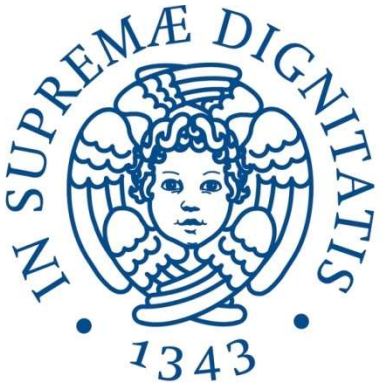

UNIVERSITÀ DI PISA

SCUOLA DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Triennale in Ingegneria Informatica

Tesi di Laurea

VALUTAZIONE DI METRICHE PER L'ANALISI DELLA POLARIZZAZIONE SU TWITTER

Candidato
Stefano Agresti

Relatori
Prof. Marco Avvenuti
Dott. Stefano Cresci
Dott. Leonardo Nizzoli

Anno accademico 2017/2018

1. SOMMARIO

2. PREFAZIONE

L'avvento delle nuove tecnologie ha portato l'umanità ad assistere, nel corso di appena qualche decennio, a drastici e improvvisi cambiamenti nella vita di tutti i giorni. Uno degli ambiti che più si è modificato, particolarmente negli ultimi 15 anni, è quello delle relazioni interpersonali, con la creazione di nuovi e potenti strumenti per permettere alle persone di comunicare fra loro.

Ciò è stato possibile grazie alla nascita di numerosi e variegati social network (come *Facebook*, *Twitter*, *Instagram* …), il cui travolgente successo ha coinvolto pressoché ogni angolo del globo, con più di 2 miliardi di persone che ogni giorno si connettono a Internet per usare questi strumenti.

Cambiamenti di questa portata, tuttavia, non possono portare solo benefici. *Fake news*, dipendenza, cyberbullismo e nuove tipologie di crimini sono solo alcuni dei problemi che i gestori di queste piattaforme, così come i nostri governi, si trovano ad affrontare.

Il lavoro del gruppo del *CNR*, presso cui ho svolto la mia tesi, punta a studiare i rischi di un ulteriore preoccupante fenomeno riguardante i social network, ovvero la sempre maggiore polarizzazione che gli utenti tendono ad assumere nel discutere temi delicati come etica o politica. Nell'ambito di questo più ampio progetto, si è rivelato necessario individuare delle metriche con cui valutare, da questo punto di vista, lo stato di salute di questi ambienti virtuali, con particolare attenzione, nel nostro caso, agli utenti italiani.

L'obiettivo di questa tesi, che si concentra sul social network *Twitter*, è la scrittura di alcuni script con cui analizzare le capacità di *tweet* legati alle elezioni politiche del 4 marzo 2018 di attraversare comunità differenti da quelle da cui sono partiti.

3. METRICHE UTILIZZATE

3.1 METRICHE E INDICI DI DISTRIBUZIONE

Gli aspetti della diffusione di un *tweet* che sono stati presi in considerazione sono i seguenti:

- Diversità fra gli utenti che hanno condiviso un certo *tweet* rispetto al suo autore. Gli utenti sono stati prima valutati in un unico gruppo, poi sono stati divisi per "livello di *retweet*". Per chiarire il concetto di "livello" possiamo fare un esempio: se A è l'autore di un *tweet*, tutti coloro che sono *follower* di A e lo retwittano vengono inseriti nel livello 1; chi retwitta uno di questi utenti viene inserito nel livello 2 e così via.
- Percentuale di *retweet*, per ogni livello, ottenuti dagli utenti compresi nel venticinquesimo e nel quinto percentile degli utenti più diversi dall'autore del *tweet* originale (ovvero il 25% e il 5% degli utenti più diversi).
- *Linear fit* della media delle diversità degli utenti divisi per livello.
- Rapporto fra numero di *follower* e di *following* dei *retweeters*.

Per ognuna di queste metriche (ad eccezione del *linear fit*) sono stati calcolati media, moda, mediano, deviazione standard, *skewness* e *kurtosis*.

Prima di proseguire oltre, è necessario specificare che la diversità fra due utenti viene valutata mediante il calcolo del coefficiente di Jaccard fra gli insiemi degli account che i due utenti seguono (i loro *following*). Tale coefficiente viene calcolato, fra due insiemi A e B, come indicato nella seguente formula:

$$Coefficiente\ di\ Jaccard = \frac{Cardinalità(A \cap B)}{Cardinalità(A \cup B)}$$

3.2 RISULTATI ATTESI

Quello che ci aspettiamo di osservare è un aumento nella diversità degli utenti per le cascate più "lunghe" (ovvero quelle con più livelli). In particolare, gli utenti dovrebbero essere tanto più diversi dall'autore di un *tweet* quanto più ci si allontana da questo (quindi quanto più alto è il livello in cui si trovano).

Ci aspettiamo inoltre che gli utenti più diversi prendano meno *retweet* rispetto agli altri utenti dello stesso livello.

Infine, valutare il rapporto fra *follower* e *following* dei *retweeters* può essere utile per individuare la presenza di "*influencer*" che possano aver influito sulla diffusione del *tweet*.

4. DATASET E TECNOLOGIE UTILIZZATE

4.1 DATASET

Al fine di poter applicare gli script realizzati su dati reali, il gruppo del *CNR* presso cui ho svolto la tesi si è occupato di costruire un database popolato con informazioni riguardanti all'incirca 100 *tweet* correlati alle elezioni politiche italiane del 4 marzo 2018 e costituito da cinque tabelle.

4.1.1 TABELLE

TWEETS

Questa tabella contiene le informazioni più strettamente collegate ad ogni *tweet*, fra cui id del *tweet* stesso, id dell'autore, testo, data di scrittura, numero di *retweet* e di risposte, nonché una serie di altri dati non utilizzati per gli scopi di questa tesi. Essa contiene poco più di 100000 record[(1)].

USERS

Questa tabella contiene le informazioni riguardanti tutti gli utenti che sono stati coinvolti nella diffusione dei *tweet* archiviati nella tabella precedente. Oltre a id dell'utente, nome e nickname, vengono salvati anche numero di *follower* e di *following*. Anche in questo caso, sono presenti altri dati che non sono stati utilizzati per questa tesi. Gli utenti coinvolti sono stati 65520.

RETWEETS_TREE

Questa tabella viene utilizzata per legare un *retweet* al corrispondente *tweet* originale. In particolare, per ogni riga possiamo trovare id del *retweet*, id del *tweet* e gli id degli utenti che li hanno realizzati. Le sue dimensioni sono più o meno le stesse della tabella TWEETS.

LINKS

Questa tabella mantiene le informazioni sui *following* degli utenti presenti nella tabella USERS. Concettualmente è la più semplice delle quattro, in quanto ogni riga mantiene solo l'id del *follower* e dell'utente seguito. Tuttavia, essa è anche di gran lunga la più grande con oltre 91 milioni di record.

METRICS

Quest'ultima tabella, costituita da 100 record (uno per ogni *tweet* analizzato), è inizialmente vuota ed è il posto in cui verranno salvati i risultati definiti nel paragrafo precedente.

4.2 TECNOLOGIE USATE

4.2.1 LINGUAGGIO DI PROGRAMMAZIONE E AMBIENTE DI SVILUPPO

Tutti gli script riguardanti questa tesi sono stati realizzati utilizzando il linguaggio *Python* nella versione 3.6 [1]. L'ambiente di sviluppo che ho scelto di utilizzare è *Spyder* [2] nella versione fornita dal software *Anaconda Navigator* [3]. I risultati che verranno presentati più avanti sono stati ottenuti eseguendo gli script su un computer con processore *Intel* i5-6600K da 3.5GHz.

4.2.2 LIBRERIE UTILIZZATE

Di seguito sono riportate le principali librerie utilizzate durante lo svolgimento della tesi.

(1) Va ricordato che i record di questa tabella non sono solo *tweet*, ma anche risposte e *retweet*, fatto che ne giustifica la notevole dimensione rispetto al numero di *tweet* effettivamente analizzati.

CSV

Vista la grande quantità di dati, i dataset sono stati analizzati nel formato CSV[2]. Per potervi accedere è stata quindi utilizzata l'omonima libreria [4].

PANDAS

Questa libreria è risultata utile perché permette di effettuare ricerche sui dataset con uno stile simile a quello utilizzato in SQL, evitando così sprechi di tempo per la realizzazione di funzioni aventi lo stesso scopo finale. Inoltre, *Pandas* mette a disposizione una serie di funzioni per calcolare direttamente media, moda, mediano, deviazione standard, skewness e kurtosis su un *DataFrame*[3] [5].

NUMPY

Libreria molto potente e popolare di *Python*, è stata utilizzata nel calcolo dei percentili (realizzabile con una semplice chiamata di funzione) e per disegnare i grafici relativi al *linear fit* (in particolare quelli in cui è stata usata l'interpolazione) [6].

SCIPY

Di questa libreria è stata utilizzata la funzione *lingress* per il calcolo degli indici relativi al *linear fit* [7].

MATPLOTLIB

Come si può dedurre dal nome, questa è la libreria utilizzata per realizzare i grafici che verranno presentati più avanti [8].

[2] Ho inizialmente provato a trasferire i dati in un database SQL, ma il tentativo si è rivelato inconcludente a causa dell'eccessiva quantità di tempo richiesta (la mia stima, basata sul tempo che vi ho dedicato prima di interromperlo, era di circa 180 giorni solo per ricostruire la tabella LINKS).

[3] Un *DataFrame* è un oggetto della libreria *Pandas* rappresentante una tabella. Esiste la possibilità, di cui mi sono avvalso, di trasformare una lista in un oggetto di questo tipo, così da poterne sfruttare le potenzialità.

5. PROBLEMATICHE E SOLUZIONI ADOTTATE

5.1 RACCOLTA DEI *FOLLOWING* DI UN UTENTE

Uno dei primi problemi che ho dovuto affrontare è stato rendere il più veloce possibile la raccolta degli account seguiti da un utente. Per eseguire questa operazione, infatti, è necessario accedere alla tabella LINKS, che, come già accennato in precedenza, è di gran lunga la più grande fra quelle del dataset.

Considerando che, solo per scorrerla tutta, sono richiesti all'incirca tre minuti, e considerando che questa operazione deve essere eseguita per ciascuno degli utenti che ha condiviso uno dei *tweet* presenti nel database, è evidente che una ricerca lineare è, in questa circostanza, inutilizzabile.

La soluzione adottata è stata l'indicizzazione della tabella su due livelli. L'idea di base è stata realizzare le due liste seguenti:

- Una lista contenente coppie del tipo {*user_id*, *indice*}
- Una formata a sua volta da liste contenenti gli id dei *following*, divisi in base all'utente *follower*

Ogni indice della prima lista punta all'entrata corrispondente della seconda.

Le liste vengono realizzate dalla classe *FollowingHasher* nel momento in cui viene istanziato un oggetto di questo tipo. Tale classe si occupa anche di creare due file in cui salvare i risultati ottenuti, in modo tale da poter successivamente evitare di ricalcolare le liste da zero.

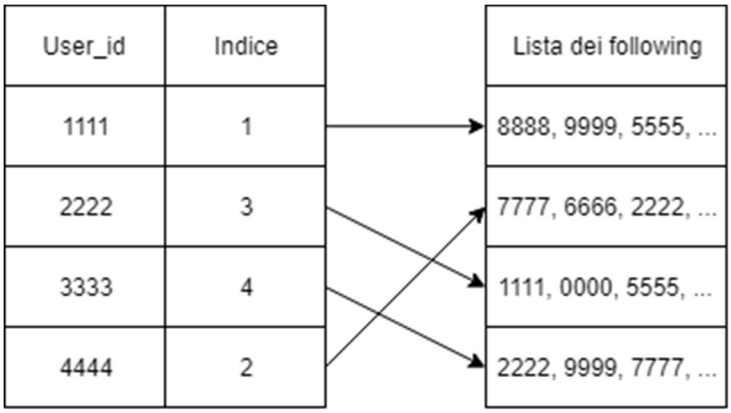

Figura 1: Visualizzazione grafica delle due liste dell'algoritmo

Per rendere il più efficiente possibile l'operazione di indicizzazione, quando si scorre la tabella LINKS, si crea una nuova coppia {*user_id*, *indice*} ogni volta che l'id dell'utente *follower* cambia (quindi non si ricerca se per quel particolare utente era già presente una entry). In questo modo la complessità dell'algoritmo risulta *O(n)* con *n* numero di record di LINKS, poiché si evita di dover scorrere la prima lista inutilmente. Naturalmente è necessario riunire alla fine le informazioni riguardanti uno stesso utente.

Alla fine, dopo aver calcolato le liste, è possibile ottenere la lista dei *following* di un utente con complessità *O(1)*, rendendo tale operazione praticamente istantanea[4].

5.2 CALCOLO DEL COEFFICIENTE DI JACCARD

Una volta trovato il modo di raccogliere i following relativi ad una coppia di utenti, è stato necessario elaborare una funzione per calcolare l'indice di Jaccard sui due insiemi[5].

[4] In realtà la complessità totale sarebbe *O(logn)* con *n* numero di utenti *follower* perché bisogna cercare la coppia {*user_id, indice*} prima di poter effettivamente recuperare la lista dei *following*.
[5] Si ricorda che l'indice di Jaccard viene calcolato tramite il rapporto fra le cardinalità dell'intersezione e dell'unione di due insiemi.

Un primo metodo per calcolarlo è tramite un semplice algoritmo lineare operante su liste. Tuttavia, questa strategia, per quanto di veloce implementazione, non risulta particolarmente efficiente in *Python*, poiché quest'ultimo non lavora bene con le liste.

Per questa ragione, si è adottato un diverso approccio basato sull'utilizzo dei *set*. Questo strumento, messo a disposizione da *Python*, consente di effettuare in maniera molto efficiente le operazioni tra insiemi come intersezione e unione.

Con questo nuovo algoritmo il calcolo del coefficiente di Jaccard risulta notevolmente più veloce [9][10].

```
21  #Calcola Jaccard fra i due array 'first' e 'second'
22  def get_Jaccard(first, second):
23
24      #Si usa set() per rendere il calcolo più efficiente rispetto all'utilizzo di liste
25      return len(set(first) & (set(second)))/(len(first) + len(second) - len(set(first) & (set(second)))) if first is not None and second is not None else None
```

Figura 2: Codice della funzione che implementa il calcolo del coefficiente di Jaccard

5.3 INDICI DI DISTRIBUZIONE

Come detto in precedenza, per tutte le metriche prese in considerazione sono stati calcolati degli indici di distribuzione per valutarne adeguatamente l'andamento per ogni cascata. Gli indici utilizzati sono:

- Media
- Moda
- Mediano
- Deviazione standard
- *Skewness*
- *Kurtosis*

Per effettuarne un calcolo efficiente si è sfruttata la libreria *Pandas*, la quale mette a disposizione una funzione per ognuno di essi.

La procedura per utilizzare tali funzioni è mostrata nella figura 3.

```
14  pd_level = pd.DataFrame(level)
15
16  mean     = float(pd_level.mean())
17  mode     = float(pd_level.mode().iloc[0])
18  median   = float(pd_level.median())
19  std      = float(pd_level.std())
20  skewness = float(pd_level.skew())
21  kurtosis = float(pd_level.kurt())
```

Figura 3: Codice con cui vengono calcolati gli indici di distribuzione

5.4 INDIVIDUAZIONE DEI LIVELLI DI *RETWEET*

Una delle funzionalità più utilizzate per il calcolo delle varie metriche è la separazione dei *retweeters* di una cascata in più livelli, a seconda dell'utente di cui hanno condiviso il *tweet* (essi possono infatti aver retwittato l'utente iniziale o un altro dei suoi *retweeters*).

Ad occuparsi di questo compito è la funzione *get_retweets_per_level* contenuta in *RetweetersHandler.py*, file che può essere importato dai vari script. La funzione in questione effettua una ricerca ricorsiva all'interno della tabella RETWEETS_TREE, tramite la quale individua, partendo da un livello iniziale costituito dal solo autore del *tweet*, coloro che hanno retwittato gli utenti del livello precedente, continuando fino a terminare tutti i *retweeters*.

Particolarmente insidioso è stato in questo caso la presenza di alcuni utenti *retweeter* di se stessi: tale possibilità, resa disponibile da *Twitter* e sfruttata per mettere in evidenza i propri *tweet*, rappresenta un problema per questo algoritmo perché in grado di mandare in loop la ricerca ricorsiva. Per questa ragione ad ogni passaggio viene effettuato un controllo sui livelli precedentemente calcolati, così da assicurarsi di non aver selezionato nuovamente utenti già presi in considerazione.

Infine, un'ultima attenzione è stata dedicata ai *retweeters* "sparsi", ovvero coloro che hanno retwittato un utente non perché suoi *follower*, ma perché hanno visto casualmente il suo *tweet* (come può capitare se quest'ultimo finisce fra i *trending topics*). Per individuare questi utenti è necessaria un'ulteriore ricerca, dato che in questo caso all'interno della tabella RETWEETS_TREE è presente il valore NULL al posto dell'id dell'utente retwittato.

```
#Ritorna una lista contenente gli utenti che hanno retwittato tweet_id, divisi per livelli
def get_retweets_per_level(tweet_id):

    tweet_id = int(tweet_id) #Serve per fare le query con pandas
    retweets_tree = pd.read_csv(PATH_FILE_RETWEET_TREE)
    levels = [[get_author(tweet_id)]] #Il livello 0 è l'utente iniziale

    #Si fa un ciclo sulla lista dei retweet, scendendo man mano di livello
    while True:

        #Questa query cerca i retweet relativi a tweet_id in cui l'utente retwittato è fra quelli dell'ultimo livello completato
        level = list(retweets_tree.loc[(retweets_tree["retweeted_user_id"].isin(levels[len(levels) - 1])) & (retweets_tree["original_tweet_id"] == tweet_id)]["retweeter_user_id"])

        #Per evitare loop, ci assicuriamo che un utente non abbia retwittato due volte lo stesso tweet (nel qual caso sarebbe già in un livello precedente)
        for row in levels:
            for item in level:
                if item in row:
                    level.remove(item)

        #Se non abbiamo trovato nuovi retweet, abbiamo terminato la cascata
        if not level:
            break
        else:
            levels.append(level)

    #Aggiungiamo un ultimo livello che racchiuderà gli utenti sparsi
    #NOTA: Questo livello sarà sempre presente, al più sarà vuoto
    levels.append(get_retweet_sparsi(tweet_id, retweets_tree))

    return levels
```

Figura 4: Codice della funzione "get_retweets_per_level"

5.5 SCRITTURA DEI RISULTATI

Tutte le metriche e tutti gli indici presentati finora vengono calcolati da cinque diversi script *Python*, ognuno dei quali salva i risultati ottenuti in un proprio file *csv*. In questo modo è stato possibile separare in maniera logica i vari aspetti della tesi, evitando di dover calcolare indici fra loro indipendenti in uno stesso file. Tuttavia, si è resa necessaria la scrittura di un ulteriore script per poter raccogliere tutti i valori in un'unica posizione, così da poter accedere a tutte le informazioni contemporaneamente.

Ad adempiere a questo scopo è *ResultsWriter.py*, il quale semplicemente ricopia in una cartella finale tutti i dati contenuti nelle cinque tabelle realizzate precedentemente. Tale cartella comprende tre diversi file in formato *csv*, uno contenente i risultati relativi alle cascate nel loro complesso, uno contenente i risultati relativi ai livelli delle varie cascate e uno dedicato ai risultati relativi alle metriche sui percentili.

6. RISULTATI

Per interpretare efficacemente i dati restituiti dall'esecuzione dei vari script si è reso necessario l'utilizzo di grafici, al fine di averne una rappresentazione intuitiva. Di seguito sono riportati quelli più significativi, insieme ad alcune considerazioni.

6.1 RISULTATI DIVISI PER CASCATA

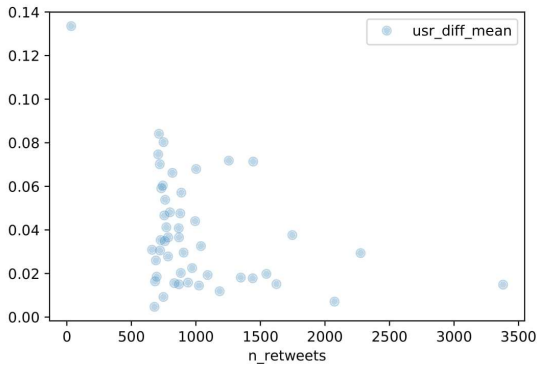

Figura 5: Media del coefficiente di Jaccard in relazione al numero di retweet di una cascata

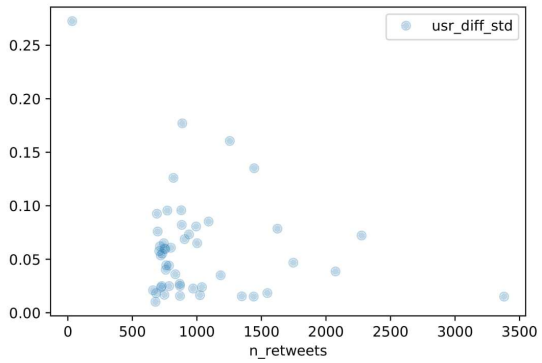

Figura 6: Deviazione standard della distribuzione dei coefficienti di Jaccard in relazione al numero di retweet di una cascata

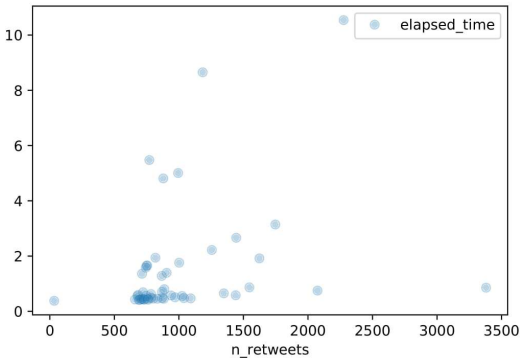

Figura 7: Tempo impiegato (secondi) per calcolare il coefficiente di Jaccard per tutti i retweeters di una cascata

I grafici riportati nelle figure 5, 6 e 7 sono relativi alla distribuzione del coefficiente di Jaccard calcolato fra l'autore di un *tweet* e i suoi *retweeters*. Tale distribuzione è presentata in funzione del numero di *retweet* della cascata associata a tale *tweet*.

Il primo grafico riporta la media di questo valore per ogni *tweet* del dataset. È possibile notare che non è presente un particolare andamento e che i risultati tendono ad essere piuttosto variegati anche per cascate con numeri simili di *retweet*. Non si possono perciò fare ipotesi valide sul comportamento di questo indice all'aumentare del numero di *retweeters*.

Nel secondo grafico è possibile osservare il variare della deviazione standard, sempre in funzione del numero di *retweet* della cascata. In questo caso è interessante notare che tale indice tende a diminuire se la cascata aumenta di dimensioni, mentre sarebbe lecito aspettarsi il contrario, poiché, essendo di più gli utenti su cui Jaccard viene calcolato, dovrebbero essere più variegati i risultati. Una possibile spiegazione è che in questi casi diventa predominante il numero di utenti con coefficiente di Jaccard prossimo allo zero (perché molto diversi dall'utente iniziale), ragion per cui la deviazione standard tenderebbe a ridursi.

Infine, nell'ultimo grafico si può osservare un'informazione più tecnica, ovvero il tempo utilizzato per calcolare tutti gli indici relativi alla singola cascata. Come si può osservare, raramente supera i due secondi per cascata, senza mai andare comunque oltre i dieci. Inoltre, si può notare che queste tempistiche appaiono indipendenti dalle dimensioni delle cascate, probabilmente perché più legate al numero di *following* dei vari *retweeters*.

6.2 RISULTATI DIVISI PER LIVELLO

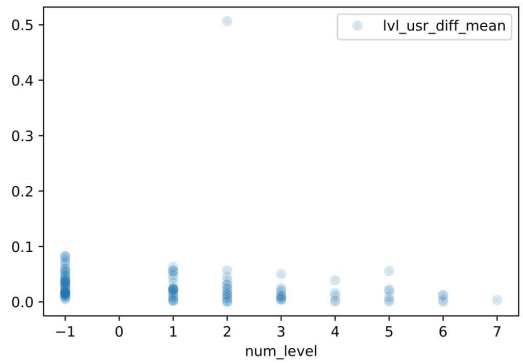

Figura 8: Media del coefficiente di Jaccard per "livello di retweet"

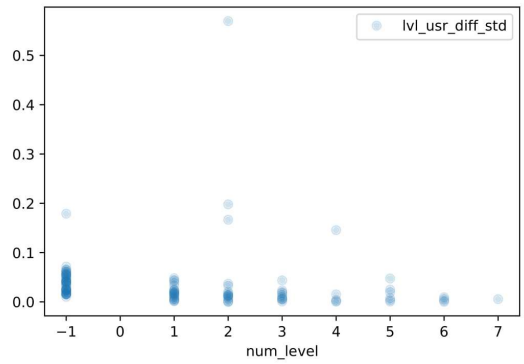

Figura 9: Deviazione standard della distribuzione dei coefficienti di Jaccard divisi in base al "livello di retweet"

In questi grafici viene presentato l'andamento del coefficiente di Jaccard quando calcolato separando i *retweeters* per livello. Sulle ascisse è quindi riportato il numero di livello a cui i valori sono associati, con -1 a rappresentare i *retweeters* "sparsi".

Nel primo grafico si può osservare la distribuzione delle medie nei vari livelli. Come si vedrà meglio più avanti con il *linear fit*, si può già notare una diminuzione di tale indice all'aumentare del livello, implicando quindi una maggiore diversità degli utenti più lontani dalla sorgente, come in effetti ci si potrebbe aspettare intuitivamente.

Il secondo grafico presenta invece la variazione della deviazione standard all'aumentare del livello. Come già osservato nel paragrafo precedente, tale indice tende a diminuire all'aumentare della dimensione della cascata, probabilmente perché, come detto sopra, il coefficiente di Jaccard tende ad assumere valori sempre più prossimi allo zero quando ci si allontana dalla sorgente.

6.3 *LINEAR FIT*

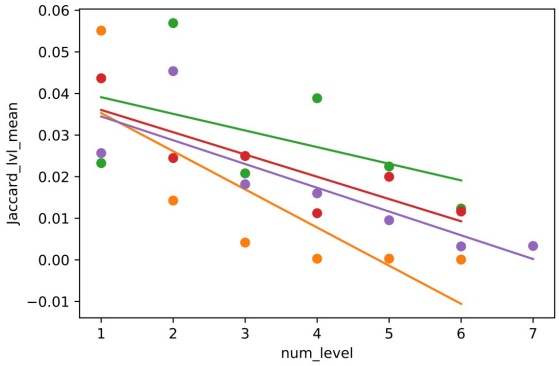

Figura 10: Rappresentazione grafica dei risultati del linear fit sulle quattro cascate più "lunghe" del dataset

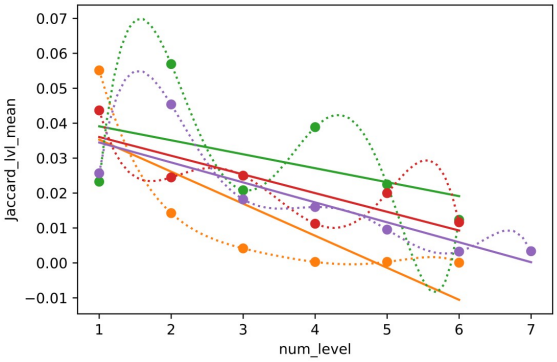

Figura 11: Stesso grafico della figura 10 con l'aggiunta delle funzioni ottenute per interpolazione dei valori analizzati

Questi grafici riportano una rappresentazione visuale dei risultati del *linear fit* eseguito sui valori delle medie degli indici di Jaccard calcolati dividendo gli utenti per livello. Per ottenere dei risultati più significativi sono state considerate solo le quattro cascate del dataset con più di cinque livelli, escludendo le altre.

Il risultato più importante è, in questo caso, la conferma di una delle ipotesi fatte all'inizio della tesi, ovvero che, effettivamente, più gli utenti si allontanano dall'autore di un *tweet*, più tendono ad essere diversi da quest'ultimo. È interessante notare, tuttavia, che esistono comunque variazioni significative fra i vari livelli, sebbene la tendenza generale sembri essere quella appena sottolineata.

6.4 ANALISI DEL QUINTO E VENTICINQUESIMO PERCENTILE

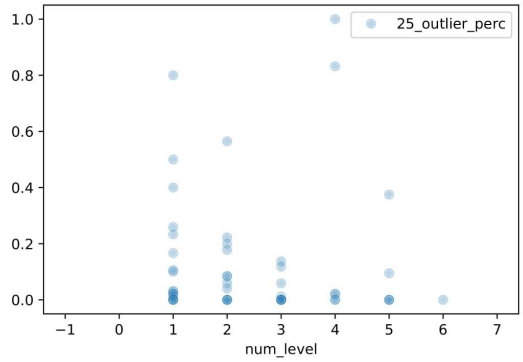

Figura 12: Percentuale di retweet ottenuti dagli utenti al di sotto del 25esimo percentile in termini di diversità dall'autore del tweet condiviso

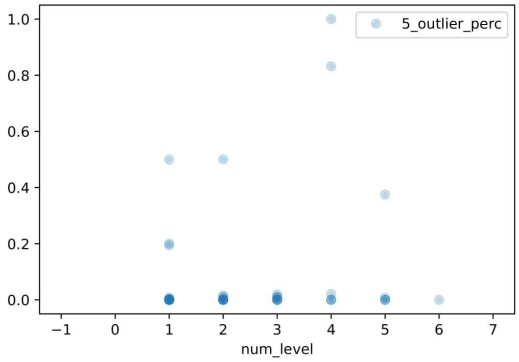

Figura 13: Stesso grafico della figura 12, valutato però rispetto al quinto percentile

In queste due immagini sono raffigurati i grafici che riportano la percentuale di *retweet* ottenuti, per ogni livello, dagli utenti più diversi dall'autore del *tweet* iniziale. Nello specifico, il primo grafico fa riferimento agli utenti il cui coefficiente di Jaccard è sotto il venticinquesimo percentile, mentre il secondo considera quelli al di sotto del quinto.

Come nel caso precedente, anche qui viene confermata una delle ipotesi iniziali, ovvero l'ipotesi secondo cui gli utenti più diversi tendono a ricevere meno *retweet* rispetto agli altri. È possibile infatti osservare che la percentuale di *retweet* degli utenti sotto il venticinquesimo percentile è ben al di sotto del 25% del totale (come dovrebbe essere se i *retweet* fossero distribuiti equamente). Lo stesso discorso vale anche nel caso del quinto percentile, dove addirittura si può notare come in molti casi il numero di *retweet* ottenuti sia prossimo allo zero.

6.5 RAPPORTO *FOLLOWER – FOLLOWING* DEI *RETWEETERS*

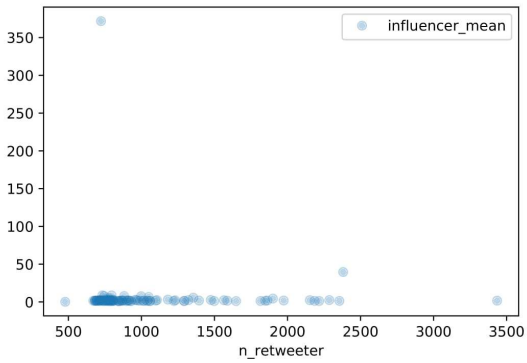

Figura 14: Media del rapporto follower - following degli utenti che hanno condiviso un tweet, mostrato in relazione al numero di questi ultimi

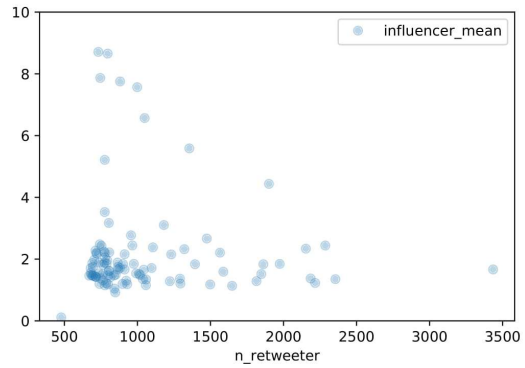

Figura 15: Ingrandimento del grafico precedente

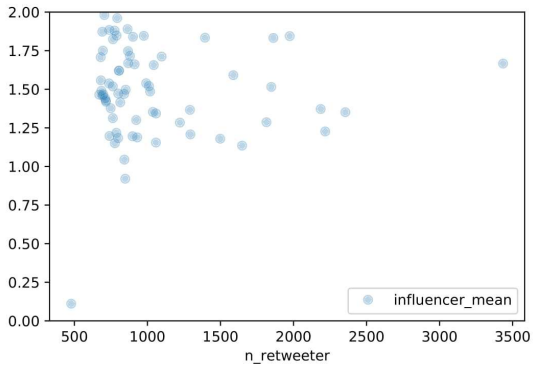

Figura 16: Ulteriore ingrandimento del grafico in figura 14

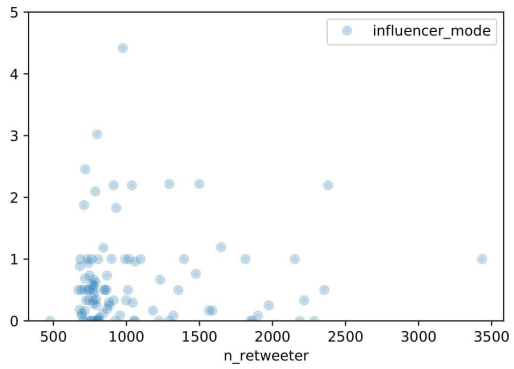

Figura 17: Moda del rapporto follower - following degli utenti che hanno condiviso un tweet in relazione al loro numero

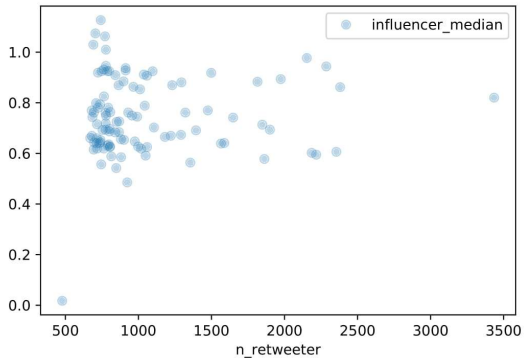

Figura 18: Mediano del rapporto follower - following degli utenti che hanno condiviso un tweet in relazione al loro numero

In questo paragrafo vengono presentati i grafici relativi alla distribuzione del rapporto *follower – following* dei *retweeters* di ogni cascata, al variare della lunghezza di quest'ultima.

Una prima informazione, facilmente osservabile, consiste nel fatto che spesso la media di questi rapporti può assumere valori molto elevati, chiaramente a causa della presenza di *influencer*, utenti per i quali il numero di *follower* è nettamente superiore rispetto a quello dei *following*. È inoltre interessante osservare che una delle cascate che spicca in questo senso è anche una di quelle con maggior numero di *retweets*.

Un'altra informazione interessante può essere ricavata dal confronto fra media, moda e mediano di questa distribuzione. Infatti, mentre la prima si aggira per lo più fra 1 e 2, negli altri due casi tende ad essere sempre al di sotto dell'unità, segno che probabilmente esistono pochi valori molto elevati che alzano il valore medio. Questo comportamento suggerisce che tutte le cascate con un numero significativo di *retweet*[(6)], anche se composte per la maggior parte da utenti "normali", presentano un certo numero di utenti più popolari, che probabilmente hanno contribuito alla loro diffusione.

Allo stesso tempo, tuttavia, bisogna sottolineare che il *tweet* più popolare del dataset utilizzato (ovvero quello che è stato condiviso circa 3500 volte), pur confermando questo trend (il suo rapporto medio è comunque superiore all'unità), non spicca all'interno di questi grafici. Da questa osservazione è possibile dedurre che un *tweet* può diffondersi in maniera anche molto significativa pur non ricevendo l'apporto di grandi *influencer*.

[(6)] Si ricorda che tutti i *tweet* analizzati hanno ricevuto almeno 500 condivisioni, numero generalmente piuttosto elevato rispetto alla normalità.

7. CONSIDERAZIONI FINALI

7.1 POSSIBILI FUTURE AREE DI STUDIO

A partire dai risultati ottenuti, è possibile formulare alcune idee su come espandere questo lavoro nel futuro, in modo da poter ottenere informazioni più dettagliate e significative.

Come prima cosa, si dovrebbero confermare i trend osservati utilizzando un dataset di dimensioni più estese. In questo senso, particolare attenzione dovrebbe essere prestata al comportamento del *linear fit*, su cui sono state formulate ipotesi basandosi su appena quattro cascate. È necessario quindi controllare che, anche con un maggior numero di cascate, la pendenza delle rette ottenute continui ad essere negativa, così da dimostrare che, effettivamente, a livelli più alti sono associati utenti più diversi.

Inoltre, potrebbe essere interessante valutare l'andamento del coefficiente di Jaccard con un maggior numero di *tweet*. Nello specifico, sarebbe utile controllare se, ad un maggior numero di *retweet*, la media del coefficiente tende a diminuire (nel paragrafo precedente non si è potuta fare questa analisi poiché la maggior parte delle cascate considerate presenta un numero simile di condivisioni).

Infine, potrebbe essere utile approfondire il discorso sugli *influencer*, ad esempio ripetendo su di essi il lavoro effettuato con i percentili (andando quindi ad analizzare la percentuale di *retweet* ottenuti da questi utenti rispetto al totale). In questo modo sarebbe possibile quantificare meglio il loro contributo alla diffusione di un *tweet*.

7.2 CASI PARTICOLARI

Come ultimo aspetto della tesi, ho ritenuto utile cercare due dei *tweet* che, dalle analisi effettuate, sono risultati più interessanti.

Il primo è il *tweet* che ha ricevuto più condivisioni all'interno del dataset (al momento della creazione di quest'ultimo aveva ottenuto poco meno di 3500 *retweet*, mentre al momento della scrittura della tesi ha superato i 4400). Il *tweet*, scritto il giorno stesso delle elezioni politiche e di natura provocatoria, è stato realizzato da un utente comune (circa 300 *follower* e 500 *following*), ragion per cui è possibile dedurre che si sia diffuso principalmente per il significato del suo messaggio, piuttosto che per altri motivi (è interessante notare che, come già accennato precedentemente, questo *tweet* non spicca nemmeno per la presenza di *influencer* fra i suoi *retweeters*) [11].

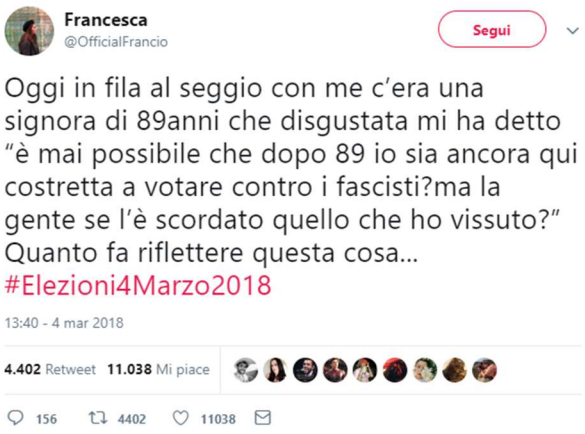

Figura 19: Tweet del dataset che ha ricevuto più condivisioni

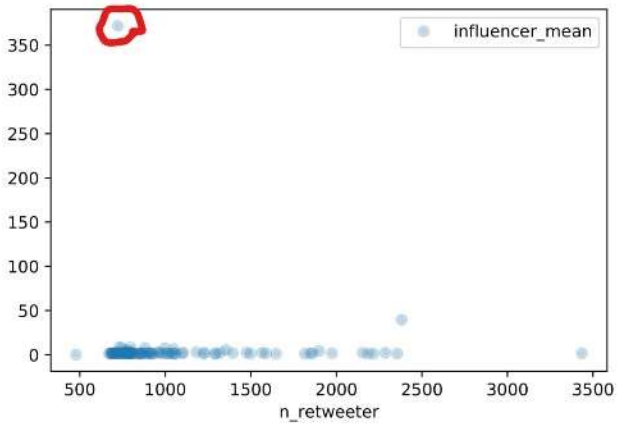

Figura 20: Al valore cerchiato in rosso corrisponde il tweet mostrato in figura 21

Figura 21: Tweet della BBC notato per l'elevato valore del rapporto medio fra follower e following dei suoi retweeters

Il secondo *tweet* analizzato è quello che più si differenzia dal punto di vista della media del rapporto *follower – following*, che, come può essere facilmente notato in figura 20, risulta abbondantemente superiore a 300. Il *tweet* in questione, riportato nella figura 21, è stato scritto dall'account della *BBC Breaking News* e in seguito retwittato dall'account della *BBC World*. Entrambi i profili possono vantare decine di milioni di *follower* a fronte di un numero irrisorio di *following*, motivo per cui tendono a differenziarsi notevolmente per quanto concerne la metrica del rapporto medio *follower - following* [12].

8. RIFERIMENTI BIBLIOGRAFICI

[1] *Python* 3.6: https://www.python.org/doc/

[2] *Spyder documentation*: https://pythonhosted.org/spyder/

[3] *Anaconda Navigator*: https://anaconda.org/anaconda/anaconda-navigator

[4] *CSV File Reading and Writing*: https://docs.python.org/2/library/csv.html

[5] *Python Data Analysis Library*: https://pandas.pydata.org/

[6] *NumPy*: http://www.numpy.org/

[7] *SciPy*: https://www.scipy.org/

[8] *Matplotlib*: https://matplotlib.org/

[9] Michele Mattioni, *List intersection in python: let's do it quickly*: http://blog.michelemattioni.me/2015/01/10/list-intersection-in-python-lets-do-it-quickly/

[10] *WikiPython*: https://wiki.python.org/moin/TimeComplexity?

[11] *Twitter Inc*: https://t.co/TxxTe2WsT2

[12] *Twitter Inc*: https://twitter.com/bbcbreaking/status/959762406728785920