UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Psicologia dello Sviluppo e della Socializzazione

Corso di Laurea in Scienze e Tecniche Psicologiche

Elaborato finale

Definizione di indicatori oggettivi attraverso il text mining: un'applicazione sui profili Instagram

The use of text mining to create objective indicators: An example based on Instagram

Relatore: Prof. Antonio Calcagnì

Laureanda: Cappon Camilla

Matricola n° 1139017

Anno Accademico 2018/2019

INDICE

SOMMARIO

Questo lavoro è improntato all'estrazione di informazioni relative a come le studentesse universitarie presentano se stesse attraverso il social media Instagram, prestando attenzione a tre dimensioni in particolare: con quanta frequenza viene utilizzato il media, come viene utilizzato per costruire la propria immagine, promuovendo se stessi attraverso una pluralità di contenuti parziali, e un primo indicatore di popolarità derivante dal rapporto tra follower e following, rispettivamente pagine e persone che seguono e seguiti. Le cartelle zip contenenti i dati relativo al proprio account Instagram vengono fatte scaricare alle utenti, alle quali vengono richiesti soltanto alcuni dei file JSON presenti. La procedura parte da un primo script in Python, il quale effettua un iniziale parsing dopo essere stato avviato all'interno delle cartelle contenenti i JSON. I file, così modificati, vengono poi analizzati utilizzando il software Rstudio e alcuni suoi pacchetti.

Nel *primo capitolo* vengono brevemente esposti il contesto di ricerca e la costruzione degli indicatori oggetti, fornendo una cornice entro cui collocare il lavoro.

Nel *secondo capitolo* viene enunciata la metodologia impiegata, analizzando l'intero codice scritto in Rstudio. I tre paragrafi principali prendono in esame l'utilizzo del media con riferimento ai dati quantitativi, la self-presentation rispetto ai sintagmi più utilizzati e rilevanti, e la popolarità.

Nel *terzo capitolo* il codice viene applicato ad un caso studio, eseguendo il text mining sui dizionari di hashtag e didascalie ricavati, generando in seguito grafici wordcloud.

CAPITOLO UNO: INTRODUZIONE

1.1 Social media e data mining

La diffusione dei social media, è ormai innegabile, è penetrata all'interno della società e si è consolidata nell'uso quotidiano attraverso la creazione di pratiche comuni, condivise (Riva, 2014). Queste piattaforme vengono frequentate da milioni di utenti, che a propria volta generano un'enorme vastità di contenuti di vario tipo, fruibili a chiunque. Estrarre, categorizzare ed ordinare le informazioni da questi dati permette di dare senso ad un'innumerevole quantità di testo e di ricavare utile conoscenza per molteplici applicazioni (Marmo, 2016). L'analisi dei Big Data, definiti dalla compagnia pubblica Teradata come un sistema che eccede, sorpassa e supera i sistemi di hardware e software comunemente usati per catturare, gestire ed elaborare i dati in un lasso di tempo ragionevole, è un dominio tradizionalmente associato ad ambiti quali l'informatica, la statistica ed il business, ma ultimamente e in modo sempre più rapido, sta penetrando anche nella ricerca psicologia e sociologica. Secondo Harlow e Oswald (2016) è chiaro e lampante come quest'ambito, ampio e in crescita, offra un'opportunità unica per gli psicologi interessati ad affrontare le complesse sfide tecniche, sostanziali ed etiche per quanto riguarda l'archiviazione, il recupero, l'analisi e la verifica di set di dati di grandi dimensioni. Diviene possibile sviluppare teorie ed ipotesi su elementi iniziali e verificare i risultati su altri set di convalida: con l'esistenza di grandi set di dati che spesso non hanno avuto una teoria dominante o un insieme di solide ipotesi che riguardano la loro formazione, un'analisi iniziale è spesso a livello esplorativo. Saranno poi necessari uno o più studi successivi per poter generalizzare i risultati, soprattutto se viene considerato un gran numero di variabili (Azzalini e Scarpa, 2012).

Questa tesi di laurea si articola a partire da un livello preliminare, a scopo esplorativo, e nasce con l'intenzione di abbozzare e inquadrare una ricerca basata sulla gestione, elaborazione e analisi (*mining*) di informazioni ricavate da Instagram.

1.2 Contesto di ricerca

Instagram è un social network rilasciato nell'ottobre 2010 dedicato all'elaborazione digitale e alla condivisione di foto. Negli ultimi anni ha incontrato il favore di molti giovani adulti, in una fascia d'età compresa tra i 18 e i 29 anni (Sensis, 2017), i quali, avendo l'opportunità di modificare a proprio piacimento le immagini prima di renderle pubbliche, regolano e controllano la propria *self-presentation* in ogni momento. Attraverso la possibilità di commentare le foto altrui, inoltre, diviene più semplice la comunicazione tra utenti, così come aumenta il confronto sociale, l'oggettivazione sessuale e l'auto-oggettivazione, intesa come la tendenza ad esternare la propria prospettiva e trattarsi alla stregua di un oggetto dal valore meramente fisico. La piattaforma di Instagram diviene così un ambiente idealizzato, intrinsecamente oggettivante, un luogo perfetto per analizzare la sessualizzazione e valutare l'impatto di immagini sessualizzate accompagnate da commenti che esprimono apprezzamenti sull'aspetto fisico delle donne.

L'effetto negativo che possono avere i giudizi scritti, sia a valenza negativa che positiva, sull'individuo può essere spiegato ipotizzando che il commento ricevuto attivi l'attenzione di chi lo riceve sul fatto che altri stanno valutando il suo aspetto, innescando il meccanismo dell'auto-oggettivazione e portandolo a concentrarsi in primo luogo sull'apparenza.

Uno studio condotto da Tiggemann e Barbato (2018) ha dimostrato un incremento dell'insoddisfazione corporea delle studentesse a cui sono state mostrate immagini di target femminili accompagnate da commenti relativi all'aspetto fisico rispetto al gruppo di controllo, il quale è stato esposto alle medesime immagini accompagnate da commenti relativi al paesaggio, spostando così l'attenzione su elementi differenti.

I giudizi positivi relativi all'aspetto fisico, oltre a focalizzare l'attenzione sul corpo intensificando l'auto oggettivazione, vengono apprezzati dagli autori delle immagini postate. Così, variabili come l'auto oggettivazione di stato e l'interiorizzazione delle norme di bellezza risultano poter moderare la relazione tra esposizione a Instagram (foto e commenti) ed effetti sul benessere psicofisico dell'utente della piattaforma.

1.3 Costruzione di indicatori oggettivi

Questo lavoro si concentra sull'estrazione di elementi pubblicati su Instagram con lo scopo di ricavare informazioni sull'oggettivazione del corpo femminile, analizzando commenti ed hashtag, oltre a fornire un'iniziale e indicativo profilo dell'utente che si trova dietro allo schermo dello smartphone o del computer. Con tale obiettivo, sono stati pensati tre indicatori oggettivi, ovvero non dipendenti dai soggetti ma basati su dati concreti, rispettivamente denominati:

- utilizzo del media, comprendente il numero di: like messi a foto e commenti altrui, fotografie, video e storie, commenti postati, contenuti salvati e ricerche effettuate.
- self-presentation, nel quale sono stati analizzati i dizionari di hashtag e commenti tramite il *text mining*.
- popolarità, interpretato come il numero di follower e following, in rapporto tra loro.

Oltre a ragionare sui contenuti in quanto tali, quantitativamente, sono stati considerati anche i tempi, sotto il profilo della fascia oraria e della mensilità, in modo da fornire indicazioni per future osservazioni.

La procedura, presentata nel capitolo successivo, è stata strutturata attraverso Rstudio, un software di tipo statistico open source, ovvero distribuito gratuitamente.

CAPITOLO DUE: METODOLOGIA

L'iniziale parsing effettuato dal codice in Python sui file JSON è finalizzato a trasformare il testo, sviluppato in una sola riga, in un elaborato ben formattato. Ciò è reso possibile da tre algoritmi, uno per ogni tipologia di struttura del file. La distribuzione degli elementi si differenzia per il momento necessario ad andare a capo. In particolare, la prima struttura ha bisogno di andare a caso dopo ogni parentesi graffa chiusa, la seconda dopo ogni parentesi quadra chiusa e la terza dopo ogni virgola. Lo scrip funziona su ogni file avente il nome richiesto, ne riconosce la struttura e lo elabora di conseguenza. Dalle cartelle in cui il parsing descritto è stato eseguito, i file JSON sono poi caricati sul software Rstudio e come di seguito analizzati.

2.1 Costruzione funzioni *utilities*

Una volta individuati i tre indicatori oggettivi ed il loro contenuto, sono state create delle funzioni generiche, da applicare singolarmente ad ogni caso, le quali permettono di individuare:

- la fascia oraria ed i mesi in cui sono stati inseriti i contenuti;
- la moda di questi parametri, mostrando l'orario ed il mese che compaiono con maggiore frequenza;
- l'indice di Entropia o di Shannon-Wiener.

2.1.1 Fascia oraria d'utilizzo e mensilità

Inizialmente, sono stati definiti i domini dei vettori `time_z` e `month_z` tramite la funzione generica `rep()`.

```
time_z = rep(NA,length(X))
month_z = rep(NA,length(X))
```

In seguito, è stato creato un ciclo for, anche detto ciclo di interazione, il quale permette la ripetizione programmata di una porzione di codice. La forma generale che assume un ciclo for è:

```
For(var in seq){
     # codice da ripetere
     }
```

In essa viene strutturata una variabile `var` che assume in sequenza tutti i valori indicati tramite il vettore `seq`. Il numero di interazioni, quindi, dipende dalla lunghezza indicata da `seq`, perché una volta che `var` avrà assunto uno dopo l'altro tutti i valori, il ciclo si concluderà.

Ciò che è di interesse trovare, ora, sono sia l'orario in cui sono stati inseriti i contenuti (approssimati alla sola ora, tralasciando dunque i minuti), sia in quale mese, così da costruire un'iniziale andamento di utilizzo del social media.
In accordo con la convenzionale divisione del giorno in quattro fasi, gli orari sono stati inseriti in altrettante categorie:

- mattino: dalle 5 alle 11;
- pomeriggio: dalle 12 alle 17;
- sera: dalle 18 alle 23;
- notte: dalle 00 alle 4.

Si potrà considerare, quindi, mattina l'ammontare di ore che trascorrono dalle 5:00 alle 11:59, pomeriggio dalle 12:00 alle 17:59 e così via. Il social media analizzato presenta la dicitura 00 per indicare la mezzanotte, dunque sarà sempre presente al posto delle 24.

Per ottenere le informazioni ricercate, viene utilizzata la funzione `strsplit(x,str)`, la quale divide le componenti del vettore alfanumerico `x` in corrispondenza alle sottostringhe specificate dalle componenti del vettore `str`. Partendo dall'orario, a `x` viene fatto corrispondere ogni elemento di `X` e allo `spit` equivale la lettera `T`: essa, infatti, è precede sempre l'orario e lo separa dalla data.

```
x = strsplit(x = X[i],split = "T",fixed = TRUE)
```

Questa prima segmentazione divide la data dal resto della stringa, composta da orario e parte testuale. Un'altra funzione `strsplit(x,str)` viene di seguito strutturata per spezzare ulteriormente la proposizione in ore, minuti e secondi (ancora uniti alla parte alfabetica). Infine, viene definito il vettore `time_y`, il quale corrisponderà esattamente all'orario in cui è stata postato il commento, ovvero la prima parte del segmento.

```
y = strsplit(x = x[[1]][2],split=":")
time_y = as.numeric(y[[1]][1])
```

Per riuscire a strutturare le fasce orarie, ho utilizzato un comando condizionale detto *if-then-else*, il quale permette di fare una scelta tra differenti funzioni alternative da eseguire. La sintassi è: if (condizione) statement (ramo then) else statement (ramo else). Se la condizione è vera, positiva, ovvero se il risultato è true, allora si esegue l'istruzione che segue la condizione (ramo true). Se la condizione non è rispettata e, quindi, la risposta è false, si esegue l'istruzione che segue la parola else (ramo else). Più semplicemente, viene indicato ad Rstudio cosa fare se (`if`) si presenta una certa situazione e cosa fare altrimenti (`else`) nel caso in cui la condizione espressa dalla `if` non fosse verificata.

La struttura generica è quindi:

```
if (condizione) {
istruzione uno
} else {
istruzione due
}
```

Applicato alla variabile `time_y` e agli orari, la struttura si complica ma mantiene la medesima base: il primo `if` inserirà i valori assunti da `time_z` in un intervallo numerico da 5 a 11 (interpretate poi come ore) e chiamerà il medesimo mattina.

Qualora il numero trovato in questa categoria, si passa al primo `else if`, dalle 12 alle 17 (pomeriggio), al secondo dalle 18 alle 23 (sera) e all'ultimo, al quale si arriva se il valore assunto da `time_z` non rientra in nessun intervallo fino ad ora presentato, ovvero se il numero rientra nell'intervallo compreso tra 0 e 4.

```
if(time_y %in% 5:11){
time_z[i] = "mattina"
}else if(time_y %in% 12:17){
time_z[i] = "pomeriggio"
}else if(time_y %in% 18:23){
time_z[i] = "sera"
}else{
time_z[i] = "notte"
}
```

In riferimento alla mensilità, invece, non è necessario alcun ciclo if, poiché essendo i mesi espressi numericamente e dunque di facile ed intuitiva comprensione, non è sensata una categorizzazione. Per accostare ad ogni numero il mese corrispondente (1 a gennaio, 2 a febbraio e così via) è sufficiente ricorrere alla dicitura `mont.abb[]`. Di seguito, si è proceduto a segmentare la stringa come fatto in precedenza, utilizzando il semplice trattino (-) come `spilt`. Questo perché la data è riportata nelle stringhe nella dicitura anno-mese-giorno. Una volta fatto ciò ed ottenuta una tripartizione di, appunto, anno, mese e giorno, al quale ultimo elemento è ancorato il commento, è stato sufficiente assegnare a `month_z` il valore numero della seconda riga.

```
x = strsplit(x = X[i],split = "-",fixed = TRUE)
month_z[i] = as.numeric(x[[1]][2])
```

Al ciclo for completo è possibile aggiungere il comando `print()` così da visualizzare facilmente il risultato:

```
for(i in 1:length(X)){
print(i)
x = strsplit(x = X[i],split = "T",fixed = TRUE)
y = strsplit(x = x[[1]][2],split=":")
time_y = as.numeric(y[[1]][1])
if(time_y %in% 5:11){
time_z[i] = "mattina"
}else if(time_y %in% 12:17){
time_z[i] = "pomeriggio"
}else if(time_y %in% 18:23){
time_z[i] = "sera"
}else{
time_z[i] = "notte"
}
x = strsplit(x = X[i],split = "-",fixed = TRUE)
month_z[i] = as.numeric(x[[1]][2])
}
```

2.1.2 Moda fascia oraria e mese

Tra gli indici di tendenza centrale, ovvero quegli indici che descrivono e riassumono la distribuzione dei dati con un valore che è centrale o molto importante per la distribuzione stessa, ho convenuto non fosse particolarmente significativo stimare la mediana o la media. Piuttosto, può rivelarsi decisamente interessante il valore assunto dalla moda (Mo), ovvero, in una distribuzione di dati rilevanti sulla variabile X, la modalità più frequente. È importante sottolineare, ai fini dell'interpretazione, che una distribuzione statistica può avere una moda, più mode o nessuna. Questo valore è stato considerato utile per definire in quale fascia oraria e in quali mesi è più frequente che una data persona utilizzi il media, inserisca contenuti, ricerchi o commenti profili altrui, incrementi o arresti la propria popolarità.

Per trovare la categoria d'orario e del mese più frequenti, ho creato un ciclo if, il quale opera diversamente a seconda che `x` contenga le serie relative al tempo o quelle relative alle mensilità. All'interno del ciclo, ho utilizzato la funzione `table()`, la cui

applicazione principale permette di tabulare i dati presenti in un dataset, ma può produrre tabelle di frequenza e di contingenza. Ottenuta la frequenza di ogni termine, è possibile ricavare la moda individuando l'elemento che si ripete il numero maggiore di volte. Nel primo caso, la funzione restituirà la fascia oraria con più alta frequenza, nel secondo il mese.

```
function(x,type=c("month","time")){
if(type=="time"){
tab = table(as.vector(x))
moda = names(tab)[tab == max(tab)]
}else if(type=="month"){
tab = table(as.vector(x))
moda = month.abb[as.numeric(names(tab)[tab == max(tab)])]
}
```

2.1.3 Indice di Entropia

Gli indici di eterogeneità permettono di definire la variabilità tra variabili categoriali e, tra essi, l'indice di Shannon-Wiener (o indice di Entropia) misura la diversità di una popolazione e si calcola tramite il negativo (perché ln(x) di un numero minore di uno è sempre negativo) della somma delle frequenze relative moltiplicate per il logaritmo delle frequenze relative. Espresso mediante una formula si ha:

$$H = -\sum_{i=1}^{k} p_i \ln(p_i)$$

dove k è l'intervallo di valori tra cui varia l'indice e i è l'indice della sommatoria. Dividendo H per il valore massimo possibile si ottiene un indice compreso tra 0 e 1. Da ciò si può ottenere:

- massima concentrazione: quando H tende a zero, tutte le risposte hanno circa lo stesso valore e l'informazione risulta utile, poiché fornire risposte nette. La diversità è minima.

- massima eterogeneità: quando H tende a uno, tutte le risposte hanno circa eguale frequenza e l'informazione è inutile poiché fornisce delle risposte vaghe. La diversità è massima.

Attraverso Rstudio, dunque, si calcola la frequenza relativa di ogni elemento di `time_z` prima e `month_z`, infine, si applica la formula sopra riportata.

```
tab = table(time_z)
x = prop.table(tab)
entropia = -(sum(x*log(x+1e-9)))/log(4)

tab = table(month_z)
x = prop.table(tab)
entropia = -(sum(x*log(x+1e-9)))/log(12)
```

2.2 Uso del media

Il primo passaggio per eseguire l'analisi è stato conteggiare separatamente tutti i parametri contenuti dall'indicatore «Uso del media». Il metodo utilizzato è simile per ogni elemento. Considerando, ad esempio, i commenti, si è proceduto come segue: viene indicato il percorso da seguire

```
path_folder =
"C:/Users/Operatore/Desktop/Tesi/Data/LauraJ/"
```

e si utilizza la funzione `readLines()` per caricare i file JSON, denominati come nello zip originale.

Parametri	File JSON
commenti	comments.json
likes	likes.json
media (foto, video e storie)	media.json
salvataggi	saved.json
ricerche	searches.json

È, inoltre, stato indicato l'utilizzata la codifica UTF-8, la quale definisce in maniera chiara ed univoca la corrispondenza fra un preciso carattere e un numero che rappresenta le coordinate di quel carattere nel repertorio di caratteri di Unicode.
È, poi, stata utilizzata la funzione `grep()` per ricercare le corrispondenze del pattern degli argomenti all'interno del file.

```
comm = readLines(paste0(path_folder,"comments.json"),
encoding = "UTF-8",warn = FALSE)
iid = grep(x = comm,pattern = "[\"",fixed = TRUE)
```

L'argomento pattern identifica una stringa di caratteri contenenti un'espressione fissa e regolare oppure una stringa di caratteri per `fixed = TRUE`, come in questo caso, da abbinare nel vettore di caratteri specificato. Per i parametri successivi, la procedura è stata la medesima, ma il pattern da utilizzare nella funzione `grep()` è cambiato in base alla struttura dei differenti documenti JSON. In particolare:

Parametri	Pattern utilizzati
commenti	`\"`
like	`[\"2`
storie	`stories/`
foto	`photos/`
video	`videos/`
salvataggi	`[\"`
ricerche	`search_click`

Dunque, sono state richiamate le funzioni *utilities* presentate nel paragrafo 2.1.1: in un primo momento è stata utilizzata la funzione `estrai_tempo`, la quale ha restituito i mesi in cui sono stati postati i contenuti nella prima lista e la categoria temporale nella seconda.

```
X_estratto = estrai_tempo(comm[iid])
month_z = X_estratto[[1]]; time_z = X_estratto[[2]
```

Similmente, la funzione `compute_moda` ha permesso di ottenere la moda del mese e della fascia oraria, così come la funzione `compute_entropy` ha portato la costruzione dei due indici di Entropia.

```
moda_mese = compute_moda(month_z,type = "month")
moda_orario = compute_moda(time_z,type = "time")
entropia_orario = compute_entropy(time_z)
entropia_mese = compute_entropy(month_z)
```

Infine, è stato proposto un unico vettore riassuntivo. Rimanendo nell'esempio dei commenti, infatti, all'interno del vettore `comments` sono stati inseriti il numero di commenti, ricavabile dalla lunghezza di `comm[iid]`, il mese e la categoria temporale con frequenza maggiore e i relativi indici di Entropia.

```
comments = NULL
comments$num = length(comm[iid]); comments$moda_time =
moda_orario; comments$moda_month = moda_mese;
comments$entropy_time = entropia_orario;
comments$entropy_month = entropia_mese
comments$months = month_z; comments$time = time_z;
```

Il tutto è stato inserito nella funzione `use_of_media = function(path_folder)` ed è quindi richiamabile con semplicità.

2.3 Self-presentation

Il secondo indice oggettivo che è stato strutturato è relativo alla self-presentation, ovvero il tentativo di presentare ed esprimere se stessi con la finalità di creare un'impressione prestabilita. Questo, all'interno di Instagram, può avvenire attraverso i contenuti caricati, come foto, video e stories, ma anche per mezzo delle didascalie e singole parole che accompagnano questi materiali. È risultato particolarmente significativo l'utilizzo degli hashtag, ovvero delle etichette che permettono di rendere più semplice la ricerca su un argomento o un tema specifico. Per questo, gli hashtag sono stati primamente conteggiati e poi analizzati attraverso il text mining.

2.3.1 Conteggio degli hashtag

Caricato il JSON relativo ai media come visto in precedenza, tramite la funzione `readLines()`, il pattern di `grep()` è stato cambiato in `caption`, in modo da isolare le stringhe contenenti la didascalia ed eliminando quelle relative alla data e l'ora in cui il post è stato creato e al percorso della foto all'interno dell'originale file zip.

```
media = readLines(paste0(path_folder,"media.json"),
encoding = "UTF-8",warn = FALSE)
iid = grep (x = media,pattern = "caption",fixed = TRUE)
```

Lavorando su queste stringhe, è stato avviato uno `strsplit()` per dividere il testo, lettera per lettera, in modo da semplificare la ricerca, e un successivo `grep()` relativo al simbolo cancelletto, il quale antecede sempre le etichette gergalmente chiamate hashtag.

```
num_hash = 0
for(i in 1:length(iid)){
x = strsplit(x = media[iid[i]],split = "",fixed = TRUE)
num_hash = num_hash + length(grep(x = x[[1]],pattern =
"#",fixed = TRUE))
}
```

2.3.2 Estrazione degli hashtag

Inizialmente, viene definito il vettore z tramite il comando `c()`, il quale consente di concatenare gli elementi che verranno inseriti. In riferimento a quanto fatto in precedenza, si considera `iid` come l'elemento contenente le didascalie, nelle quali si trovano gli hashtag. Si procede, dunque, alla costruzione di un ciclo for in cui inizialmente viene eseguito uno `split` per isolare la didascalia e, in essa, tramite la funzione `grep()` si ricerca la presenza del simbolo cancelletto.

```
z = c()
for(i in 1:length(media)){
x = strsplit(x = paste0(strsplit(x = media[iid[i]],split =
"\\\"",fixed = TRUE)[[1]],collapse = ""),
split = "\":",fixed = TRUE)
hast_yes = length(grep(x = x[[1]],pattern = "#",fixed =
TRUE))
```

Quindi, se la lunghezza del vettore è maggiore di uno (ovvero non è vuoto) e il numero di hashtag è maggiore di zero, si avvia un ulteriore ciclo if. In esso, viene eseguito un ulteriore `strsplit()` che separa il contenuto effettivo della didascalia dalla dicitura caption. Viene poi eseguito il comando per rimuovere ogni Unicode relativo alle emoticon, utilizzato uno `strsplit()` unito ad un `grep()` con pattern uguale a # per eseguire una prima ricerca degli hashtag. Il risultato è ancora grezzo e non restituisce precisamente tutte le parole di interesse: è necessario dividere lettera per lettera, individuare le posizioni del simbolo cancelletto, considerare le parole subito successive e rimuovere nuovamente gli spazi tra le lettere. Ciò che si ottiene è un'unica stringa contenente tutti gli hashtag utilizzati, i quali verranno poi collocati uno per uno in una lista. Si definisce così il vettore `z` come la divisione in sottostringhe sulla base degli hashtag.

```
if(length(x[[1]])>1 && hast_yes>0){
y = strsplit(x =
x[[1]][length(x[[1]])],split="\"")[[1]][2]
y1 = iconv(x = y, from = "latin1", to = "ASCII", sub="")
u = strsplit(x=y1,split=" ")[[1]];u=u [grep(x=u,pattern =
"#")]
 u =
mapply(function(j){v=strsplit(x=u[j],split="")[[1]];v=past
e0(v[min(grep(x=v,pattern =
"#")):length(v)],collapse="")},1:length(u))
z = c(z,strsplit(x = u,split = "#",fixed = TRUE))
    }
```

La lista di parole risultante viene dunque ordinata e vengono eliminati gli spazi vuoti. In seguito, viene eseguita un'ulteriore analisi del testo per individuare caratteri speciali come la chiocciola, comunemente usata per richiamare il nickname di un contatto, e quindi molto frequente. Stabilita che la lunghezza di y, la funzione contenente il dizionario di parole, è superiore a zero, la funzione z viene rinominata `txt_hash`.

```
z = unlist(z)
z = z[-which(z=="")]
for(i in 1:length(z)){
y = strsplit(x = z[i],split = "[\\\\@]")[[1]]
if(length(y)>0){
z[i] = y[1]
}
txt_hash = z
```

2.3.3 Estrazione delle didascalie

Similmente all'estrazione degli hashtag, anche per le didascalie, ovvero la parte testuale di senso compiuto che commenta ed accompagna il contenuto postato, il file di interesse è quello relativo ai media. In riferimento, quindi, al vettore `iid` contenente soltanto le stringhe relative alla caption, è stato strutturato un ciclo for all'interno del quale, per la lunghezza di tutto il vettore, viene inizialmente eseguito una separazione tra le parole basata sugli spazi. In seguito all'inserimento del codice ASCII, utile alla rimozione degli unicode corrispondenti alle emoticon, vengono rimossi le at (a commerciale) antecedenti al richiamo di un utente, i simboli cancelletto primamente analizzati come hashtag, e le diciture che separano video, immagini e stories. Si ottiene, in questo modo, un elenco di parole che, se riunite, compongono la singola o le molteplici frasi che accompagnano il post

```
z = c()
for(i in 1:length(iid)){
x = strsplit(x = media[iid[i]],split=" ")[[1]]
y = iconv(x = x, from = "latin1", to = "ASCII", sub="")
iid_wds = grep(x = y,pattern =
"@|#|\"caption\":|\"stories\":|\"videos\":|\"photos\":|\"p
rofile\":",invert = TRUE)
y=paste0(mapply(function(j){paste0(strsplit(x = y[j],split
= "\"\",")[[1]],collapse = " ")},iid_wds),collapse=" ")
z = c(z,y)
}
```

Può risultare interessante conteggiare quante didascalie sono presenti, poiché è possibile che il valore sia differente dalla quantità di media caricati: non sempre, infatti, vengono contestualizzati i contenuti.

```
num_captions = sum(z!="")
```

Al fine di operare un'ulteriore analisi testuale, come presentato nel capitolo tre, è utile anche accorpare tutte le didascalie in un singolo testo.

```
txt_captions = paste0(z,collapse=" ")
```

Riassumendo, viene inserito nel vettore `self_info` il numero e l'elenco degli hashtag, così come il conteggio e il testo integrale delle didascalie.

2.4 Popolarità

Il terzo indice oggettivo preso in considerazione riguarda la popolarità degli utenti all'interno del social media, ovvero quanto i loro contenuti sono in evidenza rispetto ad altri. Caricare una particolare foto e accompagnarla con i giusti termini può renderla fruibile a più persone, tendenzialmente interessate a quel contenuto. Per accrescere la propria fama e notorietà, dunque, è necessario rendersi visibili ad un pubblico

sempre più ampio. Nel dettaglio, è stato considerato il file relativo ai follower ed i following, all'interno del quale sono stati conteggiati e poi messi a confronto.

2.4.1 Conteggio follower e following

All'interno del file JSON relativo ai contatti caricato su Rstudio tramite la funzione `readLines()`, vengono individuate tre termini di soglia. Il testo inizia, infatti, con la dicitura follower, al quale seguono i nomi di tutte le persone e le pagine che seguono l'account considerato. L'ultimo di questi contatti antecede il termine following, oltre al quale si trovano i personaggi seguiti, ovvero dei quali si vedranno i contenuti all'interno della propria bacheca. L'ultimo marcatore è relativo agli hashtag seguiti, non considerati nel presente lavoro, ma utile a definire l'intervallo relativo ai following.

```
cnts = readLines(paste0(path_folder,"connections.json"),
encoding = "UTF-8",warn = FALSE)
iid = grep(x = cnts, pattern  = "follower",fixed = TRUE)
iid2 = grep(x = cnts, pattern  = "following",fixed = TRUE)
iid3 = grep(x = cnts, pattern  =
"following_hashtags",fixed = TRUE)
```

Per determinare il numero di follower, dunque, sarà sufficiente considerare la somma degli elementi tra la dicitura follower e following, priva degli spazi vuoti, a cui va sottratto uno, così da eliminare l'ultimo termine considerato, non inerente al conteggio. Lo stesso passaggio viene eseguito per calcolare il numero di following. Un primo e molto grezzo indice di popolarità può essere dato dal rapporto dei due numeri ricavati. L'utente può dirsi popolare nel momento in cui questo indicatore è molto superiore a 1, ovvero il numero delle persone seguite è molto minore rispetto a quello di coloro che lo seguono, mentre può non essere popolare se il numero ottenuto è molto inferiore a 1.

```
num_follower = sum(cnts[iid[1]:iid2[1]]!=" ")-1
num_following = sum(cnts[iid2[1]:iid3[1]]!=" ")-1
indice_popolarità = num_follower/num_following
```

Per ricavare degli indicatori di tempo legati ai follower, in termini di fasi del giorno e mensilità, è necessario considerare le stringhe non come semplici numeri, ma riferendosi al loro contenuto. Può essere significativo non tanto ai fini dell'uso del media, quanto piuttosto per comprendere i picchi di popolarità legati ad una persona, potendoli successivamente collegarli a particolari termini, frasi o contenuti caricati in quel determinato periodo. Un primo passaggio è, nuovamente, l'isolamento dei follower, all'interno dei quali vengono poi eliminati gli spazi vuoti. È necessario, come fatto in precedenza, rimuovere anche la dicitura followers.

```
X_flwr = cnts[iid[1]:iid2[1]-1]

X_flwr = X_flwr[X_flwr!=" "]

X_flwr = unlist(strsplit(x = X_flwr,split =
"followers"))[2:length(X_flwr)]
```

In questo modo si ottengono stringhe molti simili a quelle estratte nel primo indicatore oggettivo, ovvero l'utilizzo del media. E, di conseguenza, è possibile ricorrere alle funzioni *utilities* costruite nel paragrafo 2.1 per estrarre la moda e l'indice di entropia delle fasce orarie e dei mesi maggiormente frequenti.

```
X_estratto = estrai_tempo(X_flwr)

entropia_orario = compute_entropy(X_estratto[[2]])

entropia_mese = compute_entropy(X_estratto[[1]])

moda_orario = compute_moda(X_estratto[[2]] ,type = "time")

moda_mese = compute_moda(X_estratto[[1]],type="month")
```

Come previsto per gli altri indicatori oggetti, è stato infine costruito il vettore riassuntivo `info_popularity`, all'interno del quale vengono collocati:

- numero di follower e following, il loro rapporto;
- elenchi dei mesi e delle fasce orarie in cui sono stati caricati i contenuti;
- la moda e l'indice di entropia dei medesimi parametri.

```
info_popularity = NULL

info_popularity$num_follower = num_follower;
info_popularity$num_following = num_following;
info_popularity$entropy_time = entropia_orario;
info_popularity$entropy_month = entropia_mese;
info_popularity$moda_month = moda_mese;
info_popularity$moda_time = moda_orario;

info_popularity$months = X_estratto[[2]];
info_popularity$time = X_estratto[[1]];
```

CAPITOLO TRE: APPLICAZIONE

3.1 Costruzione dei dizionari di hashtag e caption

Tramite lo scaricamento, l'installazione e il richiamo dei pacchetti `readtext` e `quanteda`, viene eseguito il text mining, un'analisi improntata a dare valore aggiunto ai testi. Il primo passaggio riguarda la conversione del testo in un corpus, ovvero una collezione di unità di contesto o frammenti, ed è possibile svolgerlo tramite la funzione `corpus()`. È possibile ricavare preliminari informazioni del testo utilizzando la funzione `str()`, la quale restituisce la tipologia di documento, la lunghezza del testo, il percorso ed altre utili indicazioni descrittive. Per visualizzare la parte testuale del corpus, invece, è possibile applicare la funzione `View()`.

Ottenuto, dunque, un corpus, è possibile definire token l'occorrenza di una parola che appare o ricorre in un corpus. Attraverso il processo di tokenizzazione, anche definito con il termine parsing, ad ogni parole viene assegnato un doppio codice numerico, uno relativo alla sequenza diversa di caratteri alfabetici (tipo di parola) ed uno per ogni occorrenza incontrata, in modo da poter risalire alla sua posizione. In questo modo, se una stessa sequenza di caratteri chiamata type, ricorre più volte nel testo, essa sarà sempre associata al medesimo codice.

I token permettono così di lavorare sul testo considerando ogni singola parola, anche quando ripetuta. Tramite le funzioni `tokens()` vengono di seguito rimossi i numeri, la punteggiatura ed i simboli. Per evitare che parole con l'iniziale maiuscola vengano considerate diversamente da termini uguali ma interamente minuscoli, tutti i caratteri vengono convertiti in caratteri minuscoli.

```
hash_tok = corpus(z)
hash_tok = tokens(hash_tok, remove_punct=TRUE,
                  remove_numbers = TRUE,
                  remove_symbols=TRUE)
hash_tok = tokens_tolower(hash_tok)
```

Un ulteriore passaggio è relativo alla rimozione delle stopwords, un insieme di parole generalmente considerate ad alta frequenza e, dunque, poco significative in relazione ad un'analisi di contenuto. Il passaggio viene eseguito sia per la lingua italiana, sia per quella inglese, così da comprendere un maggior numero di hashtag. Specificando, nella funzione `tokens_remove()` padding = FALSE, si specifica che non deve essere lasciata una stringa vuota al posto dei token rimossi. Verbose = TRUE, invece, permette di ricevere il numero di token così rimossi.

```
stopit = stopwords(language = "it")
hash_tok = tokens_remove(hash_tok, stopit, verbose = TRUE,
padding = FALSE)
stopen = stopwords(language = "en")
hash_tok = tokens_remove(hash_tok, stopen, verbose = TRUE,
padding = FALSE)
```

Ciò che risulta è un testo pulito, contenente soltanto gli hashtag d'interesse. I medesimi passaggi possono essere applicati al testo contenente tutte le didascalie estratte. Di conseguenza, si otterranno due dizionari, il primo relativo agli hashtag ed il secondo alle caption.

3.1.1 Matrice DFM e Wordcloud

Può essere significativo costruire una matrice di caratteristiche del documento (document-feature matrix) per entrambi i dizionari attraverso la funzione `dfm()`. Ciò che si otterrà è una matrice contenente tutti gli hashtag da una parte e tutti i termini presenti nelle didascalie dall'altra, con la relativa frequenza assoluta, ovvero quante volte compaiono all'interno del corpus. Possono essere utilizzare le funzioni `ndoc()` per ricavare il numero di documenti all'interno della matrice e la funzione `nfeat()` per il numero di elementi.

Applicando quanto esplicitato ad un caso studio, è inoltre possibile utilizzare la funzione `textplot_wordcloud()` per ottenere un grafico wordcloud, ovvero una rappresentazione grafica di una lista di parole, rappresentate e disposte in base alla

loro frequenza. I termini più usati, infatti, saranno centrali, più grandi e di un colore differente rispetto a quelli meno usati, sempre più piccoli, lontani e dai colori tenui man mano che la frequenza diminuisce.

Applicando le procedure di text mining al caso studio, è possibile poi creare una matrice DFM e, di seguito, i wordcloud.

```
dfm_hash = dfm(hash_tok, stem=FALSE) #creazione matrice
dfm
textplot_wordcloud(dfm_hash, max_size = 4, min_size = 1,
max_word = 200, color = c("lightsteelblue3", "skyblue2",
"steelblue", "dodgerblue4"))
```

Figura 1: wordcloud relativo al dizionario degli hashtag.

Figura 2: wordcloud relativo al dizionario delle didascalie.

3.2 I tre indicatori nel caso studio

Per applicare il codice scritto in Rstudio a diversi utenti, è possibile caricare sulla piattaforma il file contenente le funzioni illustrate nel paragrafo 2.1 e poi delineare il percorso della cartella corrispondente.

```
rm(list=ls())

source("C:/Users/Cami/Desktop/functions.R")

path_folder = "C:/Users/Cami/Desktop/Tesi/LauraJ/"
```

Nel presente esempio, viene considerato l'utente LauraJ, e di conseguenza il vettore creato prende il medesimo nome. All'interno dello stesso vengono richiamate le tre sintesi degli indicatori oggettivi, contenenti tutti i fattori individuati precedentemente. In particolare, vengono riassunti l'utilizzo del media, la self-presentation e la popolarità.

```
LauraJ = list()

LauraJ[[1]] = use_of_media(path_folder)

LauraJ[[2]] = self_presentation(path_folder)

LauraJ[[3]] = popularity(path_folder)

str(LauraJ)
```

Dalla restituzione, si ottiene una lista di tre elementi, a propria volta suddivisa in nove liste. Le prima fa riferimento ai commenti, il cui conteggio ha come risultato 149. Sono stati scritti per la maggior parte al mattino e si riscontra una particolare attività ad ottobre. L'entropia della categoria temporale e del mese tendono, anche se in misura diversa, ad uno, e questo può aiutare ad interpretare le due mode: anche se una classe viene evidenziata, le restanti (pomeriggio, sera e notte nel primo caso, i restanti mesi nel secondo) hanno ugualmente una frequenza molto simile, sebbene minore. Instagram, di conseguenza, è stato utilizzato approssimativamente in modo frequente. La lista dei mesi e delle categorie relative all'orario sono poi elencate.

```
  ..$ comments:List of 8
  .. ..$ num          : int 149
  .. ..$ moda_time    : chr "mattina"
  .. ..$ moda_month   : chr "Oct"
  .. ..$ entropy_time : num 0.883
  .. ..$ entropy_month: num 0.961
  .. ..$ num_month    : int 149
  .. ..$ months       : num [1:149] 5 5 4 4 3 3 2 2 2 2 ...
  .. ..$ time         : chr [1:149] "pomeriggio" "pomeriggio"
"mattina" "mattina" ...
```

Il medesimo risultato si ottiene dalle altre componenti del primo indicatore oggettivo, ovvero l'utilizzo del media.

```
  ..$ likes   :List of 7
  .. ..$ num          : int 20923
  .. ..$ moda_time    : chr "pomeriggio"
  .. ..$ moda_month   : chr "Mar"
  .. ..$ entropy_time : num 0.889
  .. ..$ entropy_month: num 0.959
  .. ..$ months       : num [1:20923] 5 5 5 5 5 5 5 5 5 5 ...
  .. ..$ time         : chr [1:20923] "notte" "notte" "notte"
"notte" ...
  ..$ stories :List of 7
  .. ..$ num          : int 122
  .. ..$ moda_time    : chr "mattina"
  .. ..$ moda_month   : chr "Nov"
  .. ..$ entropy_time : num 0.826
  .. ..$ entropy_month: num 0.922
  .. ..$ months       : num [1:122] 5 5 5 5 5 5 5 5 4 4 ...
  .. ..$ time         : chr [1:122] "pomeriggio" "pomeriggio"
"pomeriggio" "mattina" ...
  ..$ videos  :List of 7
  .. ..$ num          : int 15
  .. ..$ moda_time    : chr "mattina"
  .. ..$ moda_month   : chr "Oct"
  .. ..$ entropy_time : num 0.919
  .. ..$ entropy_month: num 0.915
  .. ..$ months       : num [1:15] 4 4 4 3 3 3 1 1 10 10 ...
  .. ..$ time         : chr [1:15] "mattina" "notte" "notte" "
mattina" ...
  ..$ photos  :List of 7
  .. ..$ num          : int 515
  .. ..$ moda_time    : chr "mattina"
  .. ..$ moda_month   : chr "Oct"
  .. ..$ entropy_time : num 0.817
```

```
  .. ..$ entropy_month: num 0.952
  .. ..$ months       : num [1:515] 5 5 5 4 4 4 4 4 4 4 ...
  .. ..$ time         : chr [1:515] "mattina" "notte" "notte"
"mattina" ...
  ..$ saved   :List of 7
  .. ..$ num          : int 4
  .. ..$ moda_time    : chr "pomeriggio"
  .. ..$ moda_month   : chr [1:2] "Apr" "Nov"
  .. ..$ entropy_time : num 0.946
  .. ..$ entropy_month: num 1
  .. ..$ months       : num [1:4] 4 11 4 11
  .. ..$ time         : chr [1:4] "sera" "pomeriggio" "pomerig
gio" "notte"
  ..$ searches:List of 7
  .. ..$ num          : int 3
  .. ..$ moda_time    : chr "pomeriggio"
  .. ..$ moda_month   : chr "Apr"
  .. ..$ entropy_time : logi NA
  .. ..$ entropy_month: logi NA
  .. ..$  months      : num [1:3] 4 4 4
  .. ..$ time         : chr [1:3] "pomeriggio" "pomeriggio" "p
omeriggio"
```

Un particolare interessante viene riscontrato nell'ultima lista, relativa alle ricerche effettuate. Dal momento che gli oggetti ricercati sono solamente tre e tutti effettuati nella medesima fascia oraria e nello stesso mese, l'entropia non restituisce alcun risultato.

Relativamente al secondo indicatore, ovvero l'indicatore oggettivo della self-presentation, e quindi alla seconda lista, il codice restituisce il numero e l'elenco sia degli hashtag, sia delle didascalie. È riportata anche la dicitura "truncated", che indica come il testo contenente tutte le caption sia in realtà molto più grande.

```
  ..$ num_hash    : num 848
  ..$ txt_hash    : chr [1:848] "nomakeup" "tantaansia" "morta
ccciloro" "machimelohafattofare" ...
  ..$ num_captions: int 383
  ..$ txt_captions: chr "\"Dopotutto sono Sagittario ascendent
e Scorpione\\n Just Perfect\",    \"Dreaming about....   \"Swe
et memories "| __truncated__
```

La terza lista corrisponde all'indicatore oggettivo relativo alla popolarità. Viene restituito il numero di follower e di following, il loro rapporto, la moda e l'indice di entropia dei mesi e della fascia oraria relative alle persone che hanno iniziato a seguire l'utente, oltre all'elenco di questi parametri. Anche in questo caso, come in molti dei precedenti, l'indice di Shannon-Wiener sottolinea come i valori più frequenti lo siano, rispetto agli altri, di molto poco. È possibile di conseguenza dedurre che il numero di follower cresce in modo abbastanza stabile nel tempo e non ha picchi in determinati orari o in dati mesi. In merito all'elemento popolarità, inoltre, è facilmente intuibile come il numero di follower sia minore rispetto ai profili seguiti.

```
  ..$ num_follower : num 249
  ..$ num_following: num 320
  ..$ popularity   : num 0.778
  ..$ entropy_time : num 0.946
  ..$ entropy_month: num 0.986
  ..$ moda_month   : chr "Nov"
  ..$ moda_time    : chr "mattina"
  ..$ months       : chr [1:248] "sera" "mattina" "notte" "not
te" ...
  ..$ time         : num [1:248] 5 5 5 5 5 5 4 4 4 4 ...
```

APPENDICE

```
rm(list=ls())
source("C:/Users/Operatore/Desktop/functions.R")

path_folder = "C:/Users/Operatore/Desktop/Tesi/Data/LauraJ/"
LauraJ = list()
LauraJ[[1]] = use_of_media(path_folder)
LauraJ[[2]] = self_presentation(path_folder)
LauraJ[[3]] = popularity(path_folder)
str(LauraJ)

# Utilities ----------------------------------------------------
-------------
estrai_tempo = function(X){
  time_z = rep(NA,length(X))
  month_z = rep(NA,length(X))
  for(i in 1:length(X)){
    x = strsplit(x = X[i],split = "T",fixed = TRUE)
    y = strsplit(x = x[[1]][2],split=":")
    time_y = as.numeric(y[[1]][1])
    if(time_y %in% 5:11){
      time_z[i] = "mattina"
    }else if(time_y %in% 12:17){
      time_z[i] = "pomeriggio"
    }else if(time_y %in% 18:23){
      time_z[i] = "sera"
    }else{
      time_z[i] = "notte"
    }

    x = strsplit(x = X[i],split = "-",fixed = TRUE)
    month_z[i] = as.numeric(x[[1]][2])
  }
  return(list(month_z,time_z))
}
```

```
compute_moda = function(x,type=c("month","time")){
  if(type=="time"){
    tab = table(as.vector(x))
    moda = names(tab)[tab == max(tab)]
  }else if(type=="month"){
    tab = table(as.vector(x))
    moda = month.abb[as.numeric(names(tab)[tab == max(tab)])]
  }
  return(moda)
}

compute_entropy = function(x){
    x = prop.table(table(x))
    entropia=NA
    if(length(x)>1){
      entropia = -(sum(x*log(x+1e-9)))/log(length(x))
    }
  return(entropia)
}

get_os <- function(){
  sysinf <- Sys.info()
  if (!is.null(sysinf)){
    os <- sysinf['sysname']
    if (os == 'Darwin')
      os <- "osx"
  } else { ## mystery machine
    os <- .Platform$OS.type
    if (grepl("^darwin", R.version$os))
      os <- "osx"
    if (grepl("linux-gnu", R.version$os))
      os <- "linux"
  }
  tolower(os)
}
```

```
fix_path_folder = function(current_path){
  if(get_os()%in%c("linux","osx")){
    return(paste0(current_path,"/"))
  }else{
    return(paste0(current_path,"\\"))
  }
}


# Use of media ------------------------------------------------
---------------------

use_of_media = function(path_folder){

  ### --------------------------------- COMMENTS.JSON --------
------------------------- ###
  comm = readLines(paste0(path_folder,"comments.json"),
encoding = "UTF-8",warn = FALSE)
  iid = grep(x = comm,pattern = "[\"",fixed = TRUE)

  X_estratto = estrai_tempo(comm[iid])
  month_z = X_estratto[[1]]; time_z = X_estratto[[2]]

  moda_mese = compute_moda(month_z,type = "month")
  moda_orario = compute_moda(time_z,type = "time")

  entropia_orario = compute_entropy(time_z)
  entropia_mese = compute_entropy(month_z)

  comments = NULL
  comments$num = length(comm[iid]); comments$moda_time =
moda_orario; comments$moda_month = moda_mese;
comments$entropy_time = entropia_orario;
comments$entropy_month = entropia_mese
  comments$num_month = length(month_z); comments$months =
month_z; comments$time = time_z;
```

```
  ### --------------------------------- LIKES.JSON -----------
---------------------- ###
  lks = readLines(paste0(path_folder,"likes.json"), encoding =
"UTF-8",warn = FALSE)
  iid = grep(x = lks,pattern = "[\"2",fixed = TRUE) #2 Ã¨ una
costante per la ricerca delle righe (Instagram esiste solo dal
2010)


  X_estratto = estrai_tempo(lks[iid])
  month_z = X_estratto[[1]]; time_z = X_estratto[[2]]


  moda_mese = compute_moda(month_z,type = "month")
  moda_orario = compute_moda(time_z,type = "time")


  entropia_orario = compute_entropy(time_z)
  entropia_mese = compute_entropy(month_z)


  likes = NULL
  likes$num = length(lks[iid]); likes$moda_time = moda_orario;
likes$moda_month = moda_mese; likes$entropy_time =
entropia_orario; likes$entropy_month = entropia_mese
  likes$months = month_z; likes$time = time_z;


  ### --------------------------------- MEDIA.JSON -----------
---------------------- ###
  media = readLines(paste0(path_folder,"media.json"), encoding
= "UTF-8",warn = FALSE)


  ## stories
  iid = grep(x = media, pattern  = "stories/",fixed = TRUE)
  X = media[iid-1]


  X_estratto = estrai_tempo(X)
  month_z = X_estratto[[1]]; time_z = X_estratto[[2]]


  moda_mese = compute_moda(month_z,type = "month")
  moda_orario = compute_moda(time_z,type = "time")
```

```
  entropia_orario = compute_entropy(time_z)
  entropia_mese = compute_entropy(month_z)

  stories = NULL
  stories$num = length(media[iid]); stories$moda_time =
moda_orario; stories$moda_month = moda_mese;
stories$entropy_time = entropia_orario; stories$entropy_month
= entropia_mese
  stories$months = month_z; stories$time = time_z;

  ## photos
  iid = grep(x = media, pattern  = "photos/",fixed = TRUE)
  X = media[iid-1]

  X_estratto = estrai_tempo(X)
  month_z = X_estratto[[1]]; time_z = X_estratto[[2]]

  moda_mese = compute_moda(month_z,type = "month")
  moda_orario = compute_moda(time_z,type = "time")

  entropia_orario = compute_entropy(time_z)
  entropia_mese = compute_entropy(month_z)

  photos = NULL
  photos$num = length(media[iid]); photos$moda_time =
moda_orario; photos$moda_month = moda_mese;
photos$entropy_time = entropia_orario; photos$entropy_month =
entropia_mese
  photos$months = month_z; photos$time = time_z;

  ## videos
  iid = grep(x = media, pattern  = "videos/",fixed = TRUE)
  X = media[iid-1]

  X_estratto = estrai_tempo(X)
  month_z = X_estratto[[1]]; time_z = X_estratto[[2]]
```

```
  moda_mese = compute_moda(month_z,type = "month")
  moda_orario = compute_moda(time_z,type = "time")

  entropia_orario = compute_entropy(time_z)
  entropia_mese = compute_entropy(month_z)

  videos = NULL
  videos$num = length(media[iid]); videos$moda_time =
moda_orario; videos$moda_month = moda_mese;
videos$entropy_time = entropia_orario; videos$entropy_month =
entropia_mese
  videos$months = month_z; videos$time = time_z;

  ### --------------------------------- SAVED.JSON -----------
---------------------- ###
  savd = readLines(paste0(path_folder,"saved.json"), encoding
= "UTF-8",warn = FALSE)
  iid = grep (x = savd,pattern = "[\"",fixed = TRUE)

  X = savd[iid]

  X_estratto = estrai_tempo(X)
  month_z = X_estratto[[1]]; time_z = X_estratto[[2]]

  moda_mese = compute_moda(month_z,type = "month")
  moda_orario = compute_moda(time_z,type = "time")

  entropia_orario = compute_entropy(time_z)
  entropia_mese = compute_entropy(month_z)

  saved = NULL
  saved$num = length(savd[iid]); saved$moda_time =
moda_orario; saved$moda_month = moda_mese; saved$entropy_time
= entropia_orario; saved$entropy_month = entropia_mese
  saved$months = month_z; saved$time = time_z;

  ### --------------------------------- SEARCHES.JSON --------
------------------------ ###
```

```
  srch = readLines(paste0(path_folder,"searches.json"),
encoding = "UTF-8",warn = FALSE)
  iid = grep(x = srch, pattern  = "search_click",fixed = TRUE)

  X = srch[iid+1]

  X_estratto = estrai_tempo(X)
  month_z = X_estratto[[1]]; time_z = X_estratto[[2]]

  moda_mese = compute_moda(month_z,type = "month")
  moda_orario = compute_moda(time_z,type = "time")

  entropia_orario = compute_entropy(time_z)
  entropia_mese = compute_entropy(month_z)

  searches = NULL
  searches$num = length(srch[iid]); searches$moda_time =
moda_orario; searches$moda_month = moda_mese;
searches$entropy_time = entropia_orario;
searches$entropy_month = entropia_mese

  media_usage = NULL
  media_usage$comments = comments; media_usage$likes = likes;
media_usage$stories = stories; media_usage$videos = videos;
media_usage$photos = photos; media_usage$saved = saved;
media_usage$searches = searches
  media_usage$months = month_z; media_usage$time = time_z;

  return(media_usage)
}



# Self-presentation ------------------------------------------
-------------

self_presentation = function(path_folder){
```

```
  media = readLines(paste0(path_folder,"media.json"), encoding
= "UTF-8",warn = FALSE)
  iid = grep (x = media,pattern = "caption",fixed = TRUE)


  ## Numero di hashtag
  num_hash = 0
  for(i in 1:length(iid)){
    x = strsplit(x = media[iid[i]],split = "",fixed = TRUE)
    num_hash = num_hash + length(grep(x = x[[1]],pattern =
"#",fixed = TRUE))
  }



  ## Estrazione hashtag
  z = c()
  for(i in 1:length(media)){
    #x = strsplit(x = media[iid[i]],split = "\":",fixed =
TRUE)
    x = strsplit(x = paste0(strsplit(x = media[iid[i]],split =
"\\\"",fixed = TRUE)[[1]],collapse = ""),
                 split = "\":",fixed = TRUE)
    hast_yes = length(grep(x = x[[1]],pattern = "#",fixed =
TRUE))
    if(length(x[[1]])>1 && hast_yes>0){
      y = strsplit(x =
x[[1]][length(x[[1]])],split="\"")[[1]][2]
      y1 = iconv(x = y, from = "latin1", to = "ASCII", sub="")
      u=strsplit(x=y1,split=" ")[[1]];u=u[grep(x=u,pattern =
"#")] #cerca hashtag primo livello

u=mapply(function(j){v=strsplit(x=u[j],split="")[[1]];v=paste0
(v[min(grep(x=v,pattern =
"#")):length(v)],collapse="")},1:length(u))
      z = c(z,strsplit(x = u,split = "#",fixed = TRUE))
    }
  }
  z = unlist(z) #lista ordinata
  z = z[-which(z=="")] #eliminare spazi vuoti
  for(i in 1:length(z)){
```

```
    y = strsplit(x = z[i],split = "[\\\\@]")[[1]]
    if(length(y)>0){
      z[i] = y[1] #da controllare che sia sempre al primo
posto l'hastag
    }
  }
  txt_hash = z

  ## Estrazione captions
  z = c()
  for(i in 1:length(iid)){
    x = strsplit(x = media[iid[i]],split=" ")[[1]]
    y = iconv(x = x, from = "latin1", to = "ASCII", sub="")
    iid_wds = grep(x = y,pattern =
"@|#|\"caption\":|\"stories\":|\"videos\":|\"photos\":|\"profi
le\":",invert = TRUE)
    y=paste0(mapply(function(j){paste0(strsplit(x = y[j],split
= "\"\",")[[1]],collapse = " ")},iid_wds),collapse=" ")
    z = c(z,y)
  }
  num_captions = sum(z!="")
  txt_captions = paste0(z,collapse=" ")

  self_info = NULL
  self_info$num_hash = num_hash; self_info$txt_hash =
txt_hash; self_info$num_captions = num_captions;
self_info$txt_captions = txt_captions;

  return(self_info)
}



# Popularity ------------------------------------------------
-------------

popularity = function(path_folder){
```

```
  cnts = readLines(paste0(path_folder,"connections.json"),
encoding = "UTF-8",warn = FALSE)


  iid = grep(x = cnts, pattern  = "follower",fixed = TRUE)
  iid2 = grep(x = cnts, pattern  = "following",fixed = TRUE)
  iid3 = grep(x = cnts, pattern  = "following_hashtags",fixed
= TRUE)


  num_follower = sum(cnts[iid[1]:iid2[1]]!=" ")-1
  num_following = sum(cnts[iid2[1]:iid3[1]]!=" ")-1
  popularity = num_follower/num_following


  X_flwr = cnts[iid[1]:iid2[1]-1]
  X_flwr = X_flwr[X_flwr!=" "]
  X_flwr = unlist(strsplit(x = X_flwr,split =
"followers"))[2:length(X_flwr)] #elimina l'indicazione dei
followers


  X_estratto = estrai_tempo(X_flwr)
  entropia_orario = compute_entropy(X_estratto[[2]])
  entropia_mese = compute_entropy(X_estratto[[1]])
  moda_orario = compute_moda(X_estratto[[2]],type = "time")
  moda_mese = compute_moda(X_estratto[[1]],type="month")


  info_popularity = NULL
  info_popularity$num_follower = num_follower;
info_popularity$num_following = num_following;
info_popularity$popularity = popularity;
info_popularity$entropy_time = entropia_orario;
info_popularity$entropy_month = entropia_mese;
info_popularity$moda_month = moda_mese;
info_popularity$moda_time = moda_orario;
  info_popularity$months = X_estratto[[2]];
info_popularity$time = X_estratto[[1]];


  return(info_popularity)
}
```

BIBLOGRAFICA E SITOGRAFIA

Riva, G. (2014), *Nativi digitali. Crescere e apprendere nel mondo dei nuovi media*. Bologna: Il Mulino.

Marmo, R. (2016). *Social media mining: estrarre e analizzare informazioni dai social media*. Milano: Hoepli

Harlow, L. L., Oswald, F. L. (2016). Big Data in Psychology: Introduction to Special Issue. *Psychol Method*, 447-457

Azzalini, A., & Scarpa, B. (2012). *Data analysis and data mining: An introduction*. OUP USA.

Sensis (2017). Social networking sites used. *Social Media Report 2017*.

Tiggemann, M., Barbato, I. (2018). “You look great!”: The effect of viewing appearance-related Instagram comments on women’s body image. *Body Image* 27, 61-66.

Bolasco S., con prefazione di De Mauro T. (2013). *L’analisi automatica dei testi: fare ricerca con il text mining*. Roma: Carocci editore.

https://it.wikipedia.org/wiki/Big_data

https://cran.r-project.org/

http://www.insular.it/

https://www.rdocumentation.org/

https://tutorials.quanteda.io/