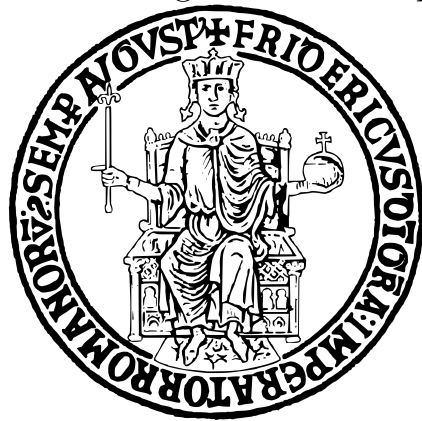


Big Data

Università degli Studi di Napoli



Debora Russo

Laura Sgammato

Luglio 2020

Indice

1	Tweet retrieval	6
1.1	Introduzione	6
1.2	Dataset	6
1.3	Tweet Retrieval	7
2	MongoDB	10
2.1	Struttura di MongoDB	10
2.1.1	Document	10
2.1.2	Collection	11
2.1.3	Aggregation	11
2.2	MongoDB Atlas	14
2.2.1	Connessione con Databricks	16
2.2.2	Confronto tra MongoDB Atlas e MongoDB in locale	17
3	Databricks	20
3.1	Preprocessing dei dati	20
3.2	VADER	22
3.3	Spark NLP	24
3.4	Evaluation	29
3.5	Rielaborazione dell'output	30

3.5.1	Collection 1 - Conteggio dei tweets nei giorni 28 febbraio, 30 marzo e aprile	30
3.5.2	Collection 1 - Sentiment dei tweets più retweetati	31
3.5.3	Collection 1 - Hashtags più utilizzati	32
3.5.4	Collection 2 - Sentiment medio positivo e negativo durante la settimana 20-26 aprile	33
3.5.5	Collection 2 - Sentiment medio per ogni Nazione	34
3.6	Confronto tra Pyspark e Pandas	34
4	Report in PowerBI	38
4.1	Power BI	38

Elenco delle figure

1.1	Fase di Sampling	8
1.2	Fase di Hydration	8
1.3	Memorizzazione dei tweets in un file JSONL	9
1.4	inserimento del file json in MongoCompass	9
2.1	Esempio di document	10
2.2	Inserimento in MongoDB da riga di comando	11
2.3	Operazione di merge	12
2.4	Operazione di project	12
2.5	Operazione di match	13
2.6	Operazione di out	13
2.7	Risultato della pipeline	14
2.8	Caricamento dataset in MongoAtlas tramite MongoShell	14
2.9	3 replica-set	15
2.10	election	16
2.11	Connettore per MongoDB	16
2.12	Connessione a MongoDB in Databricks e caricamento dei dati	17
2.13	Salvataggio in MongoDB	17
2.14	Confronto tempi di esecuzione query in MongoDB Compass e MongoAtlas	18
2.15	Query eseguita su 500.000 documents	19

3.1	Struttura di Apache Spark	20
3.2	Il testo prima del preprocessing	21
3.3	Replacing delle emoticon	22
3.4	Testo preprocessato	22
3.5	Funzione per calcolare il compound	23
3.6	Sentiment ottenuto	23
3.7	Labeling	24
3.8	Testi etichettati	24
3.9	Tokenizer	25
3.10	Normalizer	26
3.11	StopWordsCleaner	26
3.12	WordEmbeddings	26
3.13	Universal Sentence Encoder	27
3.14	Pipeline per il training	28
3.15	Epoche con relativa accuracy	28
3.16	Risultato della predizione	29
3.17	Pipeline addestrata	29
3.18	Etichette predette per il test set	29
3.19	Accuracy del modello	30
3.20	Codice per la somma del sentiment	31
3.21	Output del sentiment per i 3 giorni di Febbraio,Marzo e Aprile . .	31
3.22	Codice per ottenere i tweet più retweetati	32
3.23	Tweet in ordine di numero di retweet	32
3.24	Codice per estrarre gli Hashtags più utilizzati	32
3.25	Hashtags più utilizzati	33
3.26	Codice per ottenere i punteggi di sentiment assegnati ad ogni testo	33
3.27	Codice per ottenere i punteggi di sentiment assegnati ad ogni testo	34

3.28	Filtraggio righe il cui campo "place" sia nullo	34
3.29	Punteggio e Paese di provenienza di una serie di tweet	34
3.30	Codice Pyspark	35
3.31	Codice Python	36
3.32	Confronto Python e Pyspark	37
4.1	Tweet più retweetati e somma dei tweet in base alla sentiment . .	39
4.2	Hashtags più utilizzati e wordcloud	40
4.3	Sentiment medio positivo e negativo tra il 20 e il 26 aprile	41
4.4	Hashtags più utilizzati e wordcloud	42

Capitolo 1

Tweet retrieval

1.1 Introduzione

Attualmente esistono numerosi Social Networks che consentono alle persone di esprimere le proprie emozioni ed opinioni rispetto ad un determinato argomento come ad esempio Twitter. Obiettivo di questo progetto è analizzare le emozioni delle persone che tramite tweets si sono espresse riguardo la tematica del coronavirus. Specificamente parliamo di **Sentiment Analysis** che è la classificazione della polarità di ciascuna parola, frase o testo; in altre parole, si vuole determinare la positività, la negatività o la neutralità di quanto si sta analizzando.

1.2 Dataset

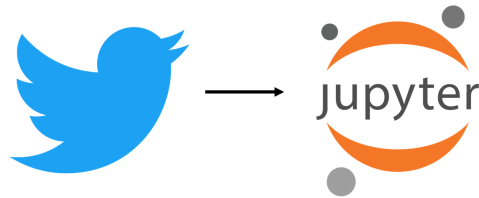
Il dataset è stato costruito prelevando i dati dal sito **CRISIS Nlp** che raccoglie i tweet dal primo febbraio al primo maggio. La nostra scelta è ricaduta sui tweets in lingua inglese di queste date:

- Selezione delle date del 30 Marzo, 28 Febbraio e 8 Aprile che sono le giornate in cui è stato registrato il più alto numero di tweets nei mesi di feb-

braio, Marzo e Aprile. In particolare sono stati prelevati 330.000 tweets per ciascun giorno con un totale di 990.000 tweets.

- selezione della settimana che va dal 20 al 26 aprile, ritenuta dopo un'accurata ricerca su internet quella di picco rispetto al numero dei contagiati in relazione ai tre mesi sopracitati. Da queste date sono stati prelevati 500.000 tweets.

1.3 Tweet Retrieval



I file scaricati dal sito sono in formato TSV e, per una questione di privacy, non contengono direttamente il testo dei tweet ma il tweet-id e l'user-id. Per ottenere i tweet reali abbiamo bisogno dell'identificatore del tweet e delle chiavi forniteci da Twitter dopo aver attivato un account developer (processo di **hydration**). Quindi, all'interno del notebook **Jupyter**, dopo una prima fase di Sampling che ci ha permesso di prelevare la sola colonna 'tweet-id' dal file TSV, abbiamo proceduto con una fase di Hydration facendo uso della libreria python **Twark**. Infine sono stati memorizzati i tweets reali in un file con estensione JSONL.


```
#importo il file .tsv scaricato da CrisisNlpCovid-19
```

```
import pandas as pd
import numpy as np
```

```
path = 'C:/Users/Laura/Desktop/'
```

```
df = pd.read_csv(path + '/tweet8_4.tsv', sep='\t')
df.head()
```

	tweet_id	user_id
0	1247867648278495232	839155711624036352
1	1247867648970457091	14621898
2	1247867666565496832	24395707
3	1247867657656979456	1217818147660562432
4	1247867668796973056	1233384897412247554

```
#seleziono solo la prima colonna del file tsv, cioè quella del tweet_id:
```

```
df = df['tweet_id']
df.to_csv(path + '/tweet_id_8Aprile.txt', header=False, index=False)
```

Figura 1.1: Fase di Sampling

```
#Hydration (ottengo i tweet reali a partire dai tweet_id con l'uso delle chiavi)
```

```
from twarc import Twarc
from itertools import islice
import time
```

```
API_key = 'ieiAWDozerwcT9kgBNfi3IFzN'
API_secret_key = 'bxfdr5HWrvxhZuDceOZY89sNa5vnoTTSVdx5Moeo4uvEPK3Bbj'
access_token = '1275590517351424003-QhDxgbBjhvowQVag04Kv9ILuu02a3G'
access_token_secret = 'fxYXGM730FSrn1N1zxQXcMt1sGGxI7rSDbWC70KEoJtK0'
```

```
twarc = Twarc(API_key, API_secret_key, access_token, access_token_secret)
```

```
tweets = []
i=0
for tweet in twarc.hydrate(open(path + '/tweet_id_8Aprile.txt')):
    tweets.append(tweet)
    i += 1
    if i% 50000 == 0 :
        print ("Tweet importati: ",i)
        time.sleep (90)
    if i == 333000 :
        break
```

Figura 1.2: Fase di Hydration

```
#scrivo i Tweet reali in un file jsonl

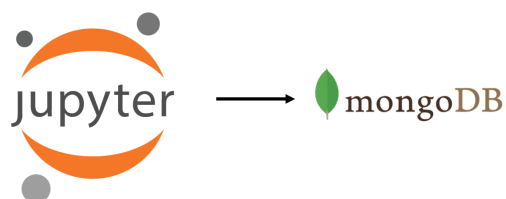
import json

f=open('tweets_8Aprile.jsonl','w')

for tweet in tweets:
    f.write(json.dumps(tweet)+'\n')

f.close()
```

Figura 1.3: Memorizzazione dei tweets in un file JSONL



Al fine di poter lavorare sui dati, ciascun file JSONL è stato importato in **MongoDB Compass** (database document-oriented) come Collection del database tramite la Mongo Shell:

```
C:\Program Files\MongoDB\Server\4.2\bin>mongoimport --db covid_bdabi --collection c1 --file C:\Users\Laura\Desktop\BigData\tweets.jsonl
2020-06-27T18:23:42.573+0200    connected to: mongodb://localhost/
2020-06-27T18:23:42.819+0200    10 document(s) imported successfully. 0 document(s) failed to import.
```

Figura 1.4: inserimento del file json in MongoCompass

Capitolo 2

MongoDB

2.1 Struttura di MongoDB

Per quanto riguarda lo storage dei dati, è stato utilizzato il database MongoDB: esso è di tipo NoSQL e orientato al documento.

2.1.1 Document

Un record in MongoDB è un documento, ovvero una struttura composta da coppie campo-valore, di tipo BSON (Binary Json):

```
_id: ObjectId("5f03d245baa72b951d561afd")
contributors: null
coordinates: null
created_at: "Mon Apr 20 13:01:24 +0000 2020"
> display_text_range: Array
> entities: Object
  favorite_count: 1
  favorited: false
  full_text: "Working in a hospital seeing first hand what COVID19 is doing, having ..."
  geo: null
  id: 1252220824414797831
  id_str: "1252220824414797831"
  in_reply_to_screen_name: null
  in_reply_to_status_id: null
  in_reply_to_status_id_str: null
  in_reply_to_user_id: null
  in_reply_to_user_id_str: null
  is_quote_status: false
  lang: "en"
  place: null
  retweet_count: 0
  retweeted: false
  source: "<a href='\"http://twitter.com/download/iphone\"' rel='\"nofollow\">Twitter fo..."
  truncated: false
> user: Object
```

Figura 2.1: Esempio di document

Ad ogni documento viene associato un id atto ad identificarlo.

2.1.2 Collection

Tutti i documenti vengono salvati all'interno di collections. Una volta quindi ottenuti tutti i file di tipo jsonl nella fase di hydration dei tweets si è passati, tramite riga di comando, ad inserirli all'interno di collections create all'interno del database:

```
C:\Program Files\MongoDB\Server\4.2\bin>mongoimport --db covidGeo --collection 22_04_20 --file C:\Users\Debora\22_04_20.jsonl
2020-07-07T03:44:41.458+0200 connected to: mongodb://localhost/
2020-07-07T03:44:44.458+0200 [.....] covidGeo.22_04_20 5.86MB/417MB (1.4%)
2020-07-07T03:44:47.469+0200 [.....] covidGeo.22_04_20 16.8MB/417MB (4.0%)
2020-07-07T03:44:50.458+0200 [#####] covidGeo.22_04_20 29.0MB/417MB (6.9%)
2020-07-07T03:44:53.465+0200 [#####] covidGeo.22_04_20 49.9MB/417MB (12.0%)
2020-07-07T03:44:56.459+0200 [#####] covidGeo.22_04_20 62.5MB/417MB (15.0%)
2020-07-07T03:44:59.458+0200 [#####] covidGeo.22_04_20 85.9MB/417MB (20.6%)
2020-07-07T03:45:02.460+0200 [#####] covidGeo.22_04_20 102MB/417MB (24.5%)
2020-07-07T03:45:05.458+0200 [#####] covidGeo.22_04_20 115MB/417MB (27.5%)
2020-07-07T03:45:08.458+0200 [#####] covidGeo.22_04_20 131MB/417MB (31.4%)
2020-07-07T03:45:11.458+0200 [#####] covidGeo.22_04_20 144MB/417MB (34.5%)
2020-07-07T03:45:14.458+0200 [#####] covidGeo.22_04_20 150MB/417MB (35.9%)
2020-07-07T03:45:17.460+0200 [#####] covidGeo.22_04_20 150MB/417MB (35.9%)
2020-07-07T03:45:20.458+0200 [#####] covidGeo.22_04_20 151MB/417MB (36.1%)
```

Figura 2.2: Inserimento in MongoDB da riga di comando

2.1.3 Aggregation

Successivamente, tramite l'**aggregation framework**, sono state effettuate diverse query (costruendo una pipeline) per ottenere due datasets su cui definire l'algoritmo di sentiment analysis. Dapprima è stato eseguito il **merging delle collezioni**, ottenendone due:

- La settimana comprendente i giorni che vanno dal 20 al 26 aprile, contenente circa 500000 documenti (tweets)
- I giorni 28 febbraio, 30 marzo e 8 aprile, contenente 1 milione di documenti

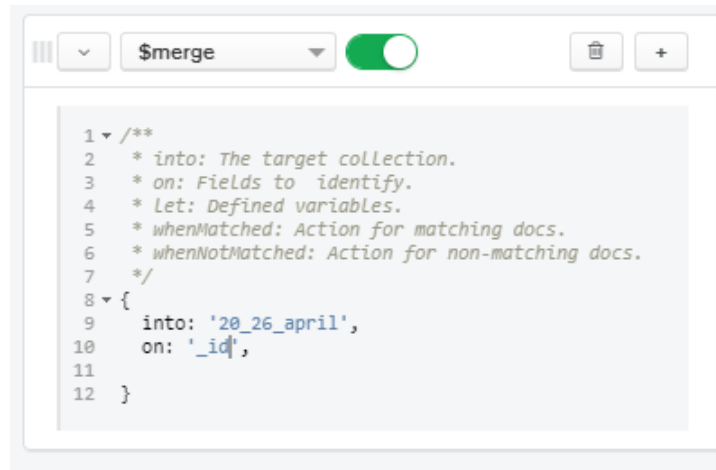


Figura 2.3: Operazione di merge

In seguito è stata realizzata un'operazione di **“project”**, in modo tale da selezionare i campi utili alla nostra analisi:

- Created-at: la data di creazione del tweet
- Entities: campo contenente gli hashtags scritti all'interno del tweet
- Full-text: il testo del tweet
- Place: se presente, la nazione da cui è stato inviato il tweet
- Retweet-count: il numero di volte in cui è stato retweetato il tweet

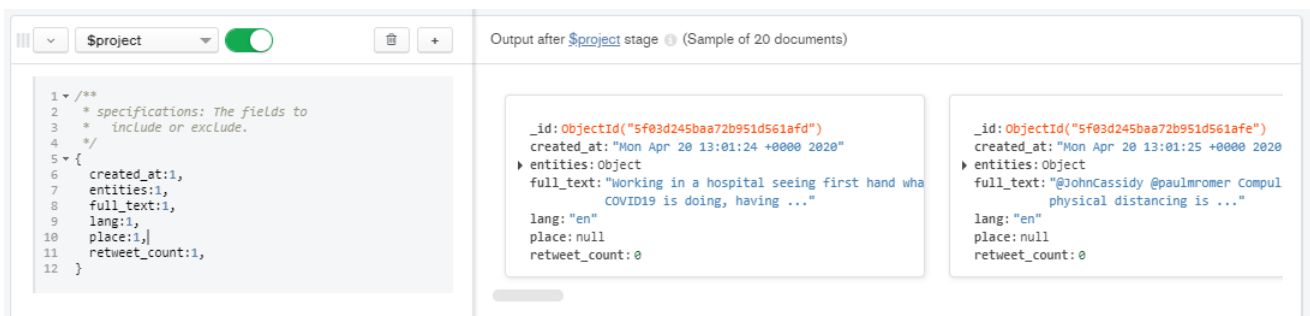


Figura 2.4: Operazione di project

Dopodichè, con l'operazione “**match**” sono stati selezionati i tweet in lingua inglese:

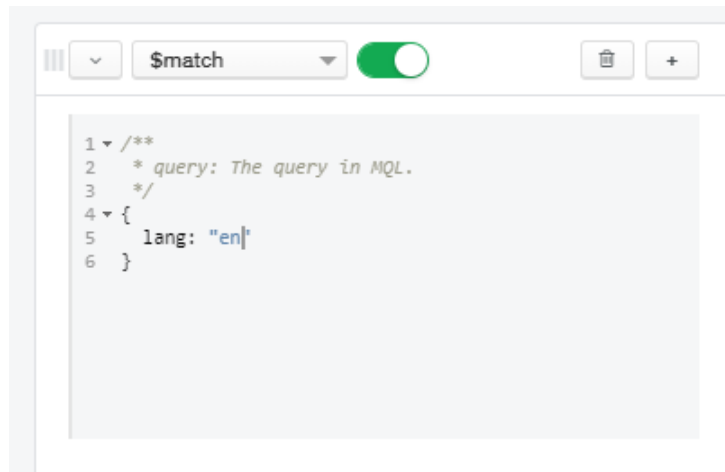


Figura 2.5: Operazione di match

Infine, tramite l'operazione “out”, è stata creata una nuova collection:

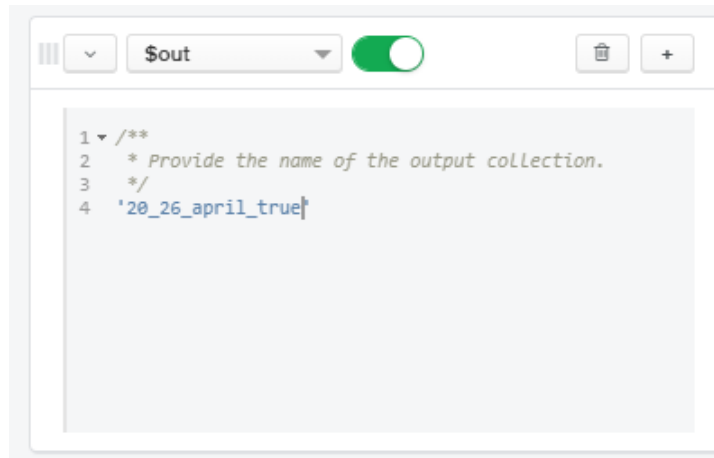


Figura 2.6: Operazione di out

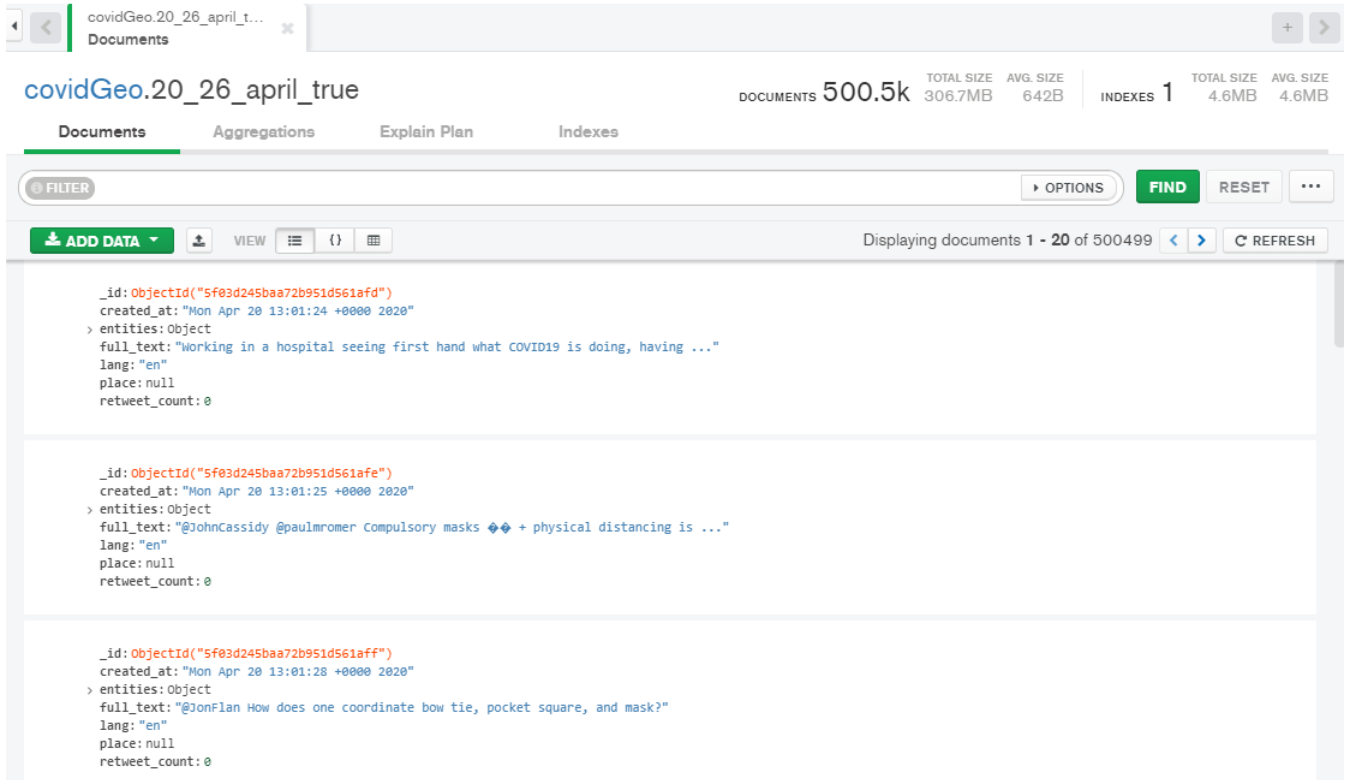


Figura 2.7: Risultato della pipeline

2.2 MongoDB Atlas

```
C:\Program Files\MongoDB\Server\4.2\bin>mongoimport --host cluster-mesi-shard-00-02.onsz5.gcp.mongodb.net:27017 --db db_mesi --collection tweets_mesi --type json --file C:\Users\Laura\Desktop\dataset\mesi_ok.json --jsonArray --authenticationDatabase admin --ssl --username laura --password laura902
connected to: mongodb://cluster-mesi-shard-00-02.onsz5.gcp.mongodb.net:27017/
[.....] db.mesi.tweets_mesi 1.04MB/511MB (0.2%)
[.....] db.mesi.tweets_mesi 1.04MB/511MB (0.2%)
[.....] db.mesi.tweets_mesi 2.07MB/511MB (0.4%)
[.....] db.mesi.tweets_mesi 2.07MB/511MB (0.4%)
[.....] db.mesi.tweets_mesi 3.05MB/511MB (0.6%)
[.....] db.mesi.tweets_mesi 3.05MB/511MB (0.6%)
[.....] db.mesi.tweets_mesi 4.03MB/511MB (0.8%)
[.....] db.mesi.tweets_mesi 4.03MB/511MB (0.8%)
[.....] db.mesi.tweets_mesi 5.02MB/511MB (1.0%)
[.....] db.mesi.tweets_mesi 5.02MB/511MB (1.0%)
```

Figura 2.8: Caricamento dataset in MongoAtlas tramite MongoShell

I file Json generati sono stato inseriti nel database cloud MongoDB Atlas: è stata scelta la versione starter costituita da uno storage di 512 mb, motivo per il quale

i file jsonl ottenuti dall'hydration sono stati trasformati in json in un database MongoDB in locale e poi esportati, e di un “3 replica-set”, costituito da:

- Un membro “primary” che riceve le operazioni di scrittura
- Due membri secondari che applicano le operazioni ai loro datasets in modo tale da riflettere il dataset del primario

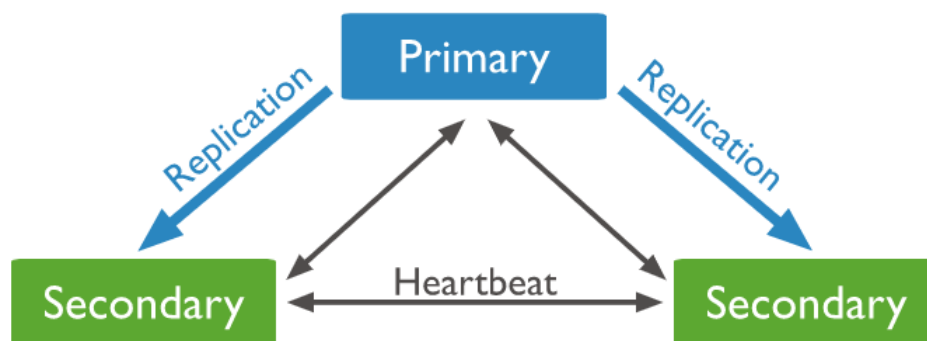


Figura 2.9: 3 replica-set

Questo tipo di configurazione fornisce due copie complete del dataset in ogni momento oltre a quello primario. I replica sets offrono ulteriore tolleranza agli errori e alta disponibilità. Infatti, se il primario non è disponibile, permettono di eleggere un secondario come primario per continuare l'operazione. Infine il vecchio primario si ricongiunge all'insieme una volta tornato disponibile.

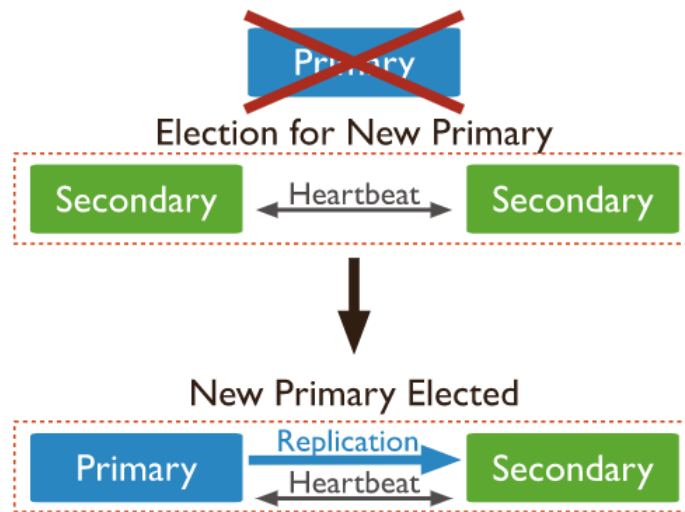


Figura 2.10: election

2.2.1 Connessione con Databricks

Per connettere MongoDB Atlas a Databricks è stato installato sul cluster il connettore ufficiale:

● sentiment | [Edit](#) [Clone](#) [Restart](#) [Terminate](#) [Delete](#)

[Configuration](#) [Notebooks \(0\)](#) [Libraries](#) [Event Log](#) [Spark UI](#) [Driver Logs](#) [Metrics](#) [Apps](#) [Spark Cluster UI - Master](#)

[Uninstall](#) [Install New](#)

<input type="checkbox"/>	Name	Type	Status	Source
<input type="checkbox"/>	com.johnsnowlabs.nlp.spark-nlp_2.11:2.5.0	Maven	● Installed	
<input type="checkbox"/>	org.mongodb.spark:mongo-spark-connector_2....	Maven	● Installed	
<input type="checkbox"/>	spark-nlp	PyPI	● Installed	

Figura 2.11: Connettore per MongoDB

All'interno del notebook è possibile leggere le collections inserite nel database e assegnarle ad un dataframe, in modo tale da effettuare le operazioni necessarie alla sentiment analysis in linguaggio pyspark.

```

1 #1-START_conessione con mongoAtlas Avvio PySpark con connettore Spark MongoDB
2
3 import pyspark
4 from pyspark.sql import SparkSession
5
6 my_spark = SparkSession \
7     .builder \
8     .appName("myApp") \
9     .config("spark.mongodb.input.uri", "spark.mongodb.input.uri mongodb+srv://Debora:ciao@cluster0.dwkae.gcp.mongodb.net/covid19geo?
    retryWrites=true&w=majority") \
10    .config("spark.mongodb.output.uri", "mongodb+srv://Debora:ciao@cluster0.dwkae.gcp.mongodb.net/covid19geo?retryWrites=true&w=majority") \
11    .getOrCreate()
12
13 #CONNECT ATLAS WITH APPLICATION: LEGGO LA COLLECTION tweets_mes1
14
15 hashtag =
16 spark.read.format("com.mongodb.spark.sql.DefaultSource").option("uri","mongodb+srv://Debora:ciao@cluster0.dwkae.gcp.mongodb.net/covid19geo.20_04_20?
    retryWrites=true&w=majority").load()
17 hashtag.show(60)

```

Figura 2.12: Connessione a MongoDB in Databricks e caricamento dei dati

Una volta effettuate le opportune rielaborazioni sui dati, essi vengono salvati in MongoDB:

```

Cmd 5
1 mes13 =
df_final_sum.write.format("com.mongodb.spark.sql.DefaultSource").mode("append").option("database","covid19geo").option("collection","mesi_sum1m1n").save()

```

Figura 2.13: Salvataggio in MongoDB

2.2.2 Confronto tra MongoDB Atlas e MongoDB in locale

Possiamo notare come, all'aumentare del numero di tweets, il tempo di esecuzione di una query aumenti e in particolare, l'esecuzione di tale query in mongoDB Compass sia molto più lenta rispetto all'esecuzione in MongoDB Atlas:

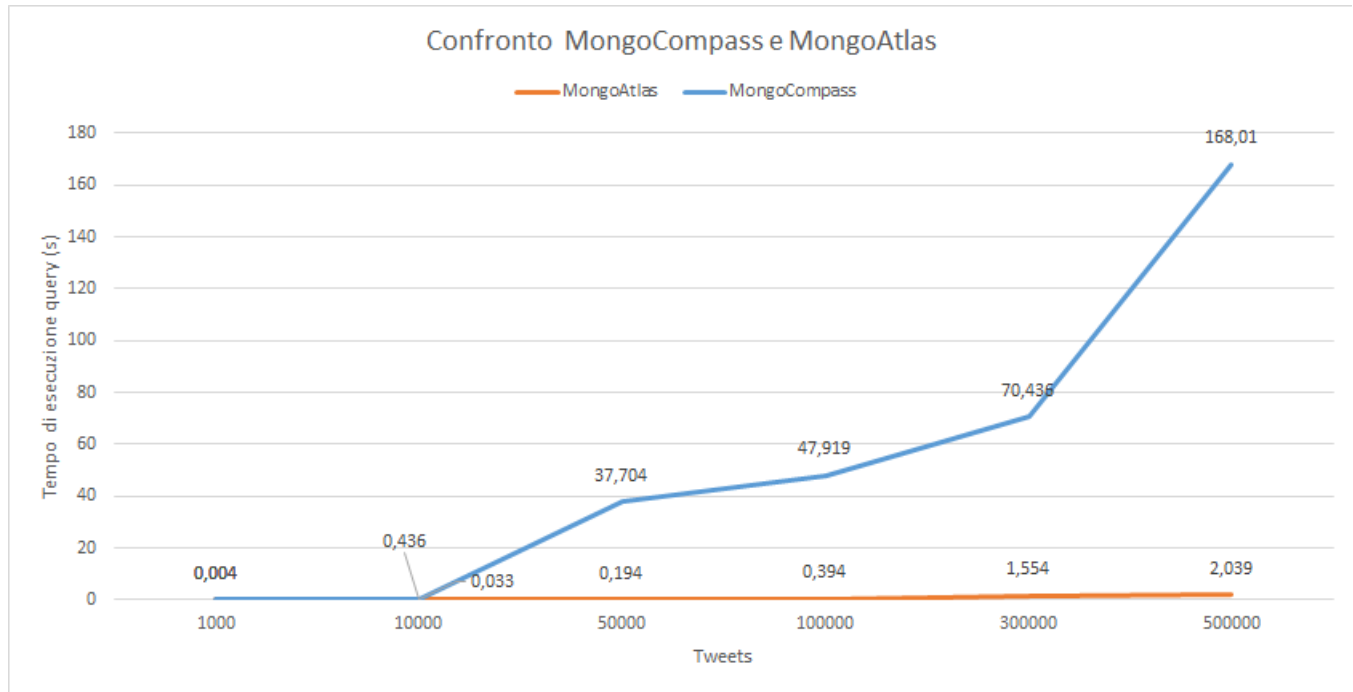


Figura 2.14: Confronto tempi di esecuzione query in MongoDB Compass e MongoAtlas

La query in questione restituisce il numero di retweet per ciascun tweet scritto in lingua inglese presente nella collection e l'analisi dei tempi di esecuzione è stato fatto su tale numero di documenti:

- 1.000
- 10.000
- 50.000
- 100.000
- 300.000
- 500.000

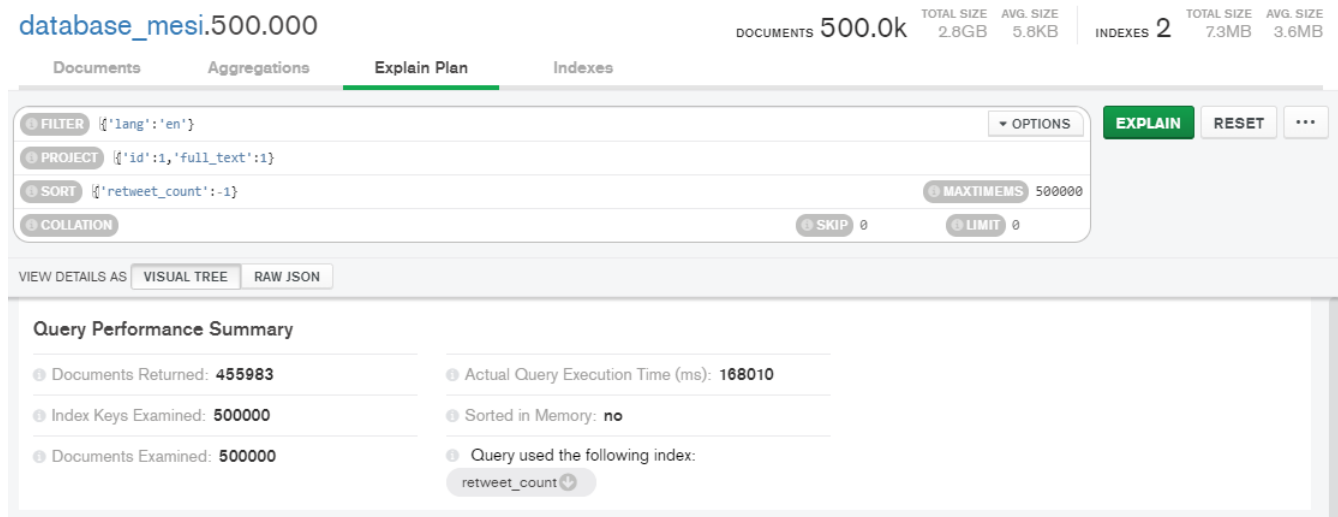


Figura 2.15: Query eseguita su 500.000 documents

Capitolo 3

Databricks

3.1 Preprocessing dei dati

Prima di eseguire la sentiment analysis sui tweet recuperati, è necessaria un'operazione di preprocessing, con il fine di ottenere risultati soddisfacenti. Le operazioni sono state eseguite in **Apache Spark**, un engine analitico unificato per il calcolo distribuito di dati su larga scala, composto da una serie di librerie tra cui spark SQL, Mlib per il machine learning e GraphX per l'elaborazione di grafici.

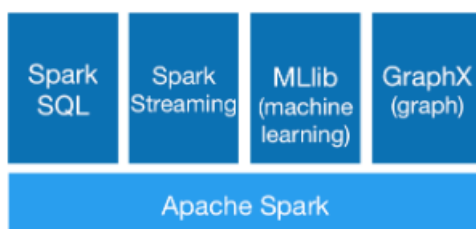


Figura 3.1: Struttura di Apache Spark

Nella colonna “**full-text**” presente nel Dataframe pyspark, possiamo notare come i tweets contengano dei caratteri speciali (@, “, !, ?, /, etc) e dei links a siti web, che potrebbero influire sull'accuratezza della sentiment analysis, non

avendo un significato semantico. Per ridurre le risorse computazionali richieste e aumentare la qualità dei dati, sono stati eliminati i seguenti caratteri:

- Le lettere RT (che stanno per retweet) con il relativo nickname
- I link (https://...)
- I caratteri speciali

Inoltre, per permettere all'algoritmo di riconoscere le parole uguali, è stato convertito ogni testo in minuscolo. Infine, sono stati eliminati i tweet uguali grazie alla funzione `distinct()`.

full_text
RT @OH_mes2: Red Velvet Irene has donated 100 million won to the Community Chest of Korea to help support those affected by the Coronavirus...
RT @MirMAKOfficial: Surgical mask packet goes from Rs 80 to Rs 480 لحت ان لوگوں پر These are the kind of people if there were more death...
RT @mattbc: Preparing for COVID19 while chronically ill / disabled: "As I write this, we do not yet know how bad the outbreak of COVID-19..."
RT @The_NewArab: Opinion - "#Iran's mishandling of coronavirus Covid-19 outbreak is putting its people and the region at risk" argues James...

Figura 3.2: Il testo prima del preprocessing

E' stata notata inoltre la presenza di emoticon in molti tweet; si è pensato quindi di sostituirli con delle etichette che esprimessero il sentiment, positivo o negativo, in base al proprio significato semantico:

```

12 df_clean = ten.select('created_at','full_text','retweet_count')
13
14 #print(UNICODE_EMOJI[emoji])
15 #CREO UNA NUOVA COLONNA TEXT IN CUI METTO IL FULL_TEXT RIPULITO:
16 df_cleantext = df_clean.withColumn("text", regexp_replace('full_text', 'RT @[\\w]*:', ' ')) #elimino gli RT e i relativi nickname
17 df_cleantext2 = df_cleantext.withColumn("text", regexp_replace('text', '@[\\w]*:', ' ')) #elimino tutti i nickname
18 df_cleantext3 = df_cleantext2.withColumn("text", regexp_replace('text', 'https://[A-Za-z0-9./]*', ' ')) #elimino i link
19 df_cleantext4 = df_cleantext3.withColumn("text", regexp_replace('text', '\\n', ' '))
20 df_cleantext5 = df_cleantext4.withColumn("text", regexp_replace('text', '[^ a-zA-ZÀ-Ú' + '\\❤️🔥' + '\\👍👎👏👀', ' '))
21
22
23 df_cleantext6 = df_cleantext5.withColumn("text", regexp_replace('text', '❤️', '||positive|'))
24 df_cleantext7 = df_cleantext6.withColumn("text", regexp_replace('text', '🔥', '||positive|'))
25 df_cleantext8 = df_cleantext7.withColumn("text", regexp_replace('text', '👍', '||positive|'))
26 df_cleantext9 = df_cleantext8.withColumn("text", regexp_replace('text', '👎', '||positive|'))
27 df_cleantext10 = df_cleantext9.withColumn("text", regexp_replace('text', '👏', '||positive|'))
28 df_cleantext11 = df_cleantext10.withColumn("text", regexp_replace('text', '👀', '||negative|'))
29 df_cleantext12 = df_cleantext11.withColumn("text", regexp_replace('text', '👍', '||negative|'))

```

Figura 3.3: Replacing delle emoticon

Ricaveremo come output il seguente campo **“text”**:

	text
r. They have the right to speak out. And we have the right to point out these are coordinated efforts linked to	they have the right to speak out and we have the right to point out these are coordinated
l. KSFPD to implement 'No Mask, No Fuel' rule at retail outlets in Karnataka https://t.co/Qr5ZPCmv5s via @Petr...	ksfpd to implement no mask no fuel rule at retail outlets in karnataka via
I need #handsanitizer 🍷 I am making #masks 🍷 & am willing to trade a mask for a bottle 🍷 Safe #NeedyinNH	i need handsanitizer positive i am making masks amp am willing to trade a mask for a l needyinnh

Figura 3.4: Testo preprocessato

3.2 VADER

VADER (Valence Aware Dictionary and sEntiment Reasoner) è un modello di analisi specifico per i sentimenti espressi nei social media. VADER utilizza un **sentiment lexicon** che è sensibile alla polarità (positiva/negativa) e intensità di un'emozione. E' disponibile all'interno del pacchetto **NLTK** e può essere applicato direttamente a del testo non etichettato.

inserisci immagine

Avendo a disposizione solo dei tweets non etichettati, abbiamo deciso di utilizzare questo modello in modo tale da poter addestrare il nostro classificatore. Calcoliamo il punteggio **Compound** per ogni parola, ovvero una metrica che

calcola la somma dei punteggi di valenza di ogni parola nel lexicon, normalizzati in modo tale da essere compresi tra -1 (più negativo) e +1 (più positivo). Se il compound è:

1. maggiore uguale a 0.05 il sentiment sarà positivo
2. maggiore di -0.05 e minore di 0.05 il sentiment sarà neutro
3. minore uguale di -0.05 il sentiment sarà negativo

```
1 #ETICHETTO I TWEETS CON VADER
2
3 import nltk
4 from nltk.sentiment.vader import SentimentIntensityAnalyzer
5
6 nltk.download('vader_lexicon')
7 sentiment = udf(lambda x: SentimentIntensityAnalyzer().polarity_scores(x)['compound'])
8 spark.udf.register("sentiment", sentiment)
9 df_cleantext14 = df_cleantext13.withColumn("sentiment",sentiment("text").cast("double"))
10
11 #df_cleantext13['polarity'] = df_cleantext13['text'].apply(lambda x: SentimentIntensityAnalyzer().polarity_scores(x)['compound'])
12
13 display(df_cleantext14)
```

Figura 3.5: Funzione per calcolare il compound

text	sentiment
My skin and my hair are tired of me putting masks everyday this quarantine needs to be over	-0.4404
Hey fun quarantine tip Dont put wasabi in your mask Its not as fun as it sounds Dont leave your mask around peopl	0.7749
Trump has been spouting weird scienceHydroxychloroquine and ZpacksGet the zincHe claimed new cases were going	-0.1779
The hooded men concept is hinted in one of their previous shows before bts cb and even before that you can see	0

Figura 3.6: Sentiment ottenuto

Successivamente abbiamo creato una funzione che in base al punteggio restituisse un'etichetta positiva, negativa o neutra:


```

1 # Definisco una funzione che calcola la sentiment in base ai valori degli scores
2 def evaluateSentiment(sentiment):
3     if sentiment >= 0.05:
4         return "positive"
5     elif sentiment <= -0.05:
6         return "negative"
7     elif sentiment > -0.05 and sentiment < 0.05:
8         return "neutral"
9
10 label = udf(lambda x: evaluateSentiment(x))
11 spark.udf.register("label", label)
12 df_cleantext15 = df_cleantext14.withColumn("label", label("sentiment"))
13 #df_cleantext15['result'] = df_cleantext14['sentiment'].apply(lambda x: evaluateSentiment(x))
14 display(df_cleantext15)

```

Figura 3.7: Labeling

text	sentiment	label
My skin and my hair are tired of me putting masks everyday this quarantine needs to be over	-0.4404	negative
Hey fun quarantine tip Dont put wasabi in your mask Its not as fun as it sounds Dont leave your mask around peopl	0.7749	positive
Trump has been spouting weird scienceHydroxychloroquine and ZpacksGet the zincHe claimed new cases were going	-0.1779	negative
The hooded men concept is hinted in one of their previous shows before bts cb and even before that you can see	0	neutral

Figura 3.8: Testi etichettati

3.3 Spark NLP

Spark NLP è una libreria di elaborazione testi open source per l'elaborazione avanzata del linguaggio naturale per i linguaggi di programmazione Python, Java e Scala. In particolare, è stato utilizzato “**SentimentDL**”, un modello che permette l'elaborazione del testo di ogni tweet con il fine di etichettarlo come “negativo” “positivo” o “neutro” in base al punteggio assegnatogli:

- Se maggiore di 0.5 è etichettato come positivo
- Se minore di 0.5 come negativo
- Se uguale a 0.5 come neutro

La pipeline è composta da una serie di **trasformatori**, ovvero degli algoritmi che trasformano un DataFrame in un altro DataFrame (aggiungendo nell'ultimo passaggio la colonna di predizione del sentiment). Inizialmente verrà utilizzato il **SentimentDLApproach()** nel quale porremo in ingresso il training set etichettato in modo tale da addestrare il modello, dopodichè nella fase di predizione caricheremo il modello così addestrato per effettuare la sentiment analysis sui dataset creati grazie al comando load. Avremo quindi:

- **Document assembler:** permette di trasformare i dati grezzi passati in ingresso, che nel nostro caso sono rappresentati dalla colonna “full-text”, in un file di tipo “Document”. In particolare, l'annotatore utilizzato è un trasformatore, ovvero un algoritmo che trasforma un DataFrame in un altro DataFrame (aggiungendo in questo caso la colonna di predizione del sentiment).
- **Tokenizer:** converte un testo in tokens (parole).

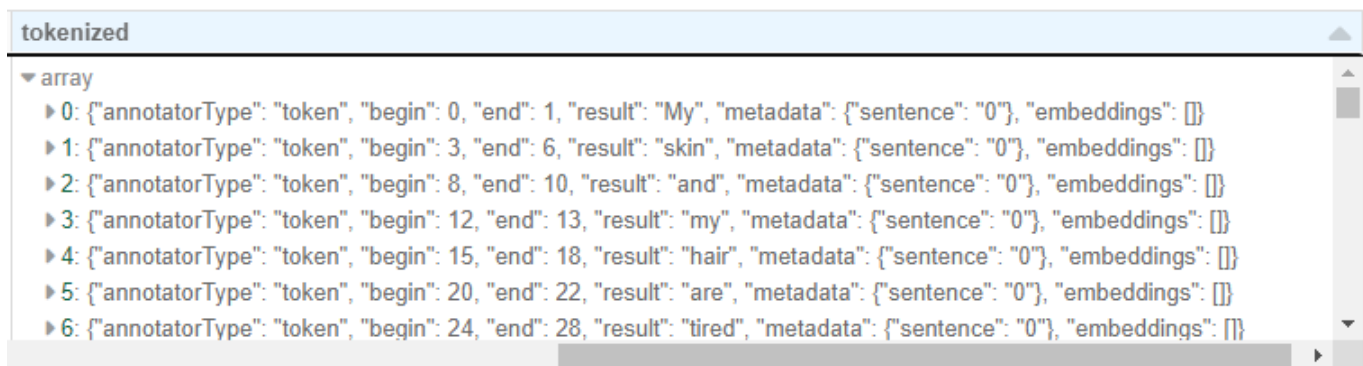


Figura 3.9: Tokenizer

- **Normalizer:** converte il testo in minuscolo in modo tale da permettere all'algoritmo di riconoscere le parole uguali.

	normalized
<pre> it": "DEAR", "metadata": {"sentence": "0"}, "embeddings": []}, it": "HOLLYWOODWE", "metadata": {"sentence": "0"}, "embeddings": []}, it": "WILL", "metadata": {"sentence": "0"}, "embeddings": []}, it": "RIP", "metadata": {"sentence": "0"}, "embeddings": []}, it": "OFF", "metadata": {"sentence": "0"}, "embeddings": []}, it": "YOUR", "metadata": {"sentence": "0"}, "embeddings": []}, it": "MASKSWE", "metadata": {"sentence": "0"}, "embeddings": []}, it": "WILL", "metadata": {"sentence": "0"}, "embeddings": []}, it": "SPEAK", "metadata": {"sentence": "0"}, "embeddings": []}. </pre>	<pre> {"annotatorType": "token", "begin": 1, "end": 4, "result": "dear", "metadata": {"sentence": "0"}, "embeddings": []}, {"annotatorType": "token", "begin": 6, "end": 16, "result": "hollywoodwe", "metadata": {"sentence": "0"}, "embeddings": []}, {"annotatorType": "token", "begin": 18, "end": 21, "result": "will", "metadata": {"sentence": "0"}, "embeddings": []}, {"annotatorType": "token", "begin": 23, "end": 25, "result": "rip", "metadata": {"sentence": "0"}, "embeddings": []}, {"annotatorType": "token", "begin": 27, "end": 29, "result": "off", "metadata": {"sentence": "0"}, "embeddings": []}, {"annotatorType": "token", "begin": 31, "end": 34, "result": "your", "metadata": {"sentence": "0"}, "embeddings": []}, {"annotatorType": "token", "begin": 36, "end": 42, "result": "maskswe", "metadata": {"sentence": "0"}, "embeddings": []}, {"annotatorType": "token", "begin": 45, "end": 48, "result": "will", "metadata": {"sentence": "0"}, "embeddings": []}, {"annotatorType": "token", "begin": 50, "end": 54, "result": "speak", "metadata": {"sentence": "0"}, "embeddings": []}. </pre>

Figura 3.10: Normalizer

- **StopWords Cleaner:** elimina gli articoli determinativi, indeterminativi, le congiunzioni, e tutte le parole prive di significato semantico utile per l'analisi.

	cleanTokens
<pre> 3, "end": 6, "result": "skin", "metadata": {"sentence": "0"}, "embeddings": []}, 8, "end": 10, "result": "and", "metadata": {"sentence": "0"}, "embeddings": []}, 12, "end": 13, "result": "my", "metadata": {"sentence": "0"}, "embeddings": []}, 15, "end": 18, "result": "hair", "metadata": {"sentence": "0"}, "embeddings": []}, 20, "end": 22, "result": "are", "metadata": {"sentence": "0"}, "embeddings": []}, 24, "end": 28, "result": "tired", "metadata": {"sentence": "0"}, "embeddings": []}, 30, "end": 31, "result": "of", "metadata": {"sentence": "0"}, "embeddings": []}, 33, "end": 34, "result": "me", "metadata": {"sentence": "0"}, "embeddings": []} </pre>	<pre> 1: {"annotatorType": "token", "begin": 15, "end": 18, "result": "hair", "metadata": {"sentence": "0"}, "embeddings": []}, 2: {"annotatorType": "token", "begin": 24, "end": 28, "result": "tired", "metadata": {"sentence": "0"}, "embeddings": []}, 3: {"annotatorType": "token", "begin": 36, "end": 42, "result": "putting", "metadata": {"sentence": "0"}, "embeddings": []}, 4: {"annotatorType": "token", "begin": 44, "end": 48, "result": "masks", "metadata": {"sentence": "0"}, "embeddings": []}, 5: {"annotatorType": "token", "begin": 50, "end": 57, "result": "everyday", "metadata": {"sentence": "0"}, "embeddings": []}, 6: {"annotatorType": "token", "begin": 64, "end": 73, "result": "quarantine", "metadata": {"sentence": "0"}, "embeddings": []}, 7: {"annotatorType": "token", "begin": 75, "end": 79, "result": "needs", "metadata": {"sentence": "0"}, "embeddings": []} </pre>

Figura 3.11: StopWordsCleaner

- **Word Embeddings:** permette di costruire uno spazio vettoriale in cui i vettori delle parole sono più vicini se le parole occorrono negli stessi contesti linguistici, cioè se sono riconosciute come semanticamente più simili.

	embeddings
<pre> "skin", "metadata": {"sentence": "0"}, "embeddings": []}, "hair", "metadata": {"sentence": "0"}, "embeddings": []}, "tired", "metadata": {"sentence": "0"}, "embeddings": []}, "putting", "metadata": {"sentence": "0"}, "embeddings": []}, "masks", "metadata": {"sentence": "0"}, "embeddings": []}, "everyday", "metadata": {"sentence": "0"}, "embeddings": []}, "quarantine", "metadata": {"sentence": "0"}, "embeddings": []}, "needs", "metadata": {"sentence": "0"}, "embeddings": []} </pre>	<pre> {"annotatorType": "word_embeddings", "begin": 3, "end": 6, "result": "skin", "metadata": {"isOOV": "false", "pieceId": "-1", "isWordStart": "true", "token": "skin", "sentence": "0"}, "embeddings": [-0.56765, 0.15144, 0.08739, -0.58177, -0.36405, 0.25263, 0.19216, 0.33921, 0.20811, 0.0036447, -0.36843, -0.1946, 1.2074, 0.65129, 0.85408, 0.61007, -0.44654, -0.79221, 0.99169, -0.57925, -0.52479, -0.12186, -0.68424, -0.24807, 0.79997, 1.8644, 0.88059, -1.1063, 0.0036065, -0.54901, 0.4942, 0.70751, -0.27179, -0.20569, 0.17593, 0.46356, -0.30265, 0.15677, -0.20668, 0.50407, -0.15696, -1.0547, -0.55695, -0.66206, -0.37376, 0.7299, -0.030611, 0.7426, -0.21264, -0.83752, 0.30062, 0.19039, -0.16993, 1.0665, -0.29448, -1.1214, 0.17906, -0.15797, 0.64386, 0.21188, 0.75228, 1.8364, -0.0076749, 0.53663, 1.1772, 0.20493, 0.65759, -0.62886, 0.1556, -1.161, 0.12446, 0.46445, 0.32222, 0.42523, 0.64108, -0.35768, -0.72128, 0.048876, 0.44941, 0.27713, 0.24611, 0.1984, -0.3408, 1.0668, -1.2682, -0.24086, 0.48103, 0.68491, -0.2286, 0.9002, 0.39679, -0.49903, 0.45872, -0.098798, 0.092992, 0.36079, -0.57676, -0.8221, -0.52318, -0.85611]}, {"annotatorType": "word_embeddings", "begin": 15, "end": 18, "result": "hair", "metadata": {"isOOV": "false", "pieceId": "-1", "isWordStart": "true", "token": "hair", "sentence": "0"}, "embeddings": [-0.56765, 0.15144, 0.08739, -0.58177, -0.36405, 0.25263, 0.19216, 0.33921, 0.20811, 0.0036447, -0.36843, -0.1946, 1.2074, 0.65129, 0.85408, 0.61007, -0.44654, -0.79221, 0.99169, -0.57925, -0.52479, -0.12186, -0.68424, -0.24807, 0.79997, 1.8644, 0.88059, -1.1063, 0.0036065, -0.54901, 0.4942, 0.70751, -0.27179, -0.20569, 0.17593, 0.46356, -0.30265, 0.15677, -0.20668, 0.50407, -0.15696, -1.0547, -0.55695, -0.66206, -0.37376, 0.7299, -0.030611, 0.7426, -0.21264, -0.83752, 0.30062, 0.19039, -0.16993, 1.0665, -0.29448, -1.1214, 0.17906, -0.15797, 0.64386, 0.21188, 0.75228, 1.8364, -0.0076749, 0.53663, 1.1772, 0.20493, 0.65759, -0.62886, 0.1556, -1.161, 0.12446, 0.46445, 0.32222, 0.42523, 0.64108, -0.35768, -0.72128, 0.048876, 0.44941, 0.27713, 0.24611, 0.1984, -0.3408, 1.0668, -1.2682, -0.24086, 0.48103, 0.68491, -0.2286, 0.9002, 0.39679, -0.49903, 0.45872, -0.098798, 0.092992, 0.36079, -0.57676, -0.8221, -0.52318, -0.85611]} </pre>

Figura 3.12: WordEmbeddings

- **Universal Sentence Encoder:** trasforma il testo in high dimensional vectors che possono essere utilizzati per la classificazione del testo, la somiglianza semantica, il cluster e altre attività per il linguaggio naturale.

	use_embeddings
d": 6, "result": "skin", "metadata": {"isOOV": "false", "pieceld": "-1", mbeddings": [-0.56765, 0.15144, 0.08739, -0.58177, -0.36405, 0.25263, 46, 1.2074, 0.65129, 0.85408, 0.61007, -0.44654, -0.79221, 0.99169, 997, 1.8644, 0.88059, -1.1063, 0.0036065, -0.54901, 0.4942, 0.70751, 77, -0.20668, 0.50407, -0.15696, -1.0547, -0.55695, -0.66206, -0.37376, 2, 0.19039, -0.16993, 1.0665, -0.29448, -1.1214, 0.17906, -0.15797, 3, 1.1772, 0.20493, 0.65759, -0.62886, 0.1556, -1.161, 0.12446, 8, 0.048876, 0.44941, 0.27713, 0.24611, 0.1984, -0.3408, 1.0668,).39679, -0.49903, 0.45872, -0.098798, 0.092992, 0.36079, -0.57676, nbeddings" "begin": 15 "end": 18 "result": "hair" "metadata": {"isOOV":	▶ [{"annotatorType": "sentence_embeddings", "begin": 0, "end": 90, "result": "My skin and my hair are tired of me putting masks everyday this quarantine needs to be over", "metadata": {"sentence": "0", "token": "My skin and my hair are tired of me putting masks everyday this quarantine needs to be over", "pieceld": "-1", "isWordStart": "true"}, "embeddings": [0.016684804, -0.059022073, -0.016696667, 0.011254006, -0.0218639, 0.037781477, 0.031142984, -0.0075261863, 0.029381393, -0.023907265, 0.03154826, 0.031458426, 0.09262305, -0.019625599, 0.022385541, -0.016741017, -0.02170135, -0.05289585, -0.058917835, 0.0927383, 0.059317257, -0.06344067, -0.005489867, -0.03816651, -0.028992178, -0.02020262, 0.047015026, -0.022449516, 0.022593953, 0.020269008, -0.08483741, 0.027685566, 0.03519335, -0.0075591505, -0.042027492, -0.05643158, -0.090646744, 0.000727128, 0.011712143, 0.0078068348, -0.024090946, 0.027929312, 0.00091612333, -0.0011945515, 0.018713228, 0.033177197, -0.04382335, 0.019999336, -0.010410875, 0.019595267, 0.008900415, -0.0037629048, -0.057220355, 0.031715404, 0.028949555, 0.052116353, 0.04899462, -0.07008181, 0.016225101

Figura 3.13: Universal Sentence Encoder

- **SentimentDLApproach:** un annotatore che effettua la sentiment analysis, permettendo la predizione del sentiment di ogni testo in base ad un punteggio assegnato. Il dataset in ingresso è stato suddiviso in training-set all'80 per cento e validation-set al 20 per cento e il numero di epoche è pari a 5.

```

14 documentAssembler = DocumentAssembler() \
15     .setInputCol("text") \
16     .setOutputCol('document')
17
18 tokenizer = Tokenizer() \
19     .setInputCols(['document']) \
20     .setOutputCol('tokenized')
21
22 normalizer = Normalizer() \
23     .setInputCols(['tokenized']) \
24     .setOutputCol('normalized') \
25     .setLowercase(True)
26
27 stopwords_cleaner = StopWordsCleaner() \
28     .setInputCols(['normalized']) \
29     .setOutputCol('cleanTokens') \
30     .setStopWords(eng_stopwords)
31
32 word_embeddings = WordEmbeddingsModel().pretrained()\
33     .setInputCols("document", "cleanTokens") \
34     .setOutputCol("embeddings")
35
36 use = UniversalSentenceEncoder.pretrained() \
37     .setInputCols("document", "embeddings") \
38     .setOutputCol("use_embeddings")
39
40 sentimentdl = SentimentDLApproach()\
41     .setInputCols("use_embeddings")\
42     .setOutputCol("result")\
43     .setMaxEpochs(5)\
44     .setValidationSplit(0.2)\
45     .setLabelColumn("label")\
46     .setEnableOutputLogs(True)

```

Figura 3.14: Pipeline per il training

```

Training started - total epochs: 5 - learning rate: 0.005 - batch size: 64 - training examples: 24097
Epoch 1/5 - 9.08s - loss: 206.24072 - accuracy: 0.7422857 - validation: 76.128815 - batches: 377
2020-08-22T23:36:39.797+0000: [GC (Allocation Failure) [PSYoungGen: 1479802K->195253K(2068992K)] 4199788K->2919195K(7720448K), 0.0771999 secs] [Times: user=0.14
sys=0.00, real=0.08 secs]
Epoch 2/5 - 8.72s - loss: 198.10564 - accuracy: 0.7747459 - validation: 76.311424 - batches: 377
Epoch 3/5 - 8.58s - loss: 192.59322 - accuracy: 0.79044896 - validation: 76.54382 - batches: 377
2020-08-22T23:37:02.656+0000: [GC (Allocation Failure) [PSYoungGen: 1504949K->192453K(2065920K)] 4228891K->2920112K(7717376K), 0.0821384 secs] [Times: user=0.15
sys=0.00, real=0.08 secs]
Epoch 4/5 - 8.96s - loss: 187.92642 - accuracy: 0.80760896 - validation: 76.726425 - batches: 377
Epoch 5/5 - 8.57s - loss: 183.40767 - accuracy: 0.8225691 - validation: 76.92563 - batches: 377

```

Figura 3.15: Epoche con relativa accuracy

	text	result
1	Working in a hospital seeing first hand what COVID is doing having daily updates of infection amp death tolls knowing what it feels like to wear an N mask until it leaves marks and seeing people STILL not taking it seriously makes me angry in a way I can't put into words	► ["negative"]
2	Compulsory masks physical distancing is a must have Given the current situation even if shutdown is relaxed of the population will venture out Add masks to that and the curve will get crushed Same solution during Spanish flu	► ["negative"]
3	How does one coordinate bow tie pocket square and mask	► ["positive"]
4	It is now mandatory to wear a face mask in Durham in places such as grocery stores pharmacies businesses and public trans	► ["negative"]
5	Hey journalists covering the COVID pandemic check out CPJEmergencies safety advisory and resources Find tips for	► ["positive"]

Figura 3.16: Risultato della predizione

3.4 Evaluation

Per valutare l'accuratezza del modello utilizzato, è stata effettuata la sentiment analysis su un test set costituito da 75000 tweets mai analizzato prima. Dopo il preprocessing e l'etichettatura, creiamo una nuova pipeline con il modello addestrato precedentemente:

```

41 sentimentdl = SentimentDLModel.load("dbfs:/tmp_sentimentdl_model") \
42   .setInputCols(["use_embeddings"]) \
43   .setOutputCol("class")
44
45 pipeline = Pipeline() \
46   .setStages([documentAssembler,
47               tokenizer,
48               normalizer,
49               stopwords_cleaner,
50               word_embeddings,
51               use,
52               sentimentdl])

```

Figura 3.17: Pipeline addestrata

	label	text	predicted
132	positive	looks good	positive
133	negative	chinas coverup of covid and their continued lies about its origin have shown us all firsthand that we cannot trust	negative
134	positive	we are celebrating students across the uk who are going above and beyond in the fight against covid wayne was nominated by his colleague matthew for his dedication to healthcare services	positive
135	positive	we love this video and its message to keep physical distancing well done pandemic parody	positive
136	positive	give us liberty so that we can catch the coronavirus	positive
137	positive	covid has led to clearer skies in cities around the world but in a way nobody would ever want when its time to rebuild lets make sure we cut pollution the right way build better together	positive

Figura 3.18: Etichette predette per il test set

Importiamo la libreria **sklearn** per poter calcolare l'accuracy del modello. Il risultato ottenuto è stato del 77 per cento:

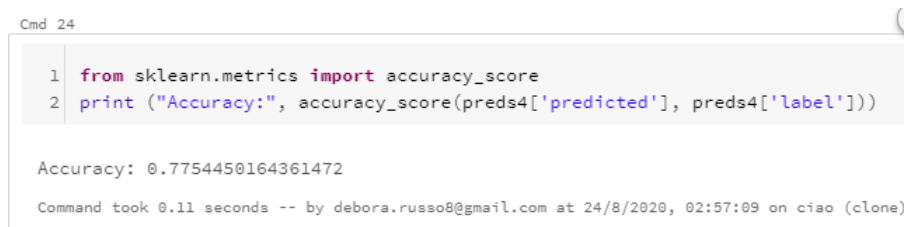
A terminal window titled 'Cmd 24' with a close button. It contains two lines of Python code: '1 from sklearn.metrics import accuracy_score' and '2 print ("Accuracy:", accuracy_score(preds4['predicted'], preds4['label']))'. Below the code, the output is 'Accuracy: 0.7754450164361472'. At the bottom, a status bar reads 'Command took 0.11 seconds -- by debora.russo8@gmail.com at 24/8/2020, 02:57:09 on ciao (clone)'.

Figura 3.19: Accuracy del modello

3.5 Rielaborazione dell'output

Per attribuire conoscenza e valore ai dati prelevati, è stata necessaria una loro opportuna rielaborazione in modo tale da poterli rappresentare mediante grafici e tabelle all'interno del software PowerBI.

3.5.1 Collection 1 - Conteggio dei tweets nei giorni 28 febbraio, 30 marzo e aprile

Sono stati raggruppati i tweet positivi, negativi e neutri di questi tre giorni con l'obiettivo di valutare l'andamento delle reazioni lungo il diffondersi del coronavirus. Dopo aver invocato la funzione `split` per dividere ogni riga contenente la data e creato due ulteriori colonne contenenti il giorno e il mese, sono state create tre variabili di appoggio per il conteggio dei tweets positivi, negativi e neutri. Per ogni colonna è stato associato un valore 1 in base al sentiment indicato e infine è stata utilizzata la funzione `sum()` per ottenere il numero totale di tweet positivi, negativi e neutri, raggruppando le righe per giorno e mese.

```

6 df_final = pipeline.fit(df_cleantext).transform(df_cleantext).select("created_at","sentiment.result")
7
8 split_columns = f.split(df_final["created_at"], " ")
9 df_final = df_final.withColumn("day", split_columns.getItem(2))
10 df_final = df_final.withColumn("month", split_columns.getItem(1))
11 df_final = df_final.withColumn("sentiment", df_final.result.getItem(0))
12
13 df_final = df_final.withColumn("positive_0" , regexp_replace('sentiment','positive', '1'))
14 df_final = df_final.withColumn("positive_1" , regexp_replace('positive_0','negative', '0'))
15 df_final = df_final.withColumn("positive_2" , regexp_replace('positive_1','neutral', '0'))
16
17 df_final = df_final.withColumn("negative_0" , regexp_replace('sentiment','negative', '1'))
18 df_final = df_final.withColumn("negative_1" , regexp_replace('negative_0','positive', '0'))
19 df_final = df_final.withColumn("negative_2" , regexp_replace('negative_1','neutral', '0'))
20
21 df_final = df_final.withColumn("neutral_0" , regexp_replace('sentiment','neutral', '1'))
22 df_final = df_final.withColumn("neutral_1" , regexp_replace('neutral_0','positive', '0'))
23 df_final = df_final.withColumn("neutral_2" , regexp_replace('neutral_1','negative', '0'))
24
25 df_final = df_final.withColumn("positive", expr('CAST(positive_2 AS INTEGER)'))
26 df_final = df_final.withColumn("negative" , expr('CAST(negative_2 AS INTEGER)'))
27 df_final = df_final.withColumn("neutral" , expr('CAST(neutral_2 AS INTEGER)'))
28
29 df_final_sum = df_final.groupby("day","month").sum("positive","negative","neutral").orderBy("day")
30
31
32 display(df_final_sum)

```

Figura 3.20: Codice per la somma del sentiment

L'output ottenuto è il seguente:

	day ▲	month ▲	sum(positive)▲	sum(negative)▲	sum(neutral)▲
1	08	Apr	123917	205129	3292
2	28	Feb	106634	220383	4266
3	30	Mar	136118	189451	6633

Figura 3.21: Output del sentiment per i 3 giorni di Febbraio,Marzo e Aprile

3.5.2 Collection 1 - Sentiment dei tweets più retweetati

Successivamente sono stati prelevati i tweet più retweetati per confrontarli con il sentiment medio dei tre giorni citati prima:


```

11 df_final = pipeline.fit(df_cleantext5).transform(df_cleantext5).select("created_at","text","sentiment.result","retweet_count")
12 #3-SELEZIONO LE PRIME 10 RIGHE DEL DATAFRAME CHE È GIÀ ORDINATO IN BASE AL NUMERO DI TWEET PIÙ RETWITTATI:
13 #df_final2 = df_final.take(10)
14 #display(df_final2)
15
16 split_columns = f.split(df_final["created_at"], " ")
17 df_final = df_final.withColumn("day", split_columns.getItem(2))
18 df_final = df_final.withColumn("month", split_columns.getItem(1))
19 df_final = df_final.withColumn("sentiment", df_final.result.getItem(0))
20 display(df_final)

```

Figura 3.22: Codice per ottenere i tweet più retweetati

Ottenendo il seguente risultato:

	text	retweet_count	day	month	sentiment
1	this is fucking bullshit	327666	30	Mar	negative
2	This morning I tested positive for Covid I feel ok I have no symptoms so far but have been isolated since I found out a	304526	30	Mar	negative
3	The World Health Organization has announced that dogs cannot contract Covid Dogs previously held in quarantine can now	247762	08	Apr	negative
4	covid please stop your world tour	231653	08	Apr	neutral
5	this is the BEST set of info I ve found on corona virus and it s worth watching the full mins	201080	30	Mar	positive
6	Coronavirus has crossed the line for Italians	201040	08	Apr	negative
7	my favourite consequence of the covid societal lockdown is aquarium penguins roaming around freely and when this is	188269	30	Mar	negative

Figura 3.23: Tweet in ordine di numero di retweet

3.5.3 Collection 1 - Hashtags più utilizzati

Sono stati selezionati gli hashtags più utilizzati, ottenuti esplodendo l'array entities grazie alla funzione explode(), che permette di dividere in colonne i campi di un array:

```

1 from pyspark.sql.functions import explode_outer, explode, posexplode_outer
2 from pyspark.sql.types import DoubleType
3 from pyspark.sql.functions import desc
4
5 df_final = hashtag.select("entities.hashtags.text")
6 df_final.printSchema()
7 df_final2 = df_final.select(explode(df_final.text).alias("hashtags"))
8 df_final3 = df_final2.select("hashtags")
9 #df_final4 = df_final3.withColumn("hashtags", df_final3["hashtags"].cast(DoubleType()))
10 #df_final2 = df_final.select("result","day","month",explode(df_final.metadata))
11 #display(df_final3)
12 df_final_sum = df_final3.groupby("hashtags").count()
13 df_final_sum2 = df_final_sum.sort(desc("count"))
14 #df_final_sum2 = df_final_sum.withColumn("count")
15 display(df_final_sum2)

```

Figura 3.24: Codice per estrarre gli Hashtags più utilizzati

	hashtags ▲	count ▲	
1	COVID19	3434	
2	coronavirus	1632	
3	lockdown	1084	
4	COVID—19	933	
5	Coronavirus	483	
6	PyaarKarna	430	
7	StaySafe	387	

Figura 3.25: Hashtags più utilizzati

3.5.4 Collection 2 - Sentiment medio positivo e negativo durante la settimana 20-26 aprile

L'array contenente i punteggi assegnati ad ogni tweet è stato esploso in modo tale da poter analizzare l'andamento medio del sentiment durante la settimana 20-26 aprile, considerata di picco durante i mesi di febbraio, marzo e aprile. Per fare questo è stata utilizzata la funzione explode che ha permesso di ottenere separatamente le due colonne "positive" e "negative".

```

2 from pyspark.sql.functions import explode_outer, explode, posexplode_outer
3 from pyspark.sql.functions import expr, col
4 from pyspark.sql.types import DoubleType
5
6 df_final = pipeline.fit(df_duplicates).transform(df_duplicates).select("created_at","sentiment.result","sentiment.metadata")
7 split_columns = f.split(df_final["created_at"], " ")
8 df_final = df_final.withColumn("day", split_columns.getItem(2))
9 df_final = df_final.withColumn("month", split_columns.getItem(1))
10
11 df_final2 = df_final.select("result","day","month",explode(df_final.metadata))
12 df_final3 = df_final2.select("result","day","month","col.positive","col.negative")
13
14 df_final4 = df_final3.withColumn("positive", df_final3["positive"].cast(DoubleType()))
15 df_final5 = df_final4.withColumn("negative", df_final3["negative"].cast(DoubleType()))
16
17 #df_final_sum = df_final5.groupby("day").avg("positive","negative").orderBy("day")
18 df_final6 = df_final5.orderBy("day")
19 display(df_final6)
20
21
22 df_final6.write.format("com.mongodb.spark.sql.DefaultSource").mode("append").option("database","covid19geo").option("collection","averageSentimentAvg").save()

```

Figura 3.26: Codice per ottenere i punteggi di sentiment assegnati ad ogni testo

	result ▲	day ▲	month ▲	positive ▲	negative ▲
1	▶["negative"]	20	Apr	1.4949785e-21	1
2	▶["negative"]	20	Apr	5.009972e-9	1
3	▶["positive"]	20	Apr	1	2.9051494e-33
4	▶["negative"]	20	Apr	0.000101174504	0.9998988
5	▶["negative"]	20	Apr	9.430282e-17	1
6	▶["negative"]	20	Apr	9.0008804e-8	0.9999999
7	▶["positive"]	20	Apr	1	3.4505284e-26

Figura 3.27: Codice per ottenere i punteggi di sentiment assegnati ad ogni testo

3.5.5 Collection 2 - Sentiment medio per ogni Nazione

Per calcolare il sentiment medio di ogni nazione, sono state filtrate tutte le righe la cui posizione fosse nulla:

```
df_cleanPlace = df_clean.where(col("place.country").isNotNull())
```

Figura 3.28: Filtraggio righe il cui campo "place" sia nullo

Successivamente sono stati prelevati i valori di sentiment:

	country ▲	result ▲	metadata ▲
1	United Kingdom	▶["negative"]	▶[{"sentence": "0", "positive": "0.0", "negative": "1.0"}]
2	South Africa	▶["negative"]	▶[{"sentence": "0", "positive": "0.0", "negative": "1.0"}]
3	United States	▶["negative"]	▶[{"sentence": "0", "positive": "0.0", "negative": "1.0"}]
4	United States	▶["positive"]	▶[{"sentence": "0", "positive": "1.0", "negative": "3.7317043E-13"}]
5	Canada	▶["negative"]	▶[{"sentence": "0", "positive": "2.5007377E-34", "negative": "1.0"}]
6	United Kingdom	▶["negative"]	▶[{"sentence": "0", "positive": "1.0498619E-5", "negative": "0.9999895"}]

Figura 3.29: Punteggio e Paese di provenienza di una serie di tweet

3.6 Confronto tra Pyspark e Pandas

Si vuole porre particolare attenzione sulle capacità di **scalabilità** dell'ambiente Spark, ovvero la capacità di un sistema di elaborazione di aumentare o diminuire

la propria dimensione in funzione della necessità e possibilità del sistema stesso. E' stato eseguito lo stesso script scritto rispettivamente in Pyspark e Python per il recupero degli hashtags più utilizzati, aumentando di volta in volta il numero di tweet:

```
15 # The applied options are for CSV files. For other file types, these will be ignored.
16 df = spark.read.format(file_type) \
17     .option("header", first_row_is_header) \
18     .option("sep", delimiter) \
19     .option("multiline", multiline) \
20     .option("escape", "\\\"") \
21     .load(filePath)
22
23 #display(df)
24 #df2.show(60)
25 #df.printSchema()
26
27 df1 = df.select('_c1')
28
29 df2 = df1.filter(df1._c1 != 'entities.hashtags')
30 df3 = df2.withColumn('_c1',explode(split('_c1','\\}',\\{'})))
31 df4 = df3.filter(df3._c1 != '[]')
32 split_columns = f.split(df4["_c1"], ",")
33 df5 = df4.withColumn("hashtags", split_columns.getItem(0))
34 df6 = df5.drop('_c1')
35 df7 = df6.withColumn('hashtags', regexp_replace('hashtags', '[\s\S]*"text":', ''))
36 df8 = df7.withColumn('hashtags', regexp_replace('hashtags', '"', ''))
37 df_final_sum = df8.groupby("hashtags").count().sort(desc("count"))
38 display(df_final_sum)
```

Figura 3.30: Codice Pyspark

```

1 import pandas as pd
2 from datetime import datetime
3 startTime = datetime.now()
4
5 filePath = "/dbfs/FileStore/tables/mesi250mila.csv"
6 file_type = "csv"
7
8 dfhash = pd.read_csv(filePath)
9 dtype={'_id': int}
10 low_memory=False
11 dfhash.dropna(inplace = True)
12 print(dfhash.head(4))
13
14 df1 = dfhash['entities.hashtags']
15 df2 = pd.DataFrame(df1.str.split('},{', expand=True))
16 df3 = df2[df2 != '[]']
17 df4 = df3.melt()
18 df5 = df4.dropna()
19 df6 = pd.DataFrame(df5.value.str.split('"', expand=True))
20 df7 = df6.drop(columns=1)
21 df8 = df7[0].str.replace('[\s\S]*"text":', '', regex = True)
22 df_final_sum = df8.value_counts()
23 |
24
25
26 print(datetime.now()-startTime)

```

Figura 3.31: Codice Python

Si nota come il tempo di esecuzione dello script in Python aumenti sempre di più, mentre in Pyspark rimane pressochè stabile sugli 0,6 secondi:

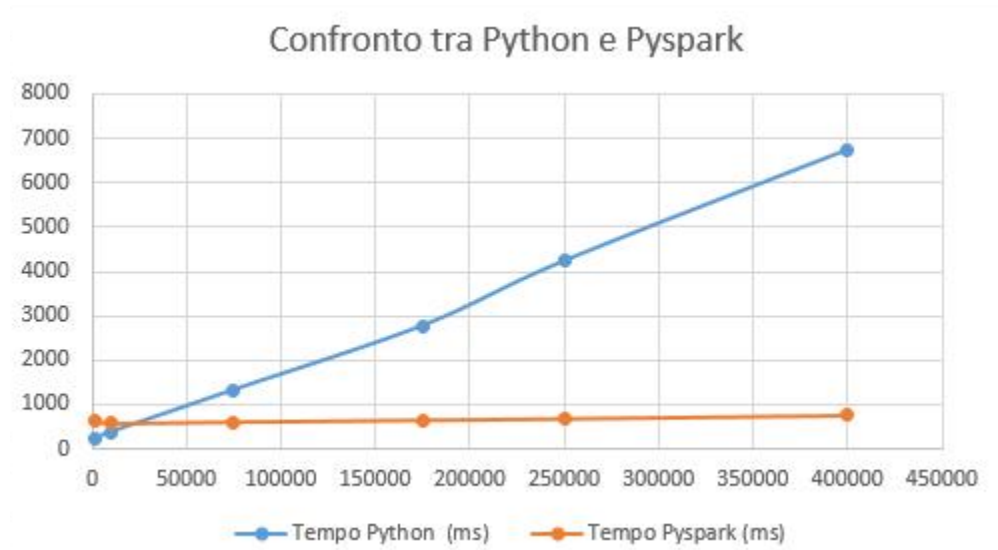


Figura 3.32: Confronto Python e Pyspark

Capitolo 4

Report in PowerBI

4.1 Power BI

Power BI è una raccolta di servizi software, app e connettori che interagiscono per trasformare le origini dei dati non correlate in un insieme di informazioni coerenti, visivamente accattivanti e interattive.

Una volta salvati i file all'interno di MongoDB Atlas, essi sono stati scaricati e inseriti all'interno del software Power BI Desktop in modo tale da proporre un'analisi descrittiva della sentiment analysis riguardante i tweet attinenti al coronavirus, grazie a delle rappresentazioni grafiche come tabelle, grafici etc.

Le prime due pagine del report riguardano i tweets del 28 febbraio, 30 marzo e 8 aprile, mentre le altre si riferiscono alla settimana 20-26 aprile.

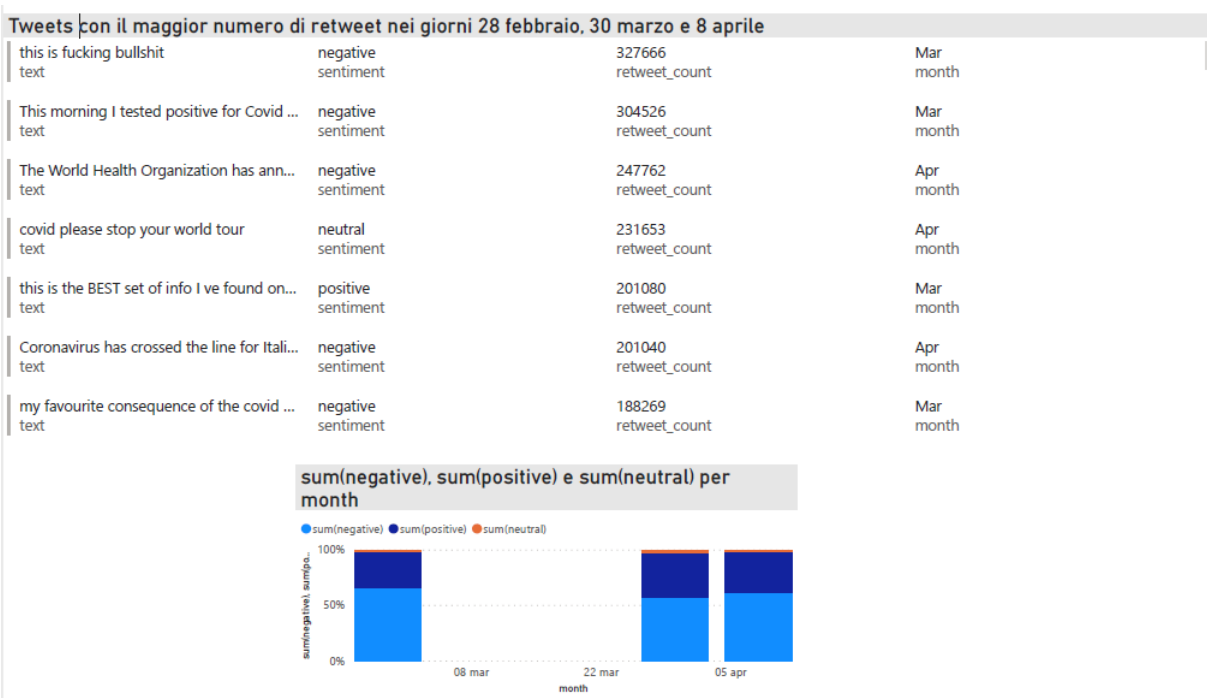


Figura 4.1: Tweet più retweetati e somma dei tweet in base alla sentiment

E' possibile notare che i tweet più retweetati siano stati scritti nei mesi di Marzo e di Aprile, e che siano per la maggioranza negativi. Questo probabilmente è dovuto al fatto che il numero dei casi in tutto il mondo sia aumentato esponenzialmente rispetto a Febbraio, e quindi se ne è parlato sempre di più. Dalla somma dei tweet in basso inoltre, vediamo come la percentuale di tweet negativi sia sempre maggiore del 50 per cento per tutti e tre i mesi. I tweet neutrali invece sono sempre pochissimi.

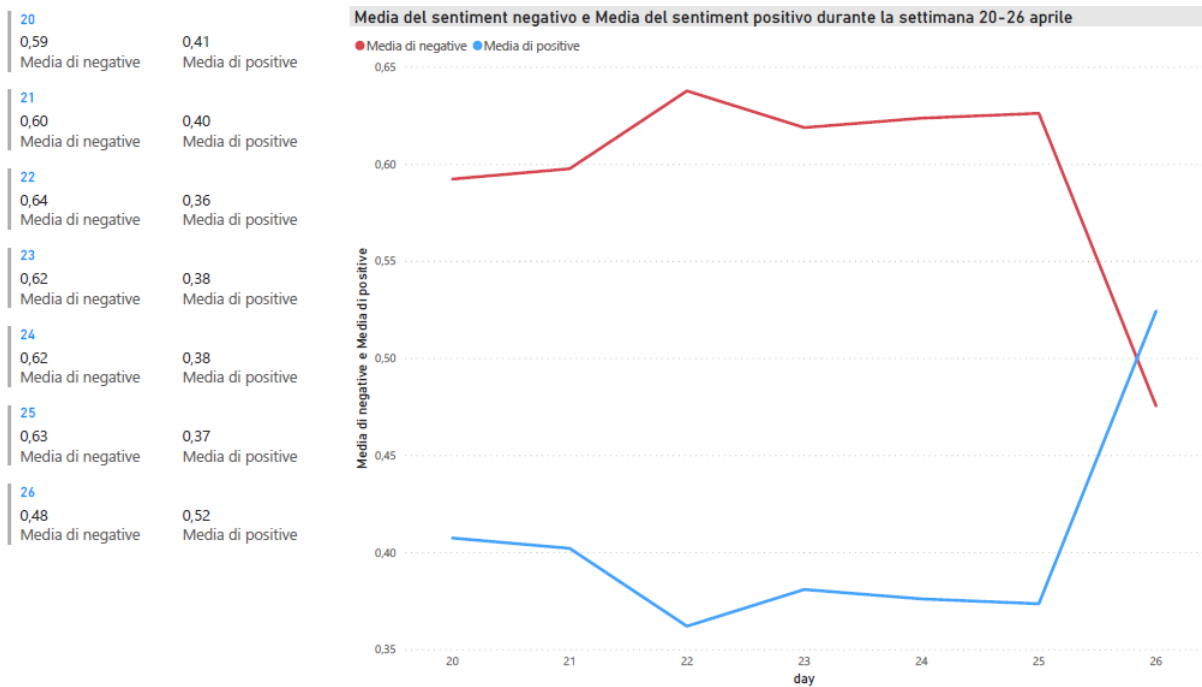


Figura 4.3: Sentiment medio positivo e negativo tra il 20 e il 26 aprile

In questa pagina sono mostrati i valori medi del sentiment lungo la settimana che va dal 20 al 26 aprile. Dal grafico è evidente che il sentiment negativo è sempre superiore a quello positivo, tranne che per il 26 aprile, dove sono comunque quasi a pari punteggio.

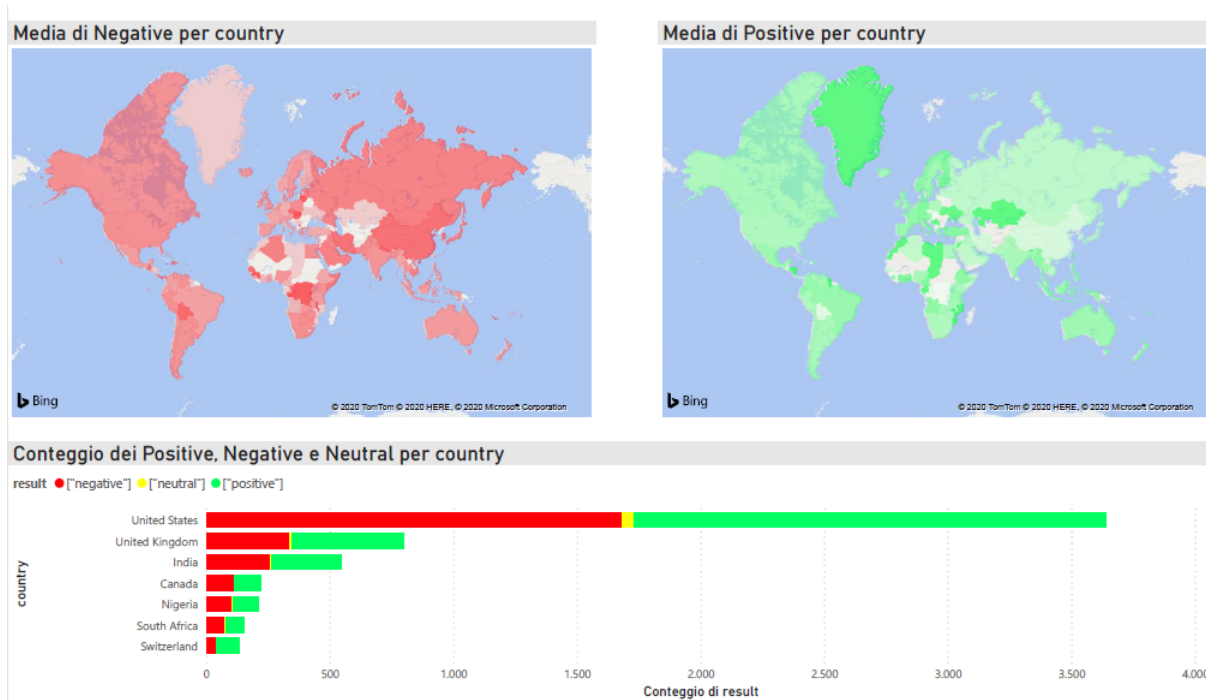


Figura 4.4: Hashtags più utilizzati e wordcloud

Infine, qui vi sono rappresentate due mappe del mondo:

- la mappa del sentiment medio negativo per Paese
- la mappa del sentiment medio positivo per Paese

Bisogna ricordare che il modello addestrato utilizzato per il sentiment riconosce solo la lingua inglese: per questo, da come è possibile vedere in basso, la maggioranza dei tweet localizzati proviene dagli Stati Uniti, dalla Gran Bretagna, dall'India e dal Canada.

Nonostante ciò, paesi come la Cina, la Russia e l'Iran presentano una media di sentiment negativo molto alta (la Cina dell'82 per cento), visibile grazie alla maggiore intensità di colore nella mappa a sinistra.

Per quanto riguarda l'Italia, probabilmente la media dei positive è al 65 per cento per via delle crescenti notizie sulla fase due che sarebbe accaduta di lì a poco, agli inizi di Maggio.