATI SUDDIVISI PER SENTIMENT
//L'OUTPUT É UN GRAFICO A TORTA E UN GRAFICO A BARRE
getSentimentWeek: (state, getters) => Week => {
  let st;
  if(Week==1){
    st= state.Week1}
  if(Week==2){
    st= state.Week2}
  if(Week==3){
    st= state.Week3}
  if(Week==4){
    st= state.Week4}
  if(Week==5){
    st= state.Week5}

  let Negative= st.filter(a=> a.Sentiment===1)
  let TendNegative= st.filter(a=> a.Sentiment===2)
  let Neutre= st.filter(a=> a.Sentiment===3)
  let TendPositive= st.filter(a=> a.Sentiment===4)
  let Positive= st.filter(a=> a.Sentiment===5)
  let retweet=getters.getRetweetMultiplier(st,Week)

  return { 'PieChart': [
    ["Sentimento", "Tweet"],
    ["Negativo", Negative.length+retweet[0]],
    ["Tendente Negativo", TendNegative.length+retweet[1]],
    ["Neutro", Neutre.length+retweet[2]],
    ["Tendente Positivo", TendPositive.length+retweet[3]],
    ["Positivo", Positive.length+retweet[4]]
  ]
},

```

```

'BarChart':[
  ["Sentimento","DATI",{ role: 'style' }],
  ["Negativo",Negative.length+retweet[0], '#FF0000'],
  ["Tendente Negativo",TendNegative.length+retweet[1], '#FF4343'],
  ["Neutro",Neutre.length+retweet[2], '#FFD000'],
  ["Tendente Positivo",TendPositive.length+retweet[3], '#00FF00'],
  ["Positivo",Positive.length+retweet[4], '#43AF43']

]
}
},

```

Le altre due funzioni, ovvero `getSentimentByData()` e `getSentimentByHashtag()`, che filtrano rispettivamente per data e per hashtag una settimana ricevuta in input, sono molto simili, ed effettuano le seguenti operazioni:

1. Caricamento della settimana data in input
2. Filtro per data o per hashtag
3. Filtro dei vari sentiment
4. Creazione array da ritornare

Per la creazione dei vari grafici ci siamo basati su una libreria messa a disposizione da Google, ovvero GChart. Una volta importata, creare un grafico è estremamente semplice.

Esempio relativo al grafico a torta presente nelle varie settimane :

```

<GChart
  type="PieChart"
  :data="charData"
  :options="charOptions"/>

```

Dove `charData` rappresenta i dati ricevuti e `charOption` è un oggetto che contiene le varie opzioni del grafico (il colore, il titolo, la grandezza, etc.)

6 Raccolta e Analisi dei Dati

In queste fasi del progetto si è provveduto alla raccolta dei tweets e all'analisi degli stessi. Durante la fase di raccolta, abbiamo creato per ogni settimana un'apposita collezione. Ognuna di esse è stata popolata dai tweets di quel caratteristico intervallo temporale. Nello specifico la cattura è avvenuta sistematicamente ogni giovedì, partendo dal 19 novembre e terminando il 17 dicembre, avendo così a disposizione tweets dal 12 novembre al 16 dicembre.

Nella fase di analisi dei dati, avendo dunque a disposizione settimanalmente le varie collezioni, si è potuto fare un'attenta analisi costituita da diverse fasi:

- Verifica dell'effettiva cattura dei tweets
- Verifica della correttezza dei tweets
- Verifica la presenza di particolari eventi
- Pulizia manuale dei tweets

6.1 Eventi Significativi

Durante l'intera fase progettuale c'è stato un susseguirsi di eventi significativi [13] che hanno avuto un impatto in termini di incremento e decremento dei dati. Le date raggruppate per settimana sono le seguenti:

1^a Settimana :

- *12 novembre:*
 - Restrizioni per Veneto, Friuli-Venezia Giulia ed Emilia-Romagna.
 - Possibili forti restrizioni per la Campania.
- *13 novembre:*
 - Speranza firma ordinanza: Tante regioni passano dall'area gialla a quella arancione e rossa.
 - Giornata di proteste a Napoli.
- *15 novembre:*
 - Stanziati 400 milioni per l'acquisto dei vaccini anti-COVID.
- *16 novembre :*
 - Firmata ordinanza, Abruzzo in zona rossa da mercoledì.

2ª Settimana :

- 21 novembre:
 - Manifestazioni di protesta in tutta Italia
- 22 novembre:
 - Pressing da parte delle regioni per la riapertura
 - Possibile distribuzione vaccino entro metà dicembre
- 23 novembre:
 - Boccia fa un'affermazione grave: "Molti italiani non ci saranno il prossimo natale"
- 25 novembre:
 - Morte Maradona

3ª Settimana :

- 29 novembre:
 - Speranza afferma: "No all'obbligo vaccinale, faremo una campagna di persuasione"
 - Azzolina afferma: "Favorevole a una graduale riapertura delle scuole superiori"
 - Aumento dei dottori morti di covid
 - Annuncio dei primi vaccini dal 15 gennaio
 - Annuncio Cashback di stato
- 30 novembre:
 - Nuovo DPCM
 - Tensioni nella maggioranza su Mes e Recovery Fund
- 1 dicembre:
 - Si discute del DPCM di Natale
 - COVID-19, il piano esecutivo vaccinale è pronto
- 2 dicembre:
 - Il piano per distribuire il vaccino anti-COVID viene comunicato
 - Decreto-legge per contrastare il contagio durante le feste
 - Record dei positivi Covid-19

4ª Settimana :

- 3 dicembre:
 - Decreto Natale
 - Le pressioni delle Regioni sul governo, per la riapertura
 - Il disastro Lombardia
- 4 dicembre:
 - Entrata in vigore DPCM
- 5 dicembre:
 - Nuova Ordinanza del Governo
- 6 dicembre:
 - Lombardia mancanza vaccini

5ª Settimana :

- 10 dicembre:
 - Il governo pronto ad un passo indietro sugli spostamenti a Natale
 - Allarme terza ondata.
- 11 dicembre:
 - Spostamenti tra comuni a Natale
- 13 dicembre:
 - Italia quasi tutta gialla

6.2 Pulizia dei Dati

La fase di pulizia avviene dopo la cattura e prima del sentiment, essa serve quindi per evitare valori scorretti ottenuti in fase di sentiment. Ciò avviene manualmente, aprendo il Dbms MongoDB e andando a ripulire i vari tweets.

La pulizia si divide in diverse fasi:

- Rimozione da ciascun tweet di eventuali parole non utili per l'analisi del sentiment.

```
1  _id: ObjectId("5fb6411f34aaaf570a081d74")
2  Id : 1329210606313029600
3  Hashtag : "#congiuntifuoriregione "
4  Retweet : 0
5  Data : "2020-11-18 23:51:37 "
6  : "APPELLO A . CHIEDIAMO DI VEDERE UN AFFETTOSTABILE LoveIsNotTourism congiuntifuoricomune congiuntifuoriregione congiunti congiuntifuoriprovincia dir.
7  : "#AFFETTOSTABILE#LoveIsNotTourism#congiuntifuoricomune#congiuntifuoriregione#congiunti#congiuntifuoriprovincia#diritti#affettistabili#PartnerNonConv.
8  Sentiment : 0
9  Compound : 0
```

- Rimozione nei limiti delle nostre possibilità dei tweets non pertinenti al contesto.

```

1  _id: ObjectId("5fbf828655f2cf4dafad8345")
2  Id : 1331544115968925696
3  Hashtag : "#coronavirus "
4  Retweet : 2
5  Data : "2020-11-25 10:23:50 "
6  Testo : "LIVE - Il bollettino e i numeri odierni della provincia di Bolzano Coronavirus Covid_19
7  Hashtags : "#Bolzano#Coronavirus#Covid_19 "
8  Sentiment : 0
9  Compound : 0

```

- Correzione dialetti

```

_id: ObjectId("5fbf7c8c23669b7c5cdf213b")
Id: 1329395473065668611
Hashtag: "#vaccinocovid"
Retweet: 0
Data: "2020-11-19 12:05:54"
Testo: "Fateve er vaccino! "
Hashtags: "#vaccinoCovid"
Sentiment: 0
Compound: 0

```

Prima

```

_id: ObjectId("5fbf7c8c23669b7c5cdf213b")
Id: 1329395473065668611
Hashtag: "#vaccinocovid"
Retweet: 0
Data: "2020-11-19 12:05:54"
Testo: "Fatevi il vaccino! "
Hashtags: "#vaccinoCovid"
Sentiment: 0
Compound: 0

```

Dopo

- Correzione di errori ortografici

Prima

```

1  _id: ObjectId("5fbf7c039f33bc1eae9ab4c")
2  Id : 1331509868646305793
3  Hashtag : "#giuseppeconte "
4  Retweet : 2
5  Data : "2020-11-25 08:07:45 "
6  Testo : "Perche Giuseppe Conte e una persona preparata e seria, non come i suoi ospiti abituali...
7  Hashtags : "#GiuseppeConte "
8  Sentiment : 0
9  Compound : 0

```

Dopo

```

1  _id: ObjectId("5fbf7c039f33bc1eae9ab4c")
2  Id : 1331509868646305793
3  Hashtag : "#giuseppeconte "
4  Retweet : 2
5  Data : "2020-11-25 08:07:45 "
6  Testo : "Perchè Giuseppe Conte è una persona preparata e seria, non come i suoi ospiti abituali...
7  Hashtags : "#GiuseppeConte "
8  Sentiment : 0
9  Compound : 0

```


- Rimozioni di eventuali tweets duplicati

```
_id: ObjectId("5fdb7627810c0f9833668ff2")
Id: 1337417576234823680
Hashtag: "#dpcm"
Retweet: 0
Data: "2020-12-11 15:22:52"
Testo: " dpcm la cosa divertente è che noi ancora li stiamo a rispettare"
Hashtags: "#dpcm"
Sentiment: 0
Compound: 0
```

```
_id: ObjectId("5fe466ab413f861cb8f8251c")
Id: 1337417576234823680
Hashtag: "#dpcm"
Retweet: 0
Data: "2020-12-11 15:22:52"
Testo: " dpcm la cosa divertente è che noi ancora li stiamo a rispettare"
Hashtags: "#dpcm"
Sentiment: 0
Compound: 0
```

```
_id: ObjectId("5fe466b8413f861cb8f8251d")
Id: 1337417576234823680
Hashtag: "#dpcm"
Retweet: 0
Data: "2020-12-11 15:22:52"
Testo: " dpcm la cosa divertente è che noi ancora li stiamo a rispettare"
Hashtags: "#dpcm"
Sentiment: 0
Compound: 0
```

- Correzione delle abbreviazioni nei tweets

Prima

```
_id: ObjectId("5fb6a49dcc527c8cafd27d9f")
Id: 1329098888603717600
Hashtag: "#zonarossa"
Retweet: 0
Data: "2020-11-18 16:27:23"
Testo: "cmq dalle mie parti + che zonarossa è un liberitutti"
Hashtags: "#zonarossa#liberitutti#Covid_19"
Sentiment: 0
Compound: 0
```

Dopo

```
_id: ObjectId("5fb6a49dcc527c8cafd27d9f")
Id: 1329098888603717600
Hashtag: "#zonarossa"
Retweet: 0
Data: "2020-11-18 16:27:23"
Testo: "Comunque dalle mie parti più che zonarossa è un liberi tutti"
Hashtags: "#zonarossa#liberitutti#Covid_19"
Sentiment: 0
Compound: 0
```

7 Elaborazione dei dati

Per l'elaborazione dei dati, si è pensato di illustrare gli stessi in modo che si potesse cogliere l'andamento del Sentiment, rispetto alle varie decisioni governative.

Per farlo, abbiamo individuato hashtags che avessero aspetti in comune, raggruppandoli dunque per "settori di interesse".

Le categorie individuate sono le seguenti: Colori Regioni, Negazionisti, Sanità, Governo, Positivisti.

Poi abbiamo appurato che altri hashtags, che nonostante fossero singoli, hanno avuto una particolare rilevanza in termini di popolarità di utilizzo, rappresentanti anch'essi dei particolari settori. Tra i quali: Scuola, Periodo natalizio, Congiunti.

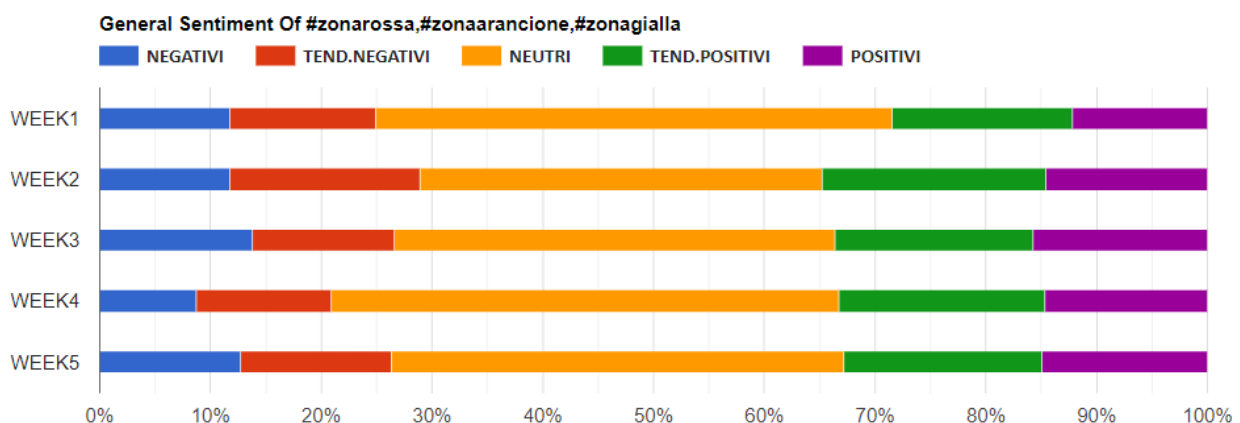
Abbiamo voluto ulteriormente fare un focus su quegli hashtags ritenuti essere da noi i più rappresentativi per poter rispondere al problema iniziale. La categoria, comprendente dunque gli hashtags scelti è COVID-19.

Abbiamo previsto il grafico *"General Sentiment of the Week"* che rappresenta il sentiment di tutti gli hashtags delle varie settimane e il grafico *"Total number of the Tweets for every Week Captured"*, indicante quanto ogni settimana sia stata più o meno discussa dagli eventi susseguirsi.

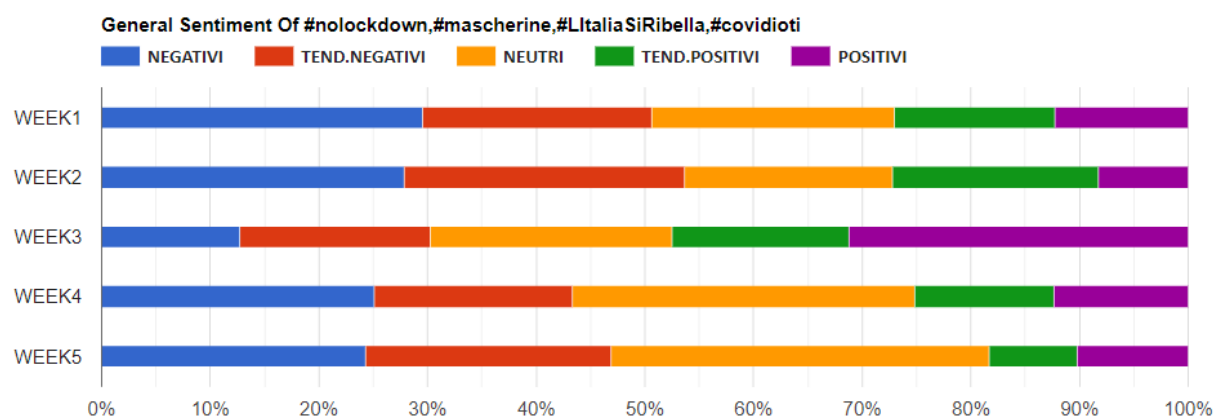
7.1 Visualizzazione Grafica dei Dati

La visualizzazione dei dati è stata ottenuta prendendo in considerazione le seguenti categorie. Per ognuna di esse, sono stati estrapolati i rispettivi hashtags:

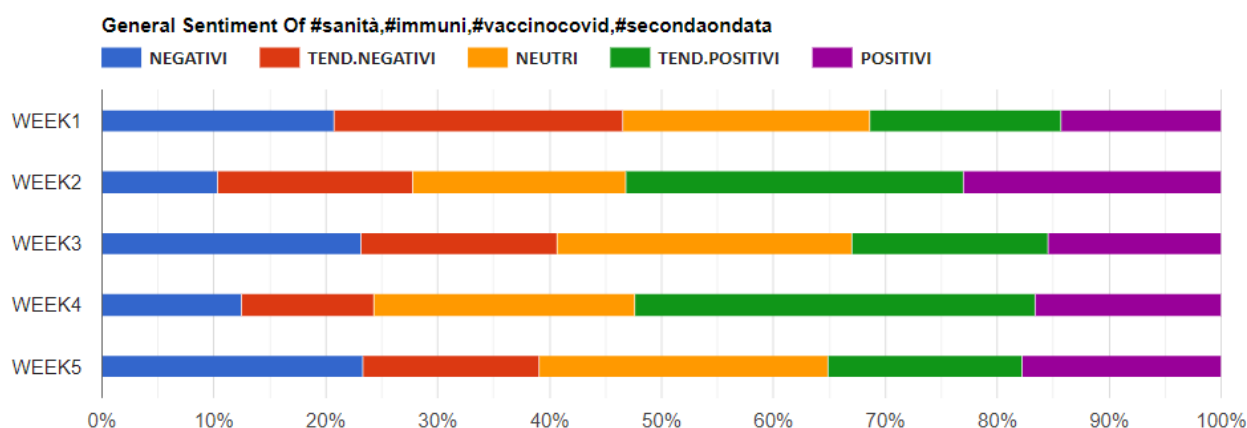
- *Colori Regioni*: Per tale categoria, gli hashtags che abbiamo individuato sono: #zonagiulla, #zonaarancione, #zonarossa.



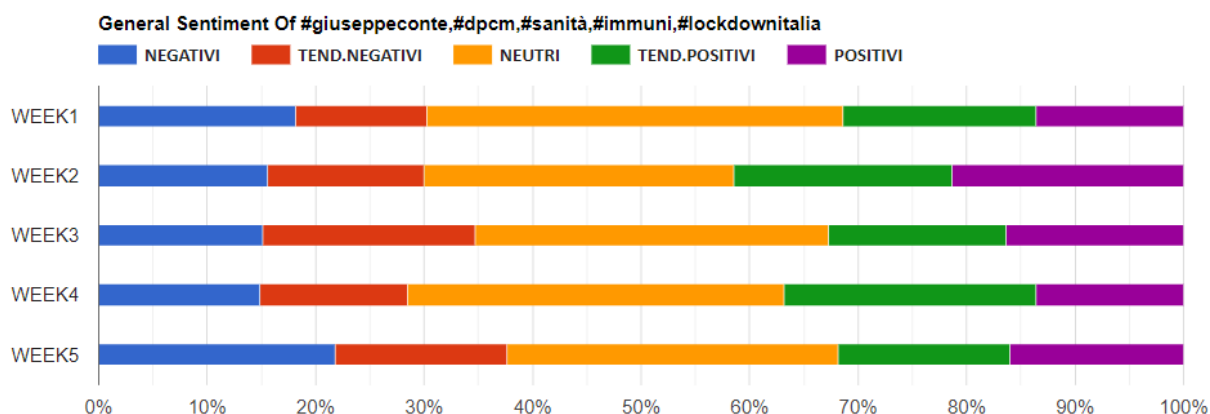
- *Negazionisti*: Per tale categoria, gli hashtags che abbiamo individuato sono: #nolockdown, #covidioti, #mascherine, #LItaliaSiRibella



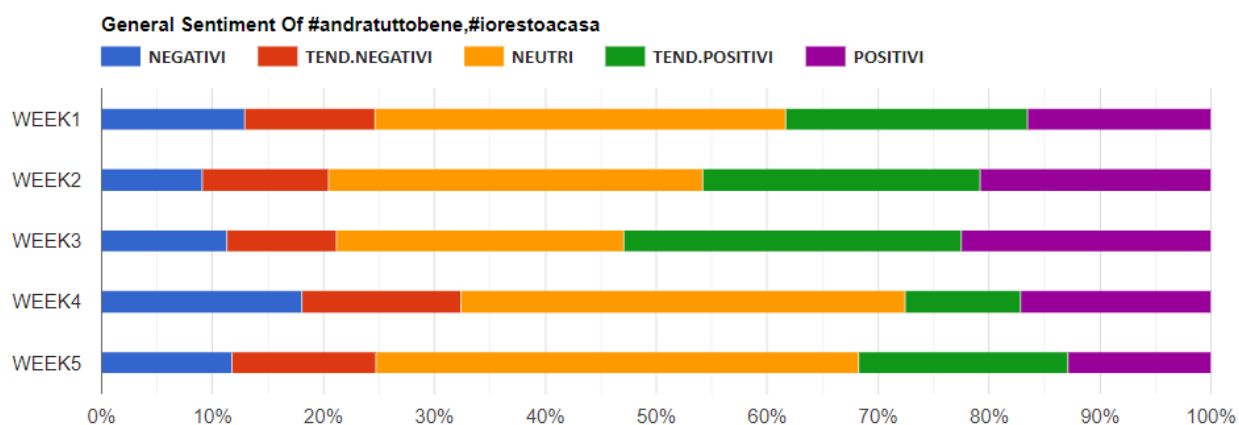
- *Sanità*: Per tale categoria, gli hashtags che abbiamo individuato sono: #vaccinocovid, #sanità, #immuni, #secondaondata



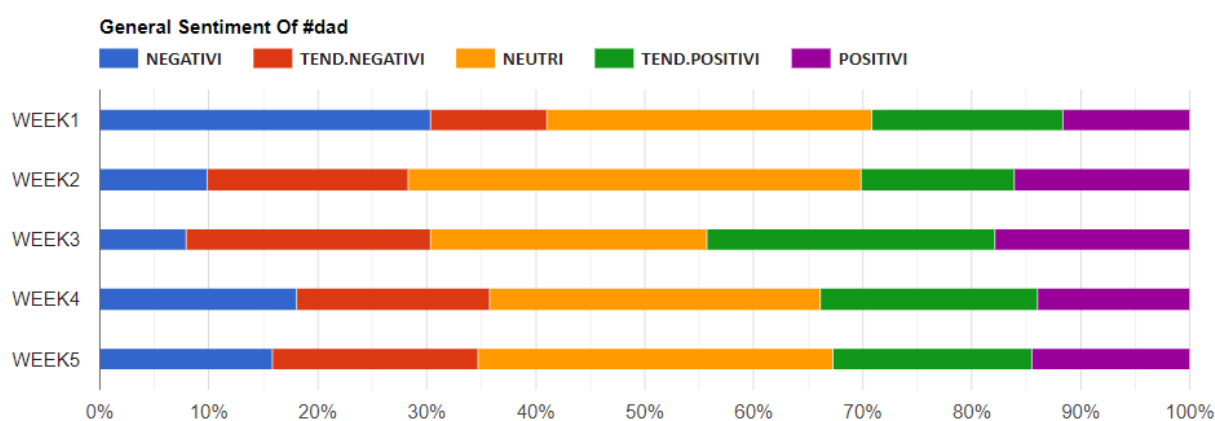
- *Governo*: Per tale categoria, gli hashtags che abbiamo individuato sono: #giuseppeconte, #dpcm, #sanità, #immuni, #lockdownitalia



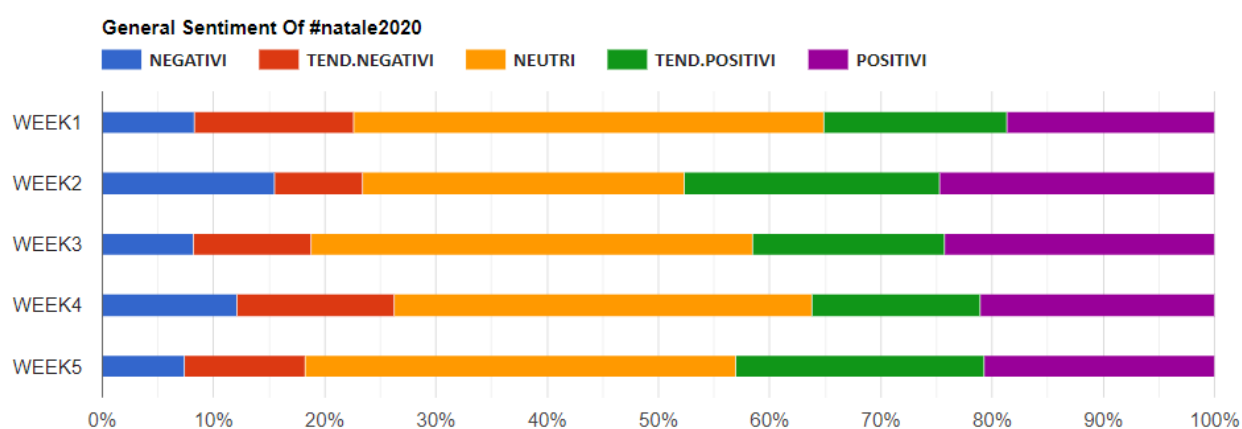
- *Positivisti*: Per tale categoria, gli hashtags che abbiamo individuato sono: **#andratuttobene** , **#iorestoacasa**



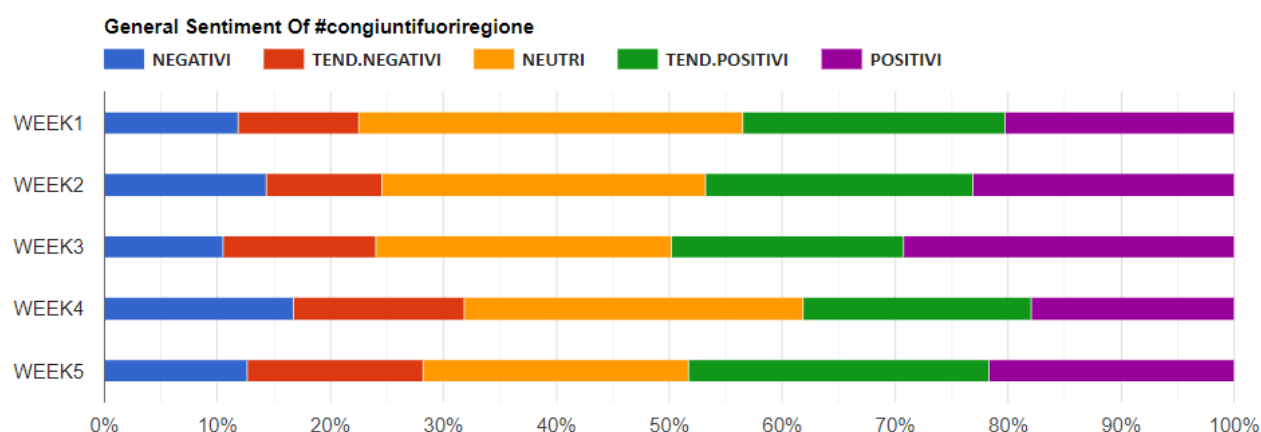
- *Scuola*: Per tale categoria, l'hashtag che abbiamo individuato è: **#dad**



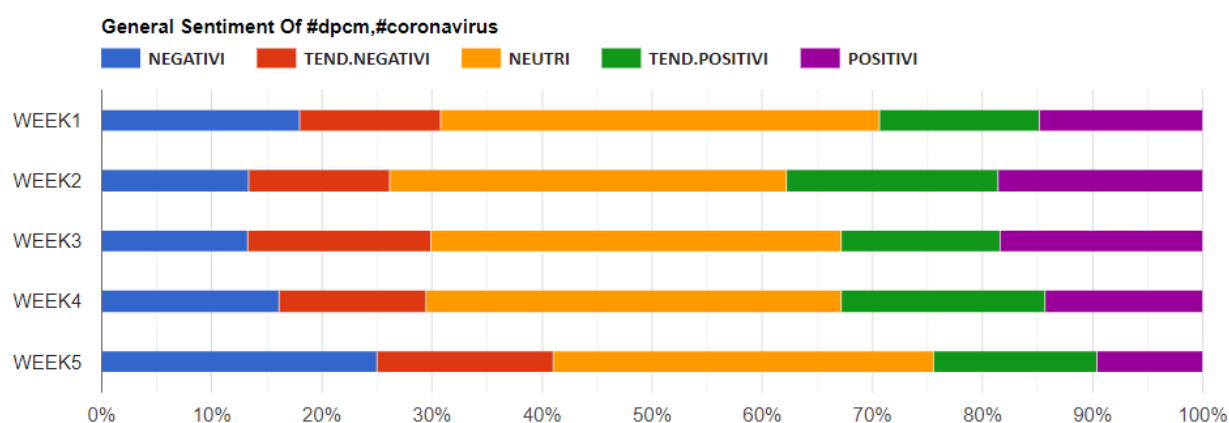
- *Periodo Natalizio*: Per tale categoria, l'hashtag che abbiamo individuato è : **#natale2020**



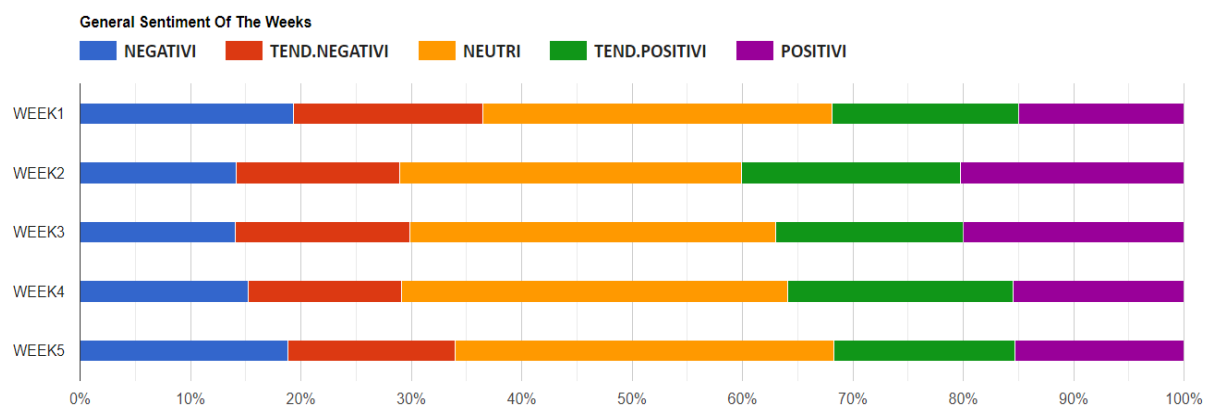
- *Congiunti*: Per tale categoria l'hashtag che abbiamo individuato è: **#congiuntifuoriregione**



- *Covid-19*: Per tale categoria gli hashtags che abbiamo individuato sono: **#dpcm, #coronavirus**

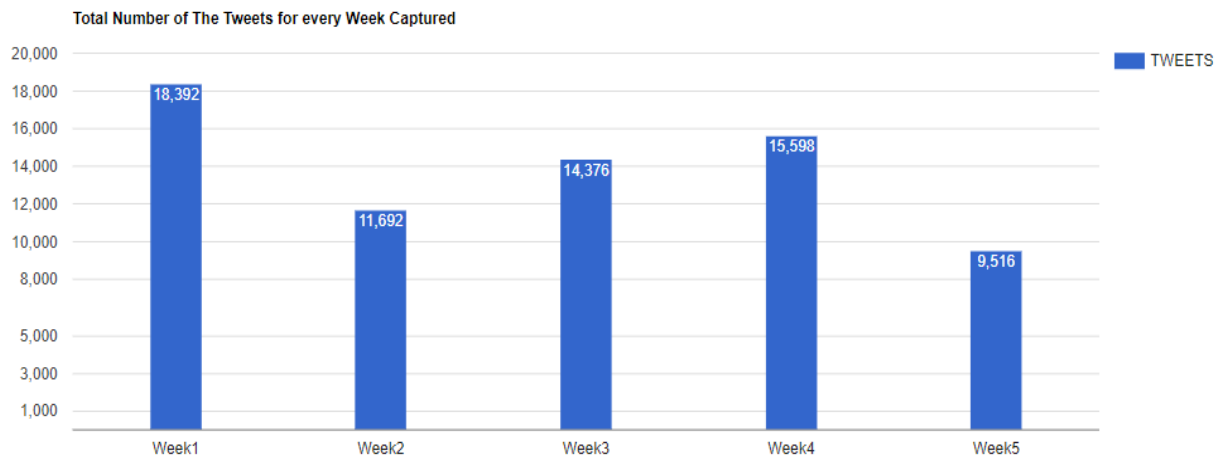


- **General Sentiment of the Weeks**



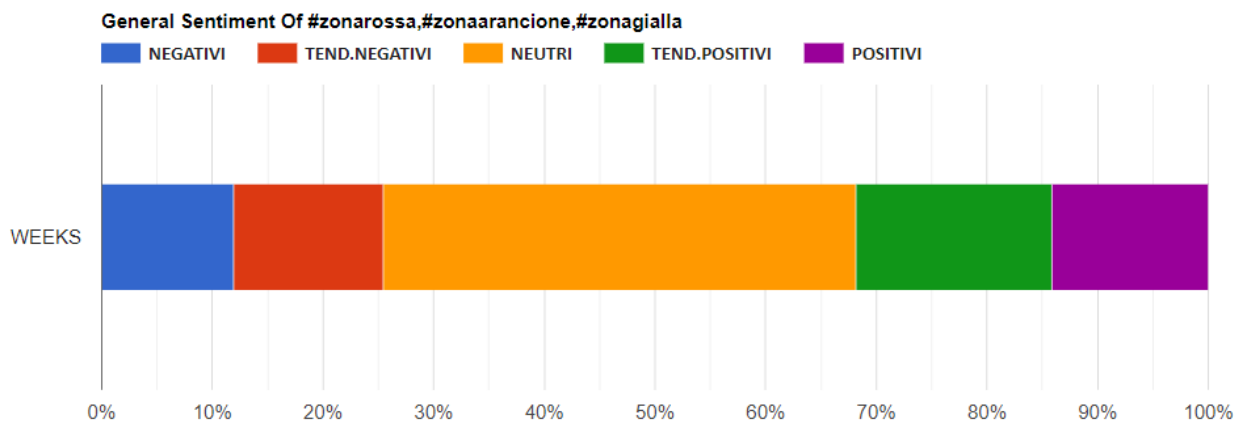
- **Total number of the Tweets for every Week Captured**

Analizziamo come cambia l'andamento in termini di quantità di tweets catturati nelle varie settimane:



7.2 Risposte al Problema

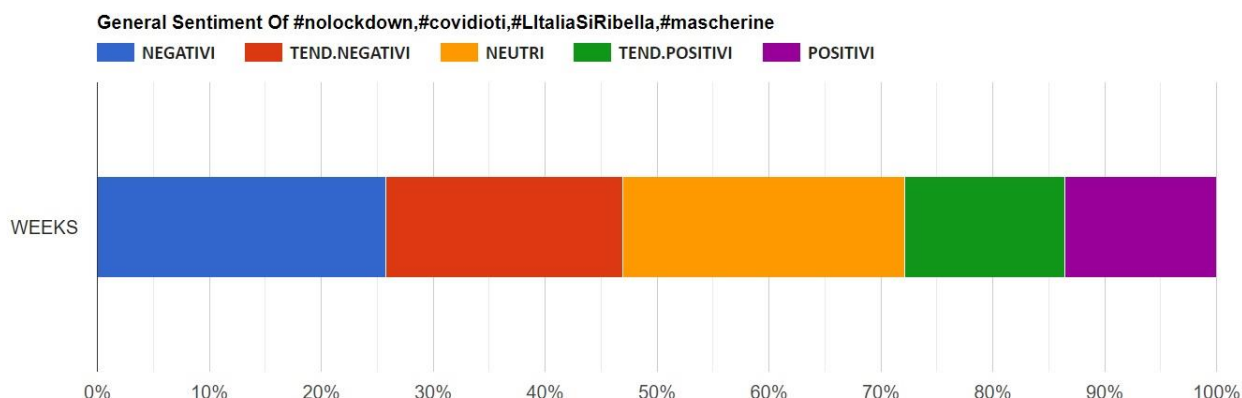
- Il primo grafico analizzato è *"Colori Regioni"*. Possiamo notare che in questa analisi il sentimento della popolazione rimane piuttosto costante nel tempo, segnalando una lieve positività nella seconda settimana, causata dalle manifestazioni di protesta avvenute in tutta Italia.



Per quanto riguarda le scelte governative rispetto alla situazione delle zone colorate, la popolazione ha avuto un sentiment fortemente neutro, con una maggiore tendenza alla positività verso le varie scelte prese.

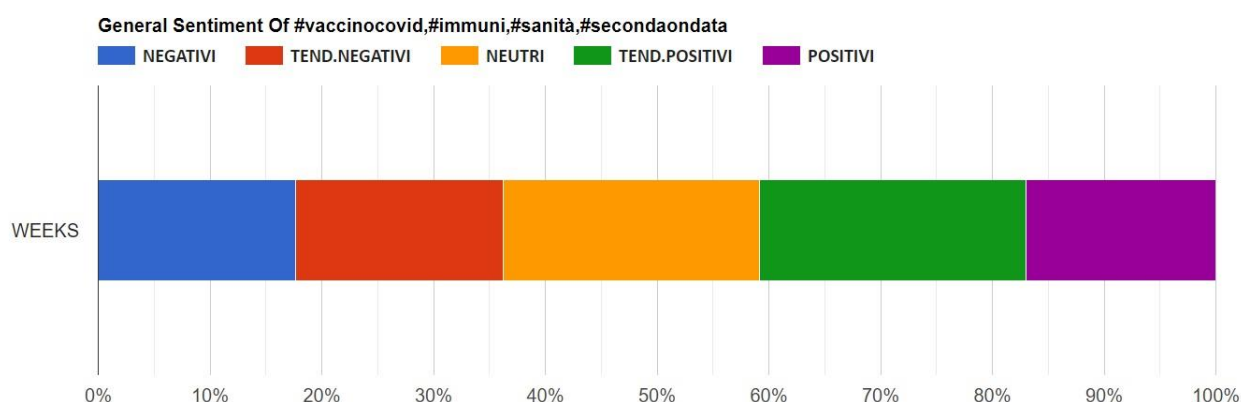
- Il secondo grafico analizzato è *"Negazionisti"*. Si può notare come il pensiero generale delle prime due settimane è di vicinanza ai principi negazionisti, mentre dalla terza settimana, complice una serie di eventi (Annuncio vaccino, record dei casi), si è avuto

un calo drastico della negatività. Nonostante tutto, il sentimento di negatività è continuato a crescere nelle restanti settimane, complice l'ultimo dpcm.



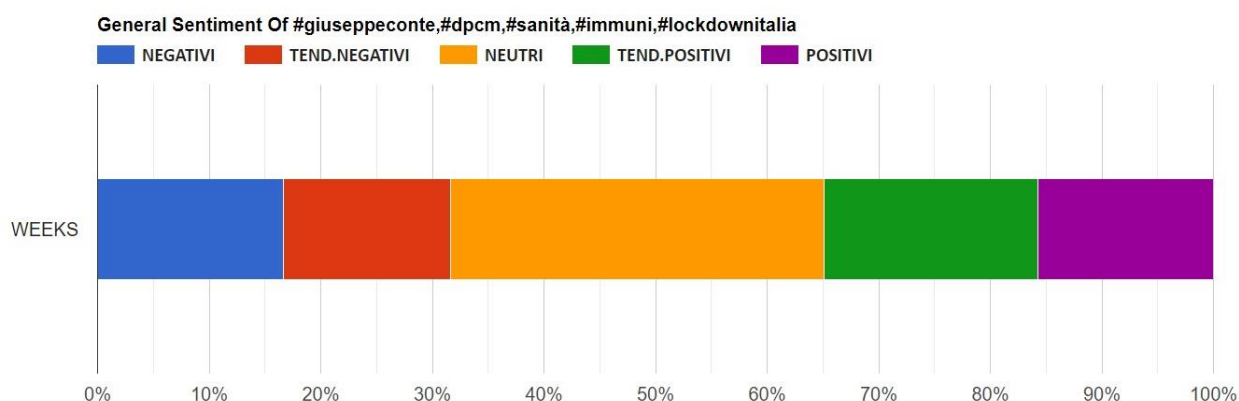
Si noti che per gli hashtags rappresentanti i negazionisti, la negatività ha prevalso fortemente, come era facilmente intuibile.

- Il terzo grafico analizzato è *"Sanità"*. In questo grafico risalta una forte percentuale di positività nella seconda e nella quarta settimana. Analizzando gli eventi significativi abbiamo constatato che ciò dipende da un evento molto importante per questa tipologia di hashtags, ovvero l'incremento nella seconda settimana è riconducibile al fatto che si è parlato della possibile distribuzione del vaccino in Italia. Per l'incremento della positività nella quarta settimana è da segnalare il fatto che il 2 dicembre è stato comunicato il piano di vaccinazione. Dal giorno seguente, che coincide con l'inizio della quarta settimana, si è quindi discusso dello stesso, facendo aumentare nuovamente il Sentiment positivo.



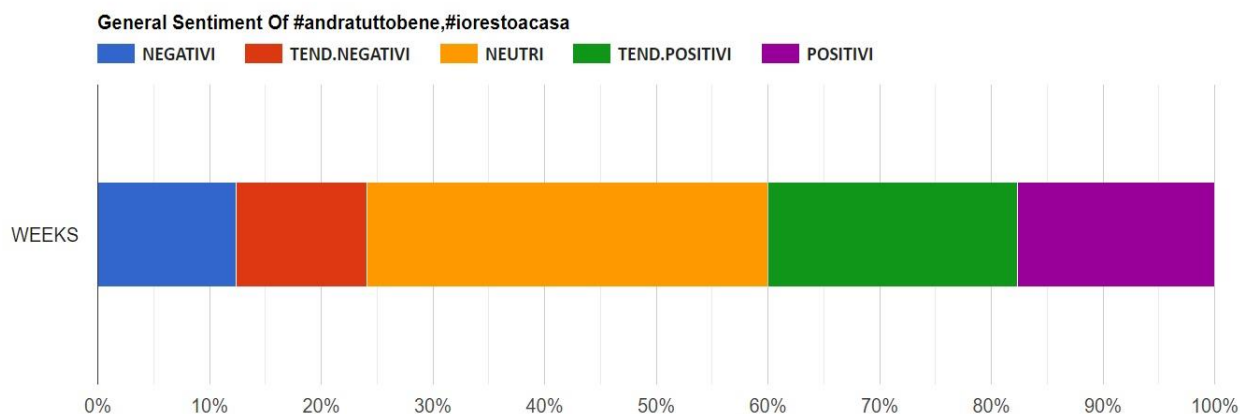
Per il settore sanitario, si presenta una minore neutralità rispetto alle altre categorie. Il Sentiment è rivolto leggermente verso la positività, presumibilmente dall'incidenza positiva di Week 2 e Week 4. Per avere una visione più precisa sulle motivazioni di questa positività, abbiamo analizzato questi hashtags escludendo sanità: il sentiment si è così bilanciato, aumentando in neutralità e calando in positività, confermando quindi l'incidenza del piano di vaccinazione.

- Il quarto grafico analizzato è *"Governo"*. L'andamento risulta essere abbastanza costante seppur con varie oscillazioni che anche in questo caso dipendono dalle notizie diffuse nella seconda e quarta settimana, riguardanti il vaccino.



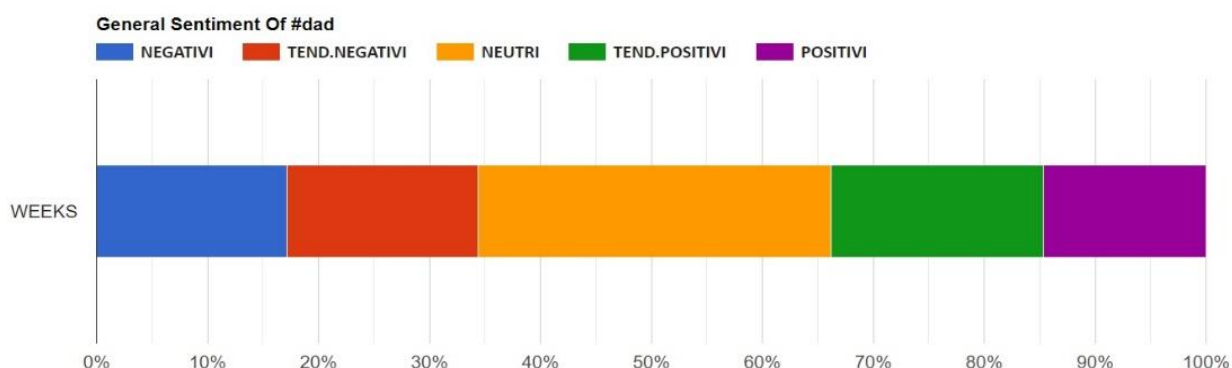
Questo grafico mostra un sentiment estremamente bilanciato anche se la positività è leggermente aumentata a causa degli avvenimenti susseguitisi in Week2 e Week4.

- Il quinto grafico analizzato è *"Positivisti"*. Essendo tipologie di hashtags aventi principi positivi, possiamo notare come sempre nella seconda settimana con il susseguirsi di notizie dell'uscita del vaccino Anti-covid, la popolazione ha avuto un sentiment fortemente positivo con un calo radicale del sentiment negativo, tuttavia studiando i dati abbiamo constatato la presenza di svariati tweets sarcastici/ironici.



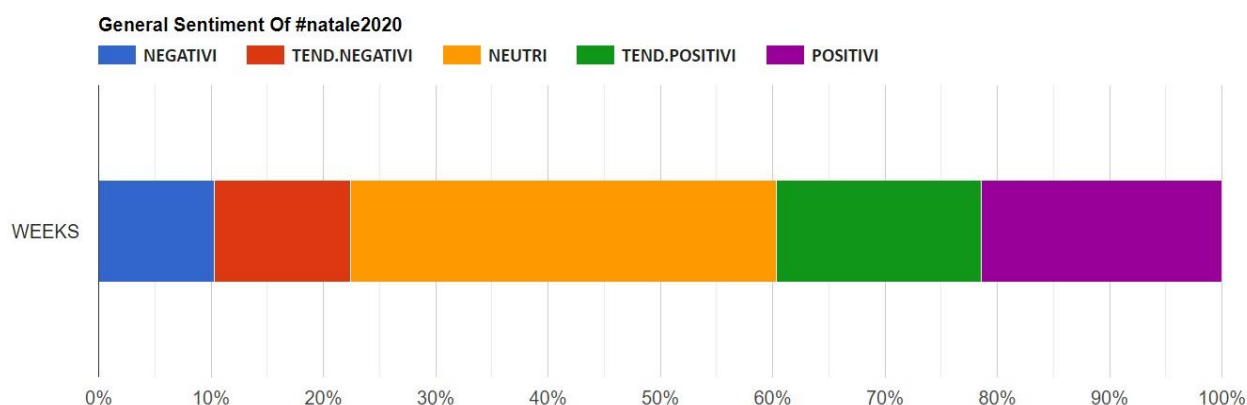
Da questo grafico possiamo notare come ci sia una tendenza alla neutralità, accompagnata da una maggiore positività, anche se questo sentiment è influenzato da altri fattori sarcastici.

- Il sesto grafico analizzato è “Scuola”. Il dato evidente in questo grafico riguarda la negatività nella prima settimana. Quest’ultima è giustificabile da accese discussioni, avute nella settimana antecedente all’inizio del progetto, in merito alle modalità della DAD e il malcontento di studenti e genitori, che avrebbero preferito il tipico percorso di studi.



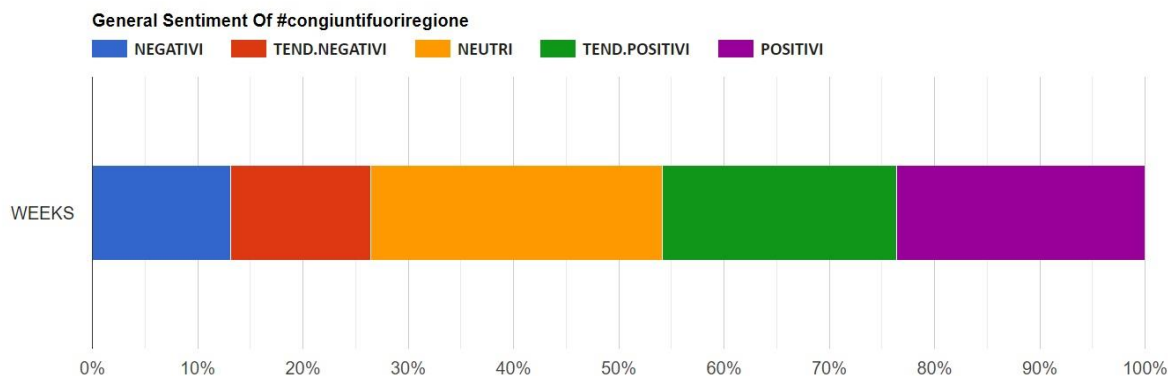
Tale categoria ha riscontrato un sentiment fortemente equilibrato, nonostante durante la prima settimana ci fosse una forte prevalenza di negatività. Questo ci fa capire come con il passare del tempo la popolazione ha pian piano iniziato a cambiare idea.

- Il settimo grafico è “Periodo Natalizio”. Dagli studi fatti sui dati è emerso che questa categoria è la più soggetta a sarcasmo e ironia, in maniera anche abbastanza rilevante, situazione che ha compromesso i risultati. In considerazione di ciò, neutralità a parte, il sentiment risulta essere prevalentemente positivo.



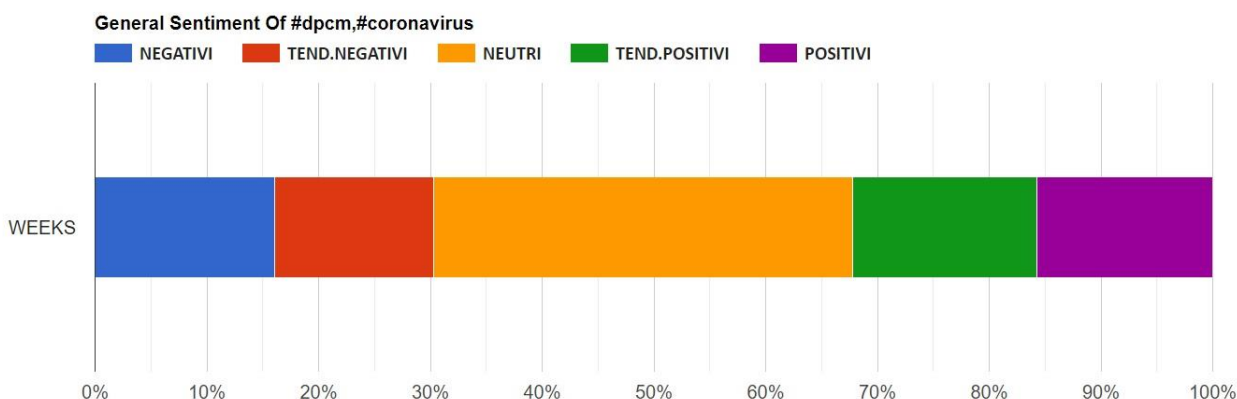
Le decisioni prese per il periodo natalizio hanno trovato un forte riscontro positivo nella popolazione a discapito della negatività, anche se come già detto sono presenti commenti ironici.

- L’ottavo grafico è “Congiunti”. Per questa categoria c’è da segnalare una problematica. All’interno del testo dei tweets si ricorre spesso a termini rilevati da Vader come positivi (affetto, amore, etc...) tuttavia, questi però appartengono a contesti negativi. Perciò il sentiment di questo hashtag è fortemente falsato.



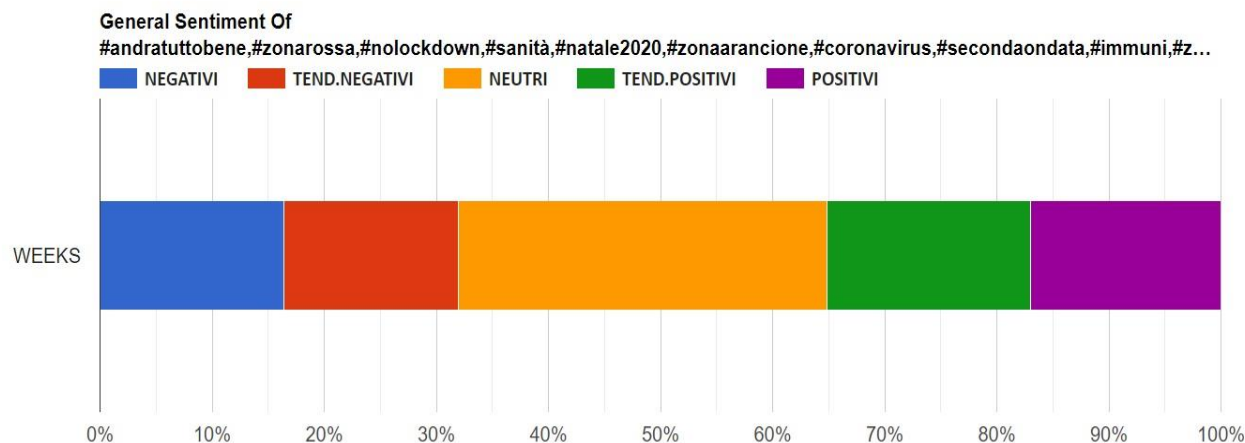
Data la problematica si è preferito non dare una risposta a questa categoria.

- Il nono grafico è "Covid-19". Esso rappresenta in modo più esplicativo il problema principale all'interno del progetto. Si può notare come tra le varie settimane ci siano cambi significativi riconducibili alle varie decisioni governative.



In generale quasi la maggioranza della popolazione ha un sentiment neutro e data anche in tal caso la presenza di ironia, possiamo dire che negativi e positivi si bilanciano.

- Il decimo grafico è "General Sentiment", descrive il sentimento generale di tutti gli hashtags analizzati per il progetto. È il grafico che fornisce una risposta ben precisa alla domanda "Come ha reagito la popolazione alle varie decisioni governative?"



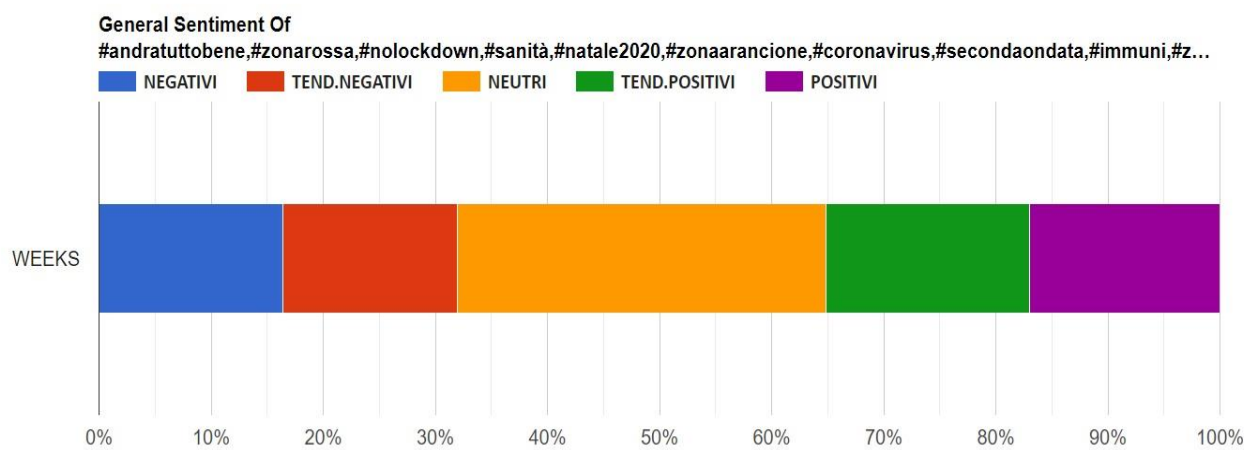
Analizzando dunque tutti gli hashtag e considerando diversi "settori" di interesse, la media del sentiment risulta essere costantemente neutra. Si può notare infatti che il Sentiment tende ad essere leggermente positivo, mantenendo comunque una forte prevalenza di neutralità.

8 Conclusioni

Nonostante le difficoltà già precedentemente descritte e l'esistenza dei fattori imponderabili, è stato comunque possibile raccogliere e analizzare i dati, ricavando dunque importanti risposte al problema. Per una visione dettagliata delle conclusioni è stato utilizzato il grafico "General Sentiment" :

Esso descrive il sentiment generale di tutti gli hashtags analizzati per il progetto. È il grafico che fornisce una risposta ben precisa alla domanda:

Come ha reagito la popolazione alle varie decisioni governative?



Analizzando tutti gli hashtags e considerando diversi "settori" di interesse, la media del sentiment risulta essere costantemente neutra. Si può notare infatti che il Sentiment tende ad essere leggermente positivo, mantenendo comunque una forte prevalenza di neutralità. In dettaglio i risultati ottenuti sono:

Negativi: 16.4%

Tendenti Negativi: 15.5%

Tendenti Positivi: 18.1%

Positivi: 17.1%

In conclusione, dalle percentuali ricavate è possibile evincere una **Positività pari al 35.2%** ed una **Negatività pari al 31.9%**. Possiamo dunque affermare che, durante il periodo studiato, la popolazione ha reagito in maniera abbastanza favorevole alle varie decisioni governative.

9 Riferimenti

- [1] Repository del Progetto <https://github.com/FrancescoPa96/ProgRetiUnisa.git>
- [2] API Twitter Developer <https://developer.twitter.com/en>
- [3] Azure Text Analytics <https://azure.microsoft.com/it-it/services/cognitive-services/text-analytics/#overview>
- [4] VPN Seed4.me <https://seed4.me/>
- [5] Sentiment Analysis. Wikipedia https://it.wikipedia.org/wiki/Analisi_del_sentiment
- [6] Tweepy <http://docs.tweepy.org/en/latest/>
- [7] MongoDB <https://www.mongodb.com/it/what-is-mongodb>
- [8] PyMongo <https://pymongo.readthedocs.io/en/stable/>
- [9] TextBlob <https://textblob.readthedocs.io/en/dev/>
- [10] Vader Sentiment Analysis <https://github.com/cjhutto/vaderSentiment>
- [11] Vue.js <https://vuejs.org/v2/guide/>
- [12] Google Charts <https://developers.google.com/chart>
- [13] Ultime notizie - Radio Popolare <https://www.radiopopolare.it/>



D² & R

**RETI GEOGRAFICHE:
STRUTTURA ANALISI E PRESTAZIONI**

Documentazione Progetto

A.A 2020/2021