

UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA
Scuola di Economia e Statistica
Corso di Laurea in Scienze Statistiche ed
Economiche



Metodi per l'elaborazione del linguaggio
naturale: un'applicazione sui messaggi di Twitter
durante la pandemia in Italia

Relatore:

Ch.mo prof. Aldo Solari

Candidato:

Paolo Bussacchini

Matricola 826677

Anno Accademico
2019/2020

Indice

Elenco delle figure	v
Elenco delle tabelle	vi
Introduzione	1
1 Collezione dei dati & Pre-processing	3
1.1 Estrazione dei tweet mediante API	3
1.1.1 Le API di Twitter	3
1.1.2 Tweet mining su R	4
1.2 Estensione del dataset con Hydrator	6
1.2.1 Il processo di hydrating	6
1.2.2 Descrizione delle variabili e creazione del corpus finale	7
1.3 Pre-processing dei testi	9
1.3.1 La libreria TextWiller	9
1.3.2 Ulteriori sviluppi	11
1.4 Operazioni preliminari	12
2 Analisi Esplorativa	14
2.1 Premesse	14
2.1.1 Tokenization	14
2.1.2 La Document Term Matrix	15
2.2 Prime analisi descrittive	15
2.2.1 Messaggi più retweettati	17
2.2.2 Keyword Trend Analysis	17
2.2.3 Hashtags & Mentions	18
2.3 Analisi degli n-grammi	20
2.4 Termini più importanti per tf-idf	22
3 Analisi dei Retweet	26
3.1 Estrazione dei tweet originali	26
3.2 Visualizzazione della rete di retweet	27

3.3	Valutazione del Time-to-Retweet	28
3.4	TF-IDF feature per i retweet più rapidi	31
4	Sentiment Analysis	33
4.1	Introduzione alla Sentiment Analysis	33
4.1.1	Metodi di classificazione	34
4.1.2	Problematiche nella Sentiment Analysis	35
4.1.3	La lingua italiana e il contesto di riferimento	36
4.2	Classificazione degli unigrammi	37
4.3	Andamento del sentiment nel tempo	39
4.4	Limiti e possibili sviluppi	41
5	Topic Modeling	42
5.1	Introduzione al Topic Modeling	42
5.2	Latent Dirichlet Allocation: il modello teorico	44
5.2.1	Stima dei parametri: il campionamento di Gibbs	46
5.3	Implementazione	47
5.3.1	Selezione del numero di topic	47
5.4	Risultati	49
5.4.1	Trend temporale dei topic	51
	Conclusioni	56
	A Codice	58
	B Italian Stopwords	72
	Bibliografia	76

Elenco delle figure

1.1	Alcuni dei dataset estratti	7
2.1	Wordcloud degli unigrammi	21
2.2	Parole più importanti rispetto alla tf-idf	24
3.1	Rete di retweet tra gli autori	28
3.2	Distribuzione logaritmica del time-to-retweet espresso in ore	31
3.3	Valori di tf-idf per i testi dei retweet più rapidi	32
3.4	Valori di tf-idf per le descrizioni degli user più retweettati	32
4.1	Primi 15 termini suddivisi per polarità	38
4.2	Andamento sentiment dal 14/02 al 5/03	39
4.3	Andamento sentiment dal 6/03 al 28/03	39
4.4	Andamento sentiment dal 29/03 al 20/04	40
4.5	Andamento sentiment dal 21/04 al 12/05	40
5.1	Esempio introduttivo di topic model	44
5.2	Modello grafico per LDA	46
5.3	Coherence Score per K da 1 a 20	49
5.4	Trend topic 2	52
5.5	Trend topic 7	53
5.6	Trend topic 1	54

Elenco delle tabelle

1.1	Conteggi dei tweet per nazionalità	13
2.1	Estratto di DTM	15
2.2	Conteggi dei tweet per settimana	16
2.3	Messaggi con il maggior numero di retweet	17
2.4	Keyword Trend Analysis	17
2.5	Top 20 mentions	18
2.6	Top 20 hashtags	19
2.7	User più attivi	19
2.8	Bigrammi individuati	21
2.9	Trigrammi individuati	22
5.1	Composizione dei topic individuati	50

Introduzione

La pandemia di COVID-19 del 2020 ha duramente colpito in ogni parte del mondo qualsiasi attività umana: dall'economia ai sistemi sanitari, passando per il turismo e i servizi alla persona, fino ad arrivare all'istruzione; sono veramente pochi i settori rimasti intoccati.

Per contrastarne le drammatiche conseguenze, concorrenti a creare una situazione di incertezza che non si era più sperimentata dal dopoguerra in poi, ai governi e alle istituzioni sono state richiesti tempestività d'azione e sforzi immani. Parimenti, la comunità scientifica, afferente ai più svariati campi, è stata sollecitata a concentrare tutte le sue risorse per aiutare le autorità competenti. Si fa ovviamente riferimento in primis alla microbiologia ed in particolare alla virologia, ma non solo: si pensi ad esempio alle task force istituite per programmare la ripartenza economica, alla necessità di un'app per il tracciamento degli infetti o al controllo della curva epidemica.

È dunque evidente che scienze quali l'epidemiologia, la statistica e l'informatica siano state protagoniste nella ricerca collegata al coronavirus. Anche in questo caso, così come accaduto in ambito biologico con, ad esempio, la pubblicazione della sequenza genetica del SARS-CoV-2, si è assistito ad una mobilitazione e condivisione di risorse che raramente si era verificata prima. È stata infatti resa disponibile online una quantità veramente notevole di articoli, dataset e codici (si faccia riferimento a [1]).

Tra i vari repository non mancano i dati estratti dai social media, diventati sempre più cassa di risonanza dell'opinione pubblica nel corso dell'ultimo decennio. In particolare, in caso di catastrofi naturali, attacchi terroristici e appunto diffusione di malattie, ad assurgere a principale indiziato per la monitorizzazione delle conversazioni pubbliche è Twitter, la piattaforma di microblogging più popolare del mondo. In tempi di incertezza le peculiarità di Twitter, quali l'accessibilità e la necessità di concentrare informazioni in pochi caratteri, ne veicolano al contempo l'utilizzo e lo studio del contenuto dei messaggi, indice dell'umore degli utenti e delle loro opinioni. Ciò risulta ancora più significativo considerando la velocità con cui cambia la situazione nel corso di una pandemia e l'imprevedibilità durante questi tempi avversi.

Partendo da questa premessa, l'elaborato in questione si pone l'obiettivo di rispondere all'incremento esponenziale dell'attività su Twitter, avvenuto contestualmente alla diffusione dell'epidemia, analizzando i tweet da un punto di vista sintattico e semantico con l'applicazione di tecniche di Natural Language Processing. In particolare, ci si concentra sui messaggi pubblicati in Italia in un periodo che copre le fasi più concitate della pandemia. Questa scelta nasce dall'assenza di analisi dettagliate su questi dati, a differenza delle numerose effettuate sui tweet in lingua inglese, e fornisce l'occasione per approfondire alcune tecniche di text mining e dare un'idea dell'intero processo che sta alla base del NLP: dalla raccolta dei dati fino alla consolidamento della conoscenza scoperta.

La tesi è articolata nei seguenti capitoli:

- **Capitolo 1:** vengono illustrati i due procedimenti che sono stati seguiti per la creazione del dataset finale. Si procede con la descrizione e la selezione delle variabili di interesse, per poi passare all'applicazione di metodi di pre-processing sul testo dei tweet, principale indiziato dell'analisi;
- **Capitolo 2:** vengono presentate alcune analisi descrittive per comprendere meglio la struttura del dataset e coglierne approssimativamente il contenuto informativo; si prosegue in questo senso concentrando l'attenzione su alcune questioni di rilevanza per il tema trattato;
- **Capitolo 3:** viene effettuato una breve approfondimento sui retweet, valutandone sia il "response time" [2] che il contenuto. In particolare, si confronta la velocità di retweet per il COVID-19 con altre emergenze, collegate specialmente alla diffusione di malattie infettive;
- **Capitolo 4:** si valutano le ripercussioni psicologiche causate dalla pandemia analizzando il contenuto sentimentale dei messaggi. In questo senso, ci si limita ad una classificazione della polarità (positiva o negativa) a livello degli unigrammi. Si perviene quindi all'andamento del sentiment nel corso dei mesi;
- **Capitolo 5:** il modello statistico Latent Dirichlet Allocation (LDA) viene applicato ai tweet al fine di ottenere una suddivisione dei testi sulla base dell'argomento trattato. Si visualizza inoltre l'andamento dei topic nel tempo, cercando di verificare se sono presenti delle dipendenze fra questi e i fatti di cronaca legati al coronavirus.

Come si evince dal contenuto dei capitoli, data l'attualità del fenomeno in esame, non manca la possibilità di effettuare un confronto tra i risultati ottenuti e quanto accaduto nel mondo reale. Tuttavia, siccome il lavoro svolto non è esente da criticità e da limiti, alla fine dell'elaborato questi verranno debitamente commentati e discussi, cercando di capire a cosa sono dovute eventuali distorsioni e quali dati sarebbe stato utile avere per migliorare l'analisi.

Capitolo 1

Collezione dei dati & Pre-processing

In questo primo capitolo vedremo come si possono estrarre tweet dal web tramite due diverse metodologie, i cui risultati saranno integrati per ottenere il dataset finale. In particolare, verranno prima sfruttate le API fornite da Twitter per un'operazione di tweet mining, mentre in un secondo momento si procederà ad ampliare il numero di osservazioni trasformando alcuni identificativi di tweet (già classificati in lingua italiana) grazie al software Hydrator. Dopo aver descritto e reso compatibili le variabili di interesse, si procederà con l'unione dei dataset ottenuti e con il pre-processing dei testi per eliminare tutto quanto andrebbe ad inficiare l'analisi. Tutte le operazioni di questo capitolo, e più in generale di tutta la tesi, sono state eseguite con il software statistico R; in questo capitolo si farà utilizzo in particolare della libreria `TextWiller` [3].

1.1 Estrazione dei tweet mediante API

1.1.1 Le API di Twitter

Data la natura fondamentalmente pubblica di Twitter, che infatti permette la lettura e la condivisione dei messaggi anche di utenti non connessi alla propria rete di amici, vengono messe a disposizione degli sviluppatori, delle società e degli utenti delle API (Application Programming Interfaces, ovvero le interfacce di programmazione delle applicazioni) che consentono l'accesso ai dati del social network. Le API nascono per permettere lo sviluppo di software che si integri con Twitter, ad esempio una soluzione che aiuti una società a rispondere ai feedback dei clienti su Twitter.

In generale, possiamo dire che le API sono il modo in cui i software “parlano” tra

di loro per richiedere e fornire informazioni. Ciò avviene consentendo a un'applicazione di chiamare un endpoint: un indirizzo che corrisponde a un tipo specifico di informazione (di regola, gli endpoint sono univoci come i numeri di telefono). Le API di Twitter includono un'ampia gamma di endpoint, che rientrano in cinque gruppi principali, fra i quali troviamo ad esempio l'endpoint dei messaggi diretti (privati) scambiati fra gli utenti o quello degli annunci pubblicitari. Per accedere all'endpoint di nostro interesse, ossia quello relativo ai tweet pubblici, è innanzitutto necessario registrare un'applicazione collegata ad un account sviluppatore, grazie alla quale avremo a disposizione le credenziali per accedere al servizio di streaming API.

Si tenga presente che non tutti gli endpoint sono uguali: un account sviluppatore di tipo Standard dà accesso ai messaggi registrati al massimo 7 giorni prima della richiesta di retrieval, prevede un tetto massimo di tweet estraibili, non garantisce la cosiddetta data quality (fondamentale per avere risultati in linea con quanto richiesto) e restituisce solo informazioni basilari, quali la data di creazione o l'identificativo unico del messaggio. Per una ricerca più approfondita, che permetta di andare anche fino a 30 giorni indietro nel tempo senza limitazioni, sono necessari account di tipo Premium o Enterprise. Questa non è una considerazione di poco conto: data l'elevata mole di tweet collegati al virus, con un account Standard come nel nostro caso, la portata viene superata nel giro di poche ore, causando così l'esigenza di scaricare i dati giornalmente. Inoltre, stante le limitazioni imposte sui parametri dei tweet, qualora si volesse integrarli con dataset ottenuti in un altro modo, bisognerebbe verificarne la coerenza (ovvero rendere i dati collegabili). Utilizzare le API di Twitter si traduce nell'esecuzione di richieste HTTP che restituiscono i risultati in un formato strutturato (JSON nel nostro caso). A seconda dell'account che si possiede, sono disponibili diversi metodi di tipo POST per eseguire queste richieste (ognuno è uno specifico endpoint). I metodi principali sono:

- il metodo POST counts, grazie al quale ci viene restituito il volume di dati inerente la query inserita;
- il metodo POST statuses/filter, con cui è possibile ricercare tweet contenenti una parola chiave o provenienti da un determinato luogo (messaggio geolocalizzato).

Tutti i metodi richiedono l'autenticazione mediante il protocollo OAuth, che si effettua con le credenziali ottenute in precedenza.

1.1.2 Tweet mining su R

Le Application Programming Interface di Twitter sono integrate nella maggior parte dei linguaggi di programmazione. Nel nostro caso sono stati utilizzati i pacchetti `streamR` (per accedere al servizio di streaming API) e `ROAuth` (per l'autenticazione).

Quest'ultima avviene tramite le credenziali di accesso a Twitter, ottenute una volta che il nostro account da sviluppatore è stato accettato:

```
1 setup_twitter_oauth(consumerKey, consumerSecret, accessToken,
  accessTokenSecret)
```

Dove:

- *Consumer key*: è la chiave API fornita da Twitter che identifica lo sviluppatore;
- *Consumer password*: è la password dello sviluppatore richiesta per accedere ai servizi offerti;
- *Access token e Access token secret*: definiscono i privilegi di accesso dello sviluppatore sulle risorse di Twitter.

La funzione `searchTwitter()` del package `streamR` ci permette di aprire una connessione all'endpoint `POST statuses/filter` per il recupero dei tweet. Essa prevede in input tre parametri: parola chiave, numero di tweet da estrarre e lingua originale del messaggio. Dal momento che per un account di tipo Standard non è garantita la data quality, spesso quest'ultimo parametro non risulta discriminante al 100%: come vedremo più avanti nell'analisi sono infatti presenti diversi tweet di altre nazionalità. Nel nostro caso si sono ricercati ogni giorno 2000 tweet in lingua italiana con la parola chiave "coronavirus", il cui hashtag è il più diffuso a livello internazionale. I risultati sono stati di volta in volta copiati sul medesimo file csv, in modo tale da avere al termine della fase di tweet mining un dataset pronto per essere analizzato. I dati recuperati comprendono, oltre al campo testuale del tweet, molte informazioni aggiuntive. Vediamo i campi più significativi:

- *id* (double): intero che identifica il tweet;
- *favorite_count* (integer): numero di volte che il messaggio è stato scelto come preferito;
- *created_at* (character): data di creazione del messaggio espresso in UTC (formato di tipo YYYY-MM-DD hh:mm:ss);
- *is_retweet* (logical): booleano (TRUE o FALSE) che identifica se è il messaggio è un retweet;
- *text* (character): testo del tweet, di massimo 280 caratteri;
- *retweet_count* (integer): numero di volte che il messaggio è stato "retweettato";
- *screen_name* (character): nome utente di chi ha postato il messaggio;

- *latitude, longitude* (double): coordinate geografiche del luogo da cui il messaggio è stato postato.

Il file csv generato in questo modo contiene in totale da 67.476 tweet raccolti tra il 15-03-2020 e il 12-05-2020. Il dataset è composto da 16 variabili, incluse quelle descritte in precedenza. Siccome il campo *created_at* è già in formato UTC non sono necessarie particolari operazioni per rendere il dataset più digeribile per l'analisi. La fase di pre-processing sul campo *text* verrà eseguita una volta combinato questo dataset con quello ottenuto tramite le operazioni descritte nella sezione successiva.

1.2 Estensione del dataset con Hydrator

1.2.1 Il processo di hydrating

Data la mole ragguardevole di tweet inerenti la pandemia postati ogni giorno, e considerando che l'estrazione mediante API è iniziata quasi un mese dopo la scoperta del primo focolaio in Italia, la quale ha generato particolare scandalo sul web, per avere un'analisi quanto più completa e inclusiva, è stato necessario ampliare il dataset iniziale aggiungendo osservazioni che si riferissero soprattutto al periodo che intercorre tra la seconda metà di febbraio e la prima di marzo. Fortunatamente, come già accennato nell'introduzione, la pandemia ha innescato una risposta vigorosa da parte di tutta la comunità scientifica. Tra le varie risorse che si possono trovare in rete non mancano collezioni di tweet collegati al coronavirus adibiti alla ricerca. Al fine di alleggerire i dataset, vengono pubblicati online solo gli identificativi numerici dei tweet (di 18 cifre), motivo per cui è necessaria la loro trasformazione in un file contenente tutti i campi che li caratterizzano, tecnica che viene comunemente denominata con il termine *hydrating*. Per svolgere questa operazione esistono diversi tool: si può procedere direttamente su R o su Python, oppure utilizzare un programma open-source come Hydrator.

Nel nostro caso si è deciso di percorrere la seconda strada: per processare i dati su Hydrator è sufficiente collegare il proprio account sviluppatore, caricare un file txt con all'interno esclusivamente gli id dei tweet e procedere con l'"hydration", al termine della quale verrà prodotto un file JSON contenente i campi associati ad ogni id. Questo file può essere in seguito facilmente trasformato in formato csv.

In genere vengono elaborati circa 100 tweet al secondo, il che rende, in particolare nel nostro caso, il procedimento alquanto lungo e macchinoso. Si tenga presente che nella maggior parte dei repository sono presenti diversi file per ogni giorno di estrazione, ognuno dei quali contiene decine di migliaia di id che raramente sono già suddivisi per lingua di origine (si veda [4]). Un approccio iniziale su questo tipo di risorse portava quindi ad avere, dopo aver aspettato anche più di 20 minuti, poche centinaia di tweet in lingua italiana. Ancora una volta, allo scopo di velocizzare l'operazione, è stato fondamentale l'apporto della comunità scientifica: sono

stati resi disponibili dei tweet già classificati per lingua, precisazione che riduce di gran lunga i tempi dell'intero procedimento (disponibili qui [5]). Difatti, una volta selezionati gli id di nostro interesse, riportati appositamente in un nuovo file txt, dovremo estrapolare tramite Hydrator solo i campi dei tweet selezionati. Come si può notare nella figura allegata, dalle decine di migliaia di tweet precedenti, si passa così a poche migliaia, delle quali peraltro viene trasformata solo una percentuale di poco superiore al 50%, dato che molti dei tweet, riferiti a mesi addietro, possono essere nel frattempo stati cancellati o non essere reperibili.


 Datasets	Add	Settings
Mar_01_2020_coronavirus		
2,271 of 2,271 ids read (1,164)	CSV	Delete
Feb_29_2020_coronavirus		
1,833 of 1,833 ids read (969)	CSV	Delete
Feb_28_2020_coronavirus		
1,616 of 1,616 ids read (797)	CSV	Delete
Feb_27_2020_coronavirus		
2,611 of 2,611 ids read (1,336)	CSV	Delete
Feb_26_2020_coronavirus		
3,600 of 3,600 ids read (1,817)	CSV	Delete

Figura 1.1: Alcuni dei dataset estratti

1.2.2 Descrizione delle variabili e creazione del corpus finale

Una volta terminata la fase di hydrating, effettuata singolarmente per ogni giorno in cui sono stati raccolti gli id, i file csv sono stati integrati in nuovo dataset che contiene in totale circa 230.000 tweet raccolti tra il 14-02-2020 e il 23-04-2020, sui quali sono state rilevate 34 variabili.

Vediamo le principali differenze tra questo dataset e il precedente, in modo da valutare la loro compatibilità.

Innanzitutto, notiamo che la variabile *is_retweet* è sostituita dal campo *retweet_id* (double), dal quale possiamo trarre la medesima informazione fornita dalla variabile booleana (è uguale a NA per i tweet originali), con l'aggiunta, nel caso si tratti di

un retweet, dell'id relativo al messaggio retweettato, che risulterà essere di particolare rilievo per la nostra analisi.

Per quanto riguarda i campi *id*, *favorite_count*, *text*, *retweet_count*, *screen_name* non sono da segnalare differenze rispetto alla descrizione precedente.

Il campo *created_at* al contrario presenta delle criticità: anziché essere già in formato UTC si presenta, ad esempio, in un formato character di tipo:

created_at
Wed Feb 19 06:09:20 +0000 2020
Sun Mar 22 09:55:35 +0000 2020
Sat Apr 11 02:15:44 +0000 2020

Dobbiamo quindi procedere trasformando la data di pubblicazione del messaggio in formato UTC, rendendolo compatibile con il dataset creato mediante le API. Spezzando la stringa, convertendo i mesi in numeri, selezionando e riordinando i token di nostro interesse, per l'esempio precedente si ottiene:

created_at
2020-02-19 06:09:20
2020-03-22 09:55:35
2020-04-11 02:15:44

Ulteriori campi di interesse non presenti precedentemente sono:

- *lang* (character): lingua d'origine del messaggio;
- *user_description* (character): stringa attraverso la quale l'utente descrive il proprio account;
- *user_location* (character): città/luogo d'origine dell'utente;
- *coordinates* (character): latitudine e longitudine (congiuntamente) del luogo da cui il messaggio è stato postato.

Dopo aver descritto nel dettaglio come si sono ottenuti i dataset, possiamo procedere creando il corpus finale, ovvero la collezione di testi impiegata nell'analisi.

A questo proposito, proseguiamo su due binari separati.

Per quanto riguarda i tweet originali, combiniamo quelli provenienti da entrambi i dataset e manteniamo tutte le variabili in comune; queste sono: *id*, *favorite_count*, *text*, *retweet_count*, *screen_name*, *created_at*, *latitude*, *longitude* (queste ultime considerate in formato double separatamente).

Dal dataset creato mediante le API estraiamo 36.104 tweet originali e li uniamo ai 71.350 provenienti dal secondo dataset, selezionati tra i messaggi in lingua italiana ponendo la variabile *retweet_id* = NA.

Ricapitolando, abbiamo infine 107.454 tweet originali per i quali sono state rilevate 8 variabili. Come ultima operazione, per evitare doppioni, selezioniamo solo gli identificativi che compaiono una sola volta: le osservazioni si riducono a 104.123. Queste osservazioni sono relative ad un periodo che intercorre tra il 14-02-2020 e il 12-05-2020, che possiamo considerare significativo per valutare le principali fasi della pandemia in Italia: dalla scoperta del primo focolaio alla riapertura con la fase 2.

Per quanto riguarda i retweet, verranno considerati solo quelli provenienti dal dataset creato con Hydrator, in quanto, come accennato in precedenza, contengono fra i vari campi l'id del messaggio che è stato retweettato, che risulterà di fondamentale importanza nel capitolo 3.

Ripetendo le operazioni di prima, ma stavolta orientandoci verso i retweet, ossia selezionando le righe che hanno il campo *retweet_id* diverso da NA, otteniamo un dataset finale contenente 137.283 osservazioni, per le quali sono state rilevate 34 variabili. Essendo provenienti solo dal secondo dataset, le osservazioni sono state estratte tra il 14-02-2020 e il 23-04-2020.

Entrambi i dataset non presentano *missing values* e sono già organizzati in formato *tidy*, ovvero: ogni riga corrisponde ad un'osservazione e ogni colonna ad una variabile rilevata.

Come considerazione finale, si tenga presente che questi messaggi non sono altro che una piccolissima percentuale della totalità di tweet pubblicati in questi mesi, infatti per avere un'analisi più accurata bisognerebbe considerare un numero di osservazioni sull'ordine del milione. Al netto di ciò, sono state prese tutte le precauzioni del caso (suddivisione in tweet e retweet, eliminazione di messaggi duplicati, pre-processing dei testi nel paragrafo successivo) affinché il campione ottenuto risulti il più possibile rappresentativo della realtà di riferimento.

1.3 Pre-processing dei testi

Terminata l'estrazione dei dati e la costruzione del corpus, procediamo con le operazioni di pre-processing sul campo *text*, fondamentali per rendere i messaggi agevolmente processabili dai vari algoritmi. Per fare ciò bisogna eseguire una serie di operazioni di carattere morfosintattico sui testi, in assenza delle quali si potrebbero ottenere risultati altamente insoddisfacenti. Ci concentreremo per il momento sul dataset dei tweet originali, che risulterà più centrale per l'analisi, nonostante le operazioni descritte siano state eseguite allo stesso modo anche per quanto riguarda i retweet.

1.3.1 La libreria TextWiller

Un primo approccio alla pulizia dei testi ha visto protagonista la libreria **TextWiller**, sviluppata appositamente per analizzare testi in lingua italiana, che attraverso la

funzione `normalizzaTesti()` ci permette di implementare le principali operazioni di pre-processing, fra le quali troviamo:

- *lowercasing*: conversione in minuscolo di tutti i caratteri che compongono il testo, per fare in modo che parole uguali vengano riconosciute come tali indipendentemente dai caratteri utilizzati per rappresentarle;
- *stopword filtering*: rimozione di una serie di termini che non contengono alcun significato semantico. Fra questi sono inclusi articoli, preposizioni, congiunzioni e verbi più comuni. La lista utilizzata (*itastopwords*) è presentata in appendice;
- *rimozione della punteggiatura*: vengono rimossi caratteri quali `(; . , : \n \)`;
- *normalizzazione URL*: tutti i link a riferimenti esterni, anch'essi non rilevanti dal punto di vista semantico, vengono modificati in una stringa di default;
- *normalizzazioni emoji*: analogamente, le emoji vengono trasformate in apposite stringhe, per esempio i cuori sono catalogati come "emotelove";
- *normalizzazione dei caratteri*: rimozione di caratteri superflui al termine delle parole.

Vediamo un primo esempio di applicazione della funzione, prima e dopo la normalizzazione:

```

1 Coronavirus, il mondo si tinge di tricolore: da Sarajevo alla
   Palestina, \"siamo tutti italiani\" https://t.co/
   SXTTrzW5Dc di @repubblica
2
3 coronavirus mondo tinge tricolore sarajevo palestina italiani
   wwwurlwww @repubblica

```

Come si può notare, nonostante il messaggio non presenti particolari problemi, sono necessari ancora alcuni aggiustamenti: se estendiamo il dizionario delle stopwords includendo tutte i termini legati al virus, quali "coronavirus", "covid19" "coronavirustaly" (superflui in questo caso), e la stringa legata alla normalizzazione degli URL otteniamo un tweet pronto per essere analizzato. Per quanto riguarda gli hashtag (#) e le mention (@), il loro trattamento risulta abbastanza delicato: da un lato illustreremo come si è proceduto per la loro rimozione nella sezione successiva, in modo tale da non confondere parole uguali solo perché precedute da un simbolo, dall'altro si tenga conto che questa è stata realizzata solo dopo aver eseguito alcune analisi descrittive (capitolo 2) che si prestano per questo tipo di etichette.

1.3.2 Ulteriori sviluppi

Al di là di queste considerazioni, restano ancora parecchie cose da sistemare per evitare di incorrere in distorsioni al momento del passaggio all'analisi semantica. Vediamo un esempio più ostico da normalizzare:

```
1 Coronavirus: â€œDona la potenza del tuo computer e troveremo
   la curaâ€\u009d [di JAIME D'ALESSANDRO] [aggiornamento
   delle 23:12] https://t.co/yXwpoBRw28
2
3 â€œdona potenza computer troveremo curaâ€\u009d jaime
   alessandro aggiornamento 23 12
```

In questo caso permangono delle problematiche che la libreria non è in grado di risolvere: in diversi messaggi sono presenti caratteri non ASCII (riferiti solitamente ad ideogrammi cinesi o a complicazioni nel processo di hydrating) e caratteri esadecimali (riferiti a emoji che non sono state convertite da `normalizzaTesti()`). Inoltre, permangono nei messaggi i numeri, alcuni caratteri speciali e di controllo, gli spazi superflui, gli hashtag e le mention. Procediamo definendo una funzione che attraverso alcune applicazioni del metodo `gsub()` ci permette di ovviare ad ognuna di queste problematiche.

```
1 textprocessing <- function(x)
2 {gsub('[0-9]+', '', x) #rimozione numeri
3   gsub('#\\S+', '', x) #rimozione hashtag
4   gsub('@\\S+', '', x) #rimozione mentions
5   gsub('[:cntrl:]', '', x) #rimozione control characters
6   gsub("\\d", '', x) #e caratteri speciali
7   gsub("^[:space:]*", "", x) #rimozione spazi all'inizio
8   gsub("[:space:]*$", "", x) #rimozione spazi alla fine
9   gsub(" *\\b[:alpha:]{1,2}\\b *", " ", x) #rimozione
   parole con meno di tre lettere
10 }
```

Utilizziamo in seguito la funzione di default di R `iconv()`¹ per passare dal formato *latin1* a quello ASCII (rimuovendo ulteriori caratteri) e come ultimo passaggio, poiché la rimozione dei numeri e dei caratteri speciali ha causato il troncamento di diverse parole, eliminiamo tutte quelle che contengono meno di tre lettere. Seguendo questa logica, eliminiamo anche i messaggi che si sono trasformati in stringhe

¹<https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/iconv>

vuote: il totale dei tweet scende così di 1000 unità. Dobbiamo inoltre estendere ancora una volta il dizionario delle stopwords includendo i residui delle rappresentazioni esadecimali (con tre lettere) che andrebbero ad "inquinare" i dati, fra questi troviamo: *fef*, *ffb*, *ffa*.

Il risultato, per un messaggio ancora più complicato è il seguente:

```
1 #coronavirusöÿ\u008d\u008f\n\nLa curva dei contagi Ã
   stabilmente sotto il 2%! https://t.co/MoWdmDpMLM
2
3 curva contagi stabilmente sotto
```

Terminata la prima fase di pre-processing, sarebbe naturale procedere con un altro dei pilastri dell'analisi morfosintattica: il cosiddetto stemming, ovvero il processo di riduzione della forma flessa di una parola alla sua forma radice, detta tema. Sfortunatamente, l'unico stemmer destinato alla lingua italiana, disponibile tramite la libreria **Snowball**, non è molto preciso e porta ad avere difficoltà nell'interpretazione di alcuni risultati, motivo per cui, per il momento, ci limiteremo a considerare i termini originali, al netto della perdita di associazione semantica tra singolari-plurali, maschili-femminili e diversi tempi verbali.

1.4 Operazioni preliminari

Proseguiamo con altre operazioni allo scopo di migliorare ulteriormente i dati a nostra disposizione.

Nonostante i tweet estratti siano già stati catalogati in italiano, spesso la caratterizzazione del linguaggio in Twitter non è precisa, assicuriamoci dunque di estrarre i soli tweet in lingua italiana, avvalendoci della funzione `textcat()` presente nella stessa libreria [6]. Questo è fondamentale dato che le analisi testuali che andremo ad effettuare sono incentrate in particolare sulla lingua italiana, e risentirebbero non poco della presenza di termini stranieri. Questa operazione non è stata replicata per i retweet, poiché l'analisi testuale per questi dati sarà meno approfondita e che in molti casi vengono retweettati messaggi provenienti dall'estero vista la loro ampia condivisibilità. Tutto ciò verrà comunque discusso più nel dettaglio nel capitolo 3. La tabella dei conteggi relativa (per i tweet originali) è la 1.1.

Lingua	Conteggi	Lingua	Conteggi
Albanian	13	Italian	95.568
Bosnian	86	Latin	1899
Croatian	83	Norwegian	58
Czech	10	Polish	29
Danish	54	Portoguese	807
Deutsch	116	Romanian	520
Dutch	62	Serbian	39
English	869	Slovak	20
Finnish	37	Slovenian	73
French	166	Spanish	974
Hungarian	27	Swedish	88

Tabella 1.1: Conteggi dei tweet per nazionalità

Mantenendo i tweet appartenenti alle categorie Italian e Latin le nostre osservazioni si riducono ulteriormente, scendendo ad un totale di 97.467.

Prima di procedere con l'analisi esplorativa, aggiungiamo alcune variabili che potranno tornarci utili in seguito. A tal proposito, spezziamo la data di creazione *created_at* del messaggio nelle sue due componenti:

- *created_day*: data effettiva espressa in YY-MM-DD, che ci permetterà di raggruppare i messaggi per giorno;
- *created_hour*: tempo di creazione espresso in hh:mm:ss.

Riepilogando quanto eseguito finora, partiamo da una base di 97.467 tweet originali, per i quali abbiamo eseguito le principali operazioni di pre-processing sul campo *text*, eliminando tutto quanto avrebbe inficiato l'analisi da un punto di vista semantico. Con le ultime aggiunte abbiamo un totale di 10 campi rilevati per ciascuna osservazione, questi sono: *id*, *text*, *created_at*, *created_day*, *created_hour*, *screen_name*, *favorite_count*, *retweet_count*, *latitude*, *longitude*. Questo è il dataset centrale per il lavoro svolto: verrà utilizzato per le analisi presentate nei capitoli 2, 4 e 5.

Per quanto riguarda i retweet, le medesime operazioni di pre-processing hanno portato le 137.283 osservazioni originali a ridursi ad un totale di 135.089, per le quali abbiamo mantenuto le 34 variabili originali. Ci concentreremo su questo secondo dataset nel capitolo 3.

Capitolo 2

Analisi Esplorativa

In questo secondo capitolo vedremo come utilizzare il dataset ottenuto per trarne informazioni attraverso alcune semplici analisi descrittive. In particolare saranno presentati: i conteggi dei tweet per settimana, una valutazione delle keywords più rilevanti relativamente al tema trattato, gli user più citati e quelli più attivi, gli hashtag e i termini più utilizzati.

Procederemo con la ricerca delle parole che co-occorrono più frequentemente e infine ci concentreremo su quali termini risaltano nel corpus rispetto alla term frequency - inverse document frequency.

La maggior parte delle analisi eseguite in questo capitolo fanno riferimento alle librerie all'interno dei pacchetti `tidyverse` [7] e `tidytext` [8], fra le quali `dplyr` [9], `tidyr` [10], `ggplot2` [11], ecc.

Sfruttando il formato *tidy* del dataset è possibile combinare i metodi forniti da queste librerie al fine di ottenere i risultati mostrati.

2.1 Premesse

Introduciamo prima di tutto alcuni concetti teorici che stanno alla base di quanto è stato svolto in questo capitolo e nei successivi.

2.1.1 Tokenization

Per tokenizzazione si intende il processo attraverso il quale una sequenza di caratteri viene suddivisa nei token che la compongono, ossia parti del testo più corte e più semplici da processare. A seconda dell'applicazione considerata possono essere considerati token singole parole, intere frasi o parte di esse. Questo dipende da che delimitatori si sono considerati, i quali sono solitamente appartenenti alla cosiddetta punteggiatura “forte”, ad esempio terminatori di riga (`\n`), di frase(`.`)

o di periodo (, ; :). Poiché nel nostro caso è stata implementata attraverso il pre-processing dei testi la rimozione della punteggiatura, l'applicazione della tokenization ci porta direttamente ad ottenere le singole parole che formano i messaggi, spezzando dunque le frasi in base agli spazi bianchi. A tale scopo utilizziamo la funzione `unnest_tokens()` della libreria `dplyr` direttamente sul campo `text`. In questo modo facilitiamo di gran lunga l'esecuzione dei conteggi presentati dei prossimi paragrafi.

2.1.2 La Document Term Matrix

La Document Term Matrix descrive le frequenze assolute dei termini presenti in una collezione di documenti: le righe corrispondono ai tweet e le colonne alle parole presenti al loro interno. La matrice ottenuta, contenente le *term frequencies* è di dimensioni 97.467×62.098 . Essa ha un livello di *sparsity* pari a 99.9%: questo valore è stato ottenuto rapportando il numero di $tf_{i,j} = 0$ ($6.050.808.792$) sul totale ($97.467 \times 62.098 = 6.052.505.766$). Il comando `DocumentTermMatrix()` necessita di argomenti di tipo vector, quindi va creato il vettore di testi attraverso il metodo `Corpus()`, entrambi utilizzabili grazie alla libreria `tm` [12]. Vediamo una sezione di esempio presa dalla matrice che sarà utilizzata in seguito (tabella 2.1).

	annuncia	aperti	emotelove	giorno	italiani	mascherine	senza...
150	1	0	2	1	0	0	0
151	0	1	0	0	1	0	0
152	0	0	1	0	0	0	0
153	0	0	0	0	0	0	1

Tabella 2.1: Estratto di DTM

2.2 Prime analisi descrittive

I tweet appena ottenuti possono essere sfruttati per qualche analisi preliminare prettamente descrittiva: vediamo ad esempio le distribuzioni giornaliere e orarie, i messaggi più retweettati, gli hashtag e gli users più citati, le keyword più utilizzate. Mostriamo innanzitutto i conteggi dei tweet per settimana nel periodo considerato: dal 14/02/2020 al 12/05/2020.

Settimana	Conteggi	Settimana	Conteggi
14/02-19/02	331	02/04-08/04	12.685
20/02-26/02	10.058	09/04-15/04	12.817
27/02-04/03	4218	16/04-22/04	14.970
05/03-11/03	5938	23/04-29/04	4619
12/03-18/03	4609	30/04-06/05	3376
19/03-25/03	9360	07/05-12/05	3883
26/03-01/04	10.603		

Tabella 2.2: Conteggi dei tweet per settimana

L'unica considerazione certa che possiamo fare su questi numeri è che l'impen-
nata dal 20/02 in poi è dovuta alla scoperta del primo focolaio in Lombardia. Se
prima i tweet si concentravano su quanto accadeva relativamente lontano da noi
(in Cina) e il coronavirus era ancora una tematica trascurabile, con i primi contagi
sono scattati i messaggi di sdegno e paura sul web, giustificati dal fatto che l'Italia
è stato il primo paese d'Europa ad essere duramente colpito dalla pandemia. Per
quanto riguarda le settimane successive, è difficile valutare se una maggiore attività
su Twitter sia avvenuta in concomitanza con degli avvenimenti particolari riguar-
danti l'evoluzione della pandemia. Questo perché, come già detto nel primo capitolo,
disponiamo di un campione esiguo di messaggi rispetto alla totalità dei tweet, e per
giunta stiamo considerando solo i messaggi originali e scritti in italiano per ragioni
semantiche; nulla infatti vieta che molti dei messaggi siano retweet dall'estero, si
pensi in particolare a quanto accaduto nello stesso periodo in tutta Europa e ne-
gli Stati Uniti. La maggiore attività tra il 26/03 e il 22/04 è data principalmente
dalla disponibilità di osservazioni provenienti da entrambi i dataset in quel perio-
do, anche se parte dell'aumento potrebbe essere riconducibile all'attenzione di cui
hanno goduto le conferenze stampa del Presidente del Consiglio (la più seguita in
assoluta quella del 10/04¹). Il calo di maggio è invece da attribuire alla mancanza
di id da trasformare mediante Hydrator in quel periodo (gli id trasformati arrivano
fino al 23/04): gli ultimi messaggi sono per questo derivanti solo dal tweet mining
effettuato con le API.

Provando ad eseguire la stessa operazione per le ore in cui i tweet vengono
pubblicati durante la giornata otteniamo dei risultati non molto significativi: circa
il 90% dei messaggi è stato pubblicato tra le 22 e 23 di sera. Non è chiaro se questo
sia dovuto all'effettivo orario di pubblicazione dei tweet estratti o se ci siano stati
problemi nella conversione dei dati che hanno portato all'attribuzione di un orario
di default.

¹<https://www.youtube.com/watch?v=2HtEkxYvelo>

2.2.1 Messaggi più retweettati

Una delle analisi più semplici che possiamo effettuare sul dataset, che ricordiamo è composto da 97.467 osservazione relative a 10 variabili, senza ancora addentrarci nel campo *text*, è visualizzare quali sono stati, fra i tweet a nostra disposizione, quelli maggiormente retweettati. Sfruttando il campo *retweet_count*, ci basta riorganizzare i messaggi in ordine decrescente tramite il comando `arrange()`. Vediamo i primi tre:

screen_name	text	retweet_count
Fedez	Fate girare! Codacons sta spacciando sul loro sito una campagna di raccolta fondi apparentemente contro il coronavirus quando basta cliccare sul banner per scoprire che le donazioni servono a sostenere SOLO loro stessi. Ma è possibile che nessuno intervenga? https://t.co/kIVesszFQy	6541
tragicom24	Coronavirus, un applauso a tutti i carcerati d'Italia, gli unici a rispettare il divieto di uscita imposto dal Governo	2528
MinisteroSalute	Siamo al lavoro per garantire i massimi livelli di sicurezza. Il nostro Servizio sanitario nazionale è attrezzato per fronteggiare l'emergenza #coronavirus e la collaborazione dei cittadini è preziosa. Il decalogo in collaborazione con @istsupsan #coronavirusitalia #covid19 https://t.co/CcUep6zC16	2260

Tabella 2.3: Messaggi con il maggior numero di retweet

Come prevedibile, i messaggi fanno riferimento a tematiche ampiamente condizionali o sono stati pubblicati da personaggi con un alto riscontro mediatico.

2.2.2 Keyword Trend Analysis

Attraverso i comandi `filter()` e `str_detect()` della libreria `dplyr` estraiamo i tweet che contengono le seguenti 12 parole chiave: "mascherine", "positivi", "tamponi", "ospedali", "letti", "medici", "infermieri", "ventilatori", "aiuto", "febbre", "tosse", "contagi", in modo tale da visualizzare quali sono le tematiche che più preoccupano i cittadini su Twitter. Le frequenze assolute dei tweet sono presentate in tabella 2.4.

Keyword	Conteggi	Keyword	Conteggi
contagi	5957	letti	1598
positivi	4508	aiuto	850
ospedali	2415	infermieri	626
mascherine	2205	ventilatori	146
medici	1808	febbre	197
tamponi	1667	tosse	86

Tabella 2.4: Keyword Trend Analysis

Abbiamo un numero maggiore di tweet per i messaggi contenenti il termine "contagi" (e derivati), seguito da "positivi" e "ospedali". Troviamo invece agli ultimi posti i testi riguardanti i principali sintomi della malattia da coronavirus. Questo può essere indice del fatto che le persone, all'interno di questo dataset, discutono più frequentemente delle tematiche collegate alla pandemia (mancanza di DPI, andamento epidemiologico) rispetto ai sintomi e alle condizioni di salute dei malati. Troveremo in seguito (capitolo 5) che diverse delle keyword cercate sono menzionate all'interno di alcuni topic individuati dalla Latent Dirichlet Allocation.

2.2.3 Hashtags & Mentions

Passiamo alla valutazione degli user e degli hashtag più citati, eseguita dopo la tokenizzazione combinando ancora una volta i comandi `filter()` e `str_detect()`, dove quest'ultima ci permette di identificare le parole che cominciano con la chiocciola o con il simbolo dell'hashtag (rispettivamente "`^@"` e "`^#`", prima della loro rimozione). Per visualizzare i conteggi in ordine decrescente utilizziamo la funzione `count()`. Le prime 20 mention fanno riferimento ai canali di stampa, politica e informazione:

Username	Conteggi	Username	Conteggi
@repubblica	2230	@ilpost	183
@LaStampa	728	@GiorgiaMeloni	174
@matteosalvinimi	621	@MinisteroSalute	173
@YouTube	560	@ilmessaggeroit	166
@GiuseppeConteIT	485	@DPCgov	156
@fattoquotidiano	340	@RaiNews	155
@RobertoBurioni	300	@GoogleNews	147
@sole24ore	258	@ChangeItalia	141
@fanpage	224	@robersperanza	140
@Corriere	190	@SkyTG24	128

Tabella 2.5: Top 20 mentions

Per quanto concerne gli hashtag, oltre alle parole chiave relative alla politica e all'informazione, troviamo alcuni termini riferiti alle campagne di sensibilizzazione lanciate sul web e altri collegati alla gestione della pandemia. Relativamente a queste ultime abbiamo filtrato tutti gli hashtag utilizzati in fase di estrazione dei dati (`#coronavirus`), chiaramente presenti in ogni singolo tweet, così come quelli che in generale fanno riferimento al virus (`#coronavirusitaly`, `#coronavirusitalia`, `#covid2019`, `#covid19italia`). Spicca al secondo posto l'hashtag `#iorestoacasa`, decisamente quello che ha risaltato di più a livello mediatico durante le fasi peggiori dell'epidemia.

Hashtag	Conteggi	Hashtag	Conteggi
#italia	1108	#lockdown	407
#iorestoacasa	1043	#milano	402
#conte	934	#pandemia	395
#aprile	922	#notizie	394
#lombardia	788	#cina	377
#quarantena	582	#news	369
#salvini	494	#governo	358
#marzo	490	#pasqua	301
#fase2	465	#mes	291
#ansa	427	#mascherine	274

Tabella 2.6: Top 20 hashtags

Si noti che per eseguire i conteggi in tabella 2.5 non era necessario alcun pre-processing dei testi. Questo perché l'identificativo "@" ha un significato chiaro all'interno di Twitter: viene appunto utilizzato per rispondere ad un altro messaggio tramite le cosiddette mention, inoltre il comando viene riconosciuto solo se si cita uno user registrato, dunque non avremmo incorso alcun rischio eseguendo questa analisi direttamente sul campo *text* originale.

Al contrario, per quanto riguarda gli hashtag, anch'essi caratteri speciali all'interno di Twitter, il pre-processing dei testi è fondamentale per almeno due ragioni: in primo luogo, grazie al *lowercasing*, non si rischia di confondere due hashtag che differiscono solo per una maiuscola/minuscola; secondariamente gli hashtag non sono predefiniti come gli username, il che significa che qualsiasi sequenza di caratteri può essere catalogata come tale, anche se priva di significato.

Restando sugli user, possiamo visualizzare quali all'interno del nostro dataset sono stati più prolifici in termini di tweet: è sufficiente raggruppare i messaggi sulla base della variabile *screen_name* tramite la funzione `group_by()` per ottenere la seguente tabella:

Username	Tweet pubblicati	Username	Tweet pubblicati
infoitinterno	2040	Italia_Notizie	397
zazoomblog	1062	TV7Benevento	343
zazoomnews	1045	bizcommunityit	316
Notiziedi_it	650	infoitsport	310
infoitestero	626	palermo24h	299
infoitsalute	618	sportli26181512	283
bnotizie	586	Adnkronos	282
repubblica	455	LaStampa	255
venti4ore	408	infoiteconomia	251
Affaritaliani	397	paoloigna1	241

Tabella 2.7: User più attivi

Salvo l'ultimo utente, tutti i conteggi si riferiscono ad account di informazione

e notizie, risultato in linea con quanto osservato in precedenza.

Abbiamo tralasciato finora alcune variabili in comune fra i dataset di partenza: per quanto riguarda le coordinate geografiche *longitude* e *latitude* abbiamo un elevato numero di NA che purtroppo non rende significativa la loro valutazione, l'unico riscontro in termini geografici che possiamo ottenere è dato dal campo *user_location* presente nel secondo dataset, che tuttavia tralasciamo per non appesantire troppo questa sezione (si tratterebbe ancora una volta di un conteggio, stavolta sulla base del luogo da cui l'utente si è registrato). Vale lo stesso discorso per *favorite_count*.

2.3 Analisi degli n-grammi

Introduciamo ora un concetto utile per proseguire le analisi, l'*n-gramma*. Un *n-gramma* è una sottosequenza di *n* elementi di una data sequenza. In base all'applicazione, gli elementi in questione possono essere fonemi, sillabe, lettere, parole, ecc. Concentriamoci sulla ricerca di parole e di *n-grammi* più frequenti, allo scopo di individuare quali sono i termini più utilizzati e quelli che co-occorrono più frequentemente.

Il risultato della tokenizzazione effettuata in precedenza sui testi già processati ci consente di valutare gli unigrammi (singole parole). Visualizziamo graficamente le parole più frequenti tramite una wordcloud, utilizzando la funzione del pacchetto `wordcloud2` [13]. Con essa si intende una rappresentazione visiva delle etichette associate ai termini. Il peso delle etichette (reso con caratteri di dimensioni diverse) è inteso esclusivamente come frequenza di utilizzo; più grande il carattere, maggiore la frequenza della parola. Dall'immagine si può notare l'efficacia del pre-processing dei testi effettuato in precedenza: la rimozione degli hashtag e delle mention permette che questi non vengano confusi con le parole, non sono presenti stopwords o parole prive di significato e infine, avendo rimosso tutti i caratteri non ASCII, non abbiamo simboli che "inquinano" i termini.



Figura 2.1: Wordcloud degli unigrammi

Prendiamo ora in considerazione bigrammi e trigrammi, ordinati per frequenza. Per ottenerli è sufficiente modificare il parametro `n` della funzione `unnest_tokens()` (setteandolo a 2 o 3) oppure ci si può appoggiare al metodo `textcnt()` della libreria `tau` [14]. È possibile ovviamente considerare lunghezze maggiori, ma trattandosi di tweet, quindi di frasi molto brevi, aumentare le parole porta a risultati di poco interesse.

Bigramma	Conteggi	Bigramma	Conteggi
nuovi casi	973	oltre mila	300
terapia intensiva	617	zona rossa	299
protezione civile	565	primo caso	296
casi positivi	558	mila morti	295
ultime notizie	406	fino emotelove	287
non solo	4001	prima linea	268
emilia romagna	343	perché non	261
nuovi contagi	304	fatto quotidiano	259
ultime ore	303	task force	258
medici infermieri	301	stati uniti	255

Tabella 2.8: Bigrammi individuati

Una volta individuati i bigrammi più rilevanti, possiamo sfruttarli nell'analisi sostituendoli nei tweet come parola unica grazie al comando `gsub()`: in questo modo "emilia romagna" diventerà "emilia_romagna". Evitiamo dunque che i conteggi delle parole più usate aumentino vertiginosamente, facendo diminuire inevitabilmente l'importanza di altre.

Trigramma	Conteggi	Trigramma	Conteggi
nuovi casi positivi	140	ricoverati terapia intensiva	71
fino emotelove maggio	137	nuovo articolo emotebad	67
live ultime notizie	110	ultima ora ansa	66
von der leyn	100	pio albergo trivulzio	65
istituto superiore sanità	77	decreto cura italia	54
bollettino protezione civile	76	nessun nuovo caso	53

Tabella 2.9: Trigrammi individuati

2.4 Termini più importanti per tf-idf

Introduciamo una nuova nozione che ci tornerà utile in questo paragrafo: la tf-idf. Con questa sigla si intende la term frequency - inverse document frequency, ovvero una funzione utilizzata per misurare l'importanza di un termine rispetto ad un documento o ad una collezione di documenti, la quale può essere calcolata a livello dell'n-gramma. Tale funzione aumenta proporzionalmente al numero di volte che il termine è contenuto nel documento (in questo caso nel tweet), ma cresce in maniera inversamente proporzionale con la frequenza del termine nella collezione. L'idea alla base di questo comportamento è di dare più importanza ai termini che compaiono nel documento, ma che in generale sono poco frequenti.

La tf-idf è stata ampiamente utilizzata per ridurre la dimensionalità di tweet utilizzati per problemi di classificazione [15] e analisi del sentiment [16].

La funzione può essere scomposta in due fattori:

- il primo fattore della funzione è il numero dei termini presenti nel documento (elementi della Document Term Matrix). In genere questo numero viene diviso per la lunghezza del documento stesso per evitare che siano privilegiati i documenti più lunghi.

$$tf_{i,j} = \frac{n_{i,j}}{|d_j|}$$

dove $n_{i,j}$ è il numero di occorrenze del termine i nel documento j , mentre il denominatore $|d_j|$ è semplicemente la dimensione, espressa in numero di termini, del documento j ;

- il secondo fattore della funzione indica l'importanza generale del termine i nella collezione:

$$idf_i = \log \frac{|D|}{|\{d \in D : i \in d\}|}$$

dove $|D|$ è il numero di documenti nella collezione ($|\cdot|$ indica la cardinalità di un insieme), mentre il denominatore è il numero di documenti che contengono il termine i .

Abbiamo quindi che:

$$tf-idf_{i,j} = tf_{i,j} \times idf_i$$

È interessante vedere quali sono state le parole più importanti per tf-idf suddividendo temporalmente il nostro dataset. Consideriamo i messaggi in base ai mesi in cui sono stati pubblicati:

- *feb*: osservazioni dal 14/02 al 7/03: dal primo focolaio di Codogno alla dichiarazione del Nord Italia come zona rossa;
- *mar*: osservazioni dal 08/03 al 1/04: fase più buia della pandemia;
- *apr*: osservazioni dal 02/04 al 24/04: annullamento delle festività pasquali e prolungamento della quarantena fino a maggio;
- *mag*: osservazioni dal 25/04 al 12/05: prime riaperture dopo la festa della Liberazione e inizio della fase 2;

Dopo aver creato una variabile categoriale (*month*) per specificare le quattro suddivisioni, utilizziamo la funzione `bind_tf_idf()` per attribuire ad ogni singolo token la relativa tf-idf. Riordiniamo poi queste in ordine decrescente suddividendo in base a *month*.

I risultati ottenuti sono i seguenti:

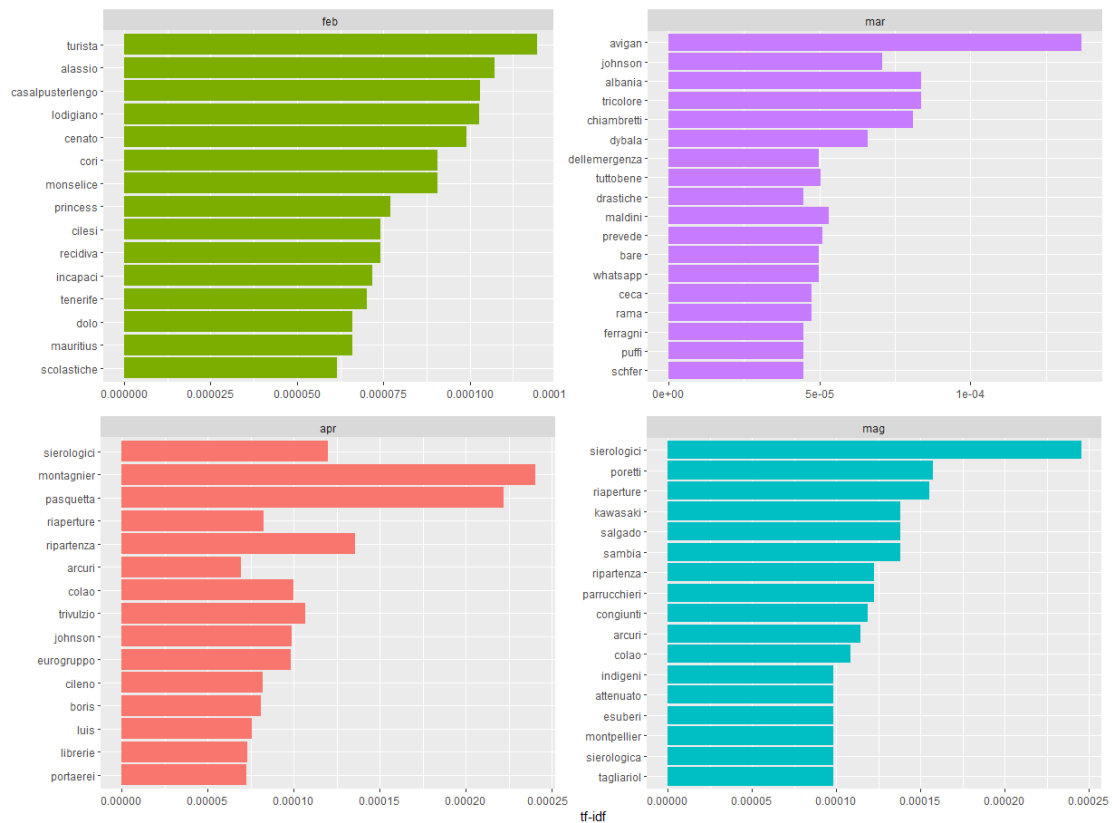


Figura 2.2: Parole più importanti rispetto alla tf-idf

Dai grafici allegati si può notare come la suddivisione temporale porti a sua volta ad una distinzione semantica tra le parole che risaltano per tf-idf. Durante il mese di febbraio i termini si riferiscono principalmente ai primi focolai in Italia (Alassio, Casalpusterlengo, Dolo), in particolare quello di Codogno (lodigiano, cenato), e ai concittadini rimasti bloccati all'estero (Tenerife, Mauritius, Princess). A marzo abbiamo le prime menzioni dei personaggi pubblici che sono risultati positivi al virus (Dybala, Johnson) e messaggi legati alla drammaticità di quelle settimane (Albania, tricolore, bare). Per quanto riguarda aprile e maggio acquistano sempre più importanza i termini relativi alla ripartenza (parrucchieri, riaperture, librerie) e permangono quelli relativi ai contagi fra personaggi famosi (cileno, Poretti, Sambia, Salgado). Troviamo inoltre alcune personalità politiche legate alla gestione dell'emergenza (Colao, Arcuri). Sono presenti infine da marzo in poi riferimenti al virus stesso: discussioni sulla sua origine (Montagnier), efficacia dei test per individuarlo (sierologici), eventuali farmaci contro la malattia che genera (tagliariol, avigan).

Concludiamo qui questo breve excursus che ci ha permesso di investigare in

superficie il dataset a disposizione. Nei capitoli che seguono la nostra conoscenza sull'argomento viene ulteriormente approfondita mediante l'applicazione di diverse tecniche afferenti al Natural Language Processing.

Capitolo 3

Analisi dei Retweet

L'attività di "retweeting" è una particolare funzionalità di Twitter attraverso la quale gli utenti possono pubblicare sul proprio profilo messaggi scritti da altri account, veicolandone in questo modo la condivisione. I messaggi con un alto numero di retweet, ad esempio quelli presentati nel capitolo 2, sono spesso rappresentativi di un argomento che ha generato un singolare interesse fra gli utenti nell'ambiente altamente competitivo di Twitter, e potrebbero essere indice di quali tematiche hanno risonanza nell'opinione pubblica.

Se le analisi del primo capitolo si focalizzavano sui tweet originali, in questa sezione ci concentreremo sul secondo dataset che abbiamo ottenuto precedentemente, ovvero quello contenente i soli retweet. Iniziamo con una valutazione del time-to-retweet, chiamato anche "response time" [2], statistica utilizzata per misurare la velocità di propagazione delle informazioni attraverso i tweet.

In seconda analisi, esploriamo il contenuto dei messaggi che sono stati retweettati sfruttando ancora una volta la tf-idf: valutiamo inizialmente il contenuto dei retweet più rapidi a livello dell'unigramma e procediamo focalizzandoci sul campo *user_description* dell'autore del messaggio originale.

3.1 Estrazione dei tweet originali

Il dataset di partenza, generato selezionando i retweet fra i messaggi estratti con Hydrator, contiene in totale 135.089 osservazioni relative a 34 variabili, la maggior parte delle quali descrivono caratteristiche dell'utente che ha pubblicato il messaggio in questione (*user_location*, *user_time_zone*, *user_followers_count*).

Disponiamo sfortunatamente solo di un campo che fornisce informazioni riguardanti l'Original Poster, autore del messaggio che è stato retweettato, variabile che tuttavia risulta cruciale in questo caso: grazie al campo *retweet_id*, identificativo unico del tweet originale, possiamo risalire a tutte le feature aggiuntive che lo caratterizzano.

Il procedimento seguito per estrarre questi dati è il medesimo illustrato nella seconda sezione del primo capitolo, durante la quale abbiamo esteso le osservazioni a nostra disposizione mediante Hydrator. In questo caso, dobbiamo semplicemente selezionare la colonna *retweet_id* dal dataset e riportarla su un apposito file txt da elaborare.

Vediamo nel dettaglio i passaggi che sono stati seguiti per collegare i retweet ai corrispondenti messaggi originali:

1. Come potevamo prevedere, non tutti i retweet che abbiamo estratto fanno riferimento ad un diverso tweet di partenza: selezionando tramite il comando `distinct()` la variabile *retweet_id* otteniamo 72.360 osservazioni, il che indica che ognuno dei messaggi originali è stato retweetato mediamente circa due volte.
2. Attraverso il processo di hydrating queste osservazioni si riducono notevolmente: dalle oltre 70.000 di partenza passiamo ad un totale di 2991 (originals dataset), ammontare purtroppo non molto significativo per le analisi descritte nei paragrafi a seguire.
3. Procediamo collegando i tweet estratti ai rispettivi retweet, operazione eseguibile semplicemente con il comando `inner_join()` basandoci sulla variabile *retweet_id* comune a entrambi i dataset. Otteniamo in questo modo 7434 osservazioni (join dataset) per le quali sono state rilevate 67 variabili, rispettivamente: 34 per il messaggio originale più 34 per il retweet, che si riducono perché ovviamente consideriamo solo una volta l'id utilizzato per effettuare l'unione.

3.2 Visualizzazione della rete di retweet

Visualizziamo per ragioni puramente descrittive le relazioni fra utenti che hanno retweetato un messaggio e autori dei tweet originali. Per rendere più comprensibile la figura ci limitiamo a rappresentare i primi 100 user per numero di retweet (ottenuti dal dataset join) in relazione ai primi 100 user più retweetati (ottenuti dal dataset originals). Le relazioni fra gli account sono rappresentate dalle linee che collegano gli username nel grafico sottostante, mentre i nodi all'esterno dell'immagine hanno una circonferenza direttamente proporzionale al numero di retweet (primi in assoluto francescatotolo e Agenzia_Ansa con 43 retweet).

Il grafico è stato realizzando combinando alcune funzioni delle librerie `igraph` [17] e `ggraph` [18].

formato UTC, considerate al secondo) in cui sono stati creati i retweet, poiché ci siamo già assicurati nel primo capitolo che gli identificativi estratti fossero univoci. Concentriamoci per il momento proprio sulle variabili temporali del dataset appena creato, per quali abbiamo effettuato le stesse operazioni di pre-processing descritte nel primo capitolo (trasformazione da character a UTC) e abbiamo creato le variabili aggiuntive suddivise per giorno e ora. Queste sono:

- *created_at_rt*: tempo di creazione del retweet;
- *created_at_orig*: tempo di creazione del tweet originale.
- *created_day_rt*, *created_hour_rt*: suddivisione per il retweet;
- *created_day_orig*, *created_hour_orig*: suddivisione per il messaggio originale;

Definiamo ora il time-to-retweet come:

$$time_to_rt = created_at_orig - created_at_rt$$

Il valore di *time_to_rt* non indica altro che il tempo che ha impiegato l'autore del tweet corrente a retweetare il messaggio originale (definizione simile a quanto descritto in [19] come "response time" o "waiting time").

Seconda la definizione, dal momento che il tempo di creazione del tweet originale è antecedente rispetto a quello del retweet, otterremo sempre un valore negativo.

Utilizziamo la funzione `difftime()` per calcolare il *time_to_rt* di ogni osservazione: il tempo mediano di retweet calcolato (in valore assoluto) è pari a 9820 secondi (2,7 ore), dunque la metà dei retweet è stata pubblicata entro questo lasso di tempo, mentre invece il tempo medio di retweet è uguale a 50.432 secondi (14 ore), il che indica la presenza di valori estremi che si discostano di molto dalla mediana. Dal grafico 3.2 si nota chiaramente che la stragrande maggioranza dei retweet avviene entro le prime 48 ore, con un picco in corrispondenza del superamento dei primi 15 minuti dal post originale. Notiamo anche un secondo picco che si manifesta tra le 9 e le 14 ore. Analizzando i retweet con un response time in questo range, constatiamo come molti di questi si riferiscano agli stessi messaggi originali, spesso pubblicati da personaggi pubblici, giornalisti o account ufficiali dello Stato, il che potrebbe causare l'effetto catena nella loro diffusione, suddivisa nel corso di un'intera giornata. Questo accade specialmente per messaggi postati tra le 20 e le 23 di sera, per forza di cose visualizzati da molti utenti il giorno successivo. Inoltre, grazie alla scala logaritmica, notiamo graficamente alcuni outlier che presentano dei time-to-retweet ben oltre la settimana.

I valori ricavati sono in linea con quanto elaborato da C. Ordun, S. Purushotham e E. Raff in [20], articolo nel quale sono stati analizzati oltre 700.000 retweet pubblicati negli Stati Uniti tra il 24 e il 28 marzo 2020, periodo di crescita esponenziale dei casi positivi seguito alla scoperta dei primi focolai statunitensi, e per i quali è stato calcolato un valore mediano di time-to-retweet pari a 2,87 ore e uno medio di

12,3 ore.

Sebbene i valori siano simili, dobbiamo considerare che i risultati ottenuti sono ristretti ad un campione estremamente esiguo di dati. Ciò nonostante, analizzando più nel dettaglio i messaggi che abbiamo ricavato, notiamo che oltre il 95% di questi è stato pubblicato tra il 20 e il 23 di febbraio, di fatto un intervallo di tempo simile a quello dell'articolo appena citato e paragonabile per ragioni epidemiologiche. È lecito assumere che i retweet in questione siano collegati all'intensa attività su Twitter seguita alla scoperta dei primi focolai in Italia, che è stata caratterizzata appunto dal rilancio delle notizie riguardanti i primi contagi. Valgono le stesse considerazioni per i messaggi originali estratti con Hydrator (2769 su 2991 pubblicati tra il 20 e il 23 febbraio), probabilmente fra i pochi ancora reperibili perché legati ad account di agenzie di informazione o di giornalisti (è più difficile che spariscano o cancellino i propri messaggi).

Proviamo adesso a confrontare i valori ottenuti con quanto calcolato nel 2013 a seguito dell'epidemia di influenza aviaria in Cina ([21]): i retweet collegati al coronavirus sono più rapidi rispetto agli omologhi collegati al virus H7N9, che manifestano un tempo mediano di 3,7 ore (60 minuti più lento) e una media di 142 ore. Anche in questo caso, incorriamo in alcune criticità: in primis sono pochi gli studi che documentano la velocità dei retweet durante lo scoppio di pandemie, pertanto non è così semplice effettuare un confronto tra il comportamento dei retweet durante la propagazione di COVID-19 e simili situazioni; ancora, l'epidemia di aviaria in Cina è avvenuta 7 anni fa e potrebbe non essere direttamente comparabile con quella attuale per diverse ragioni, si pensi ad esempio al diverso utilizzo di Twitter in Cina rispetto all'Italia e in generale al mondo occidentale.

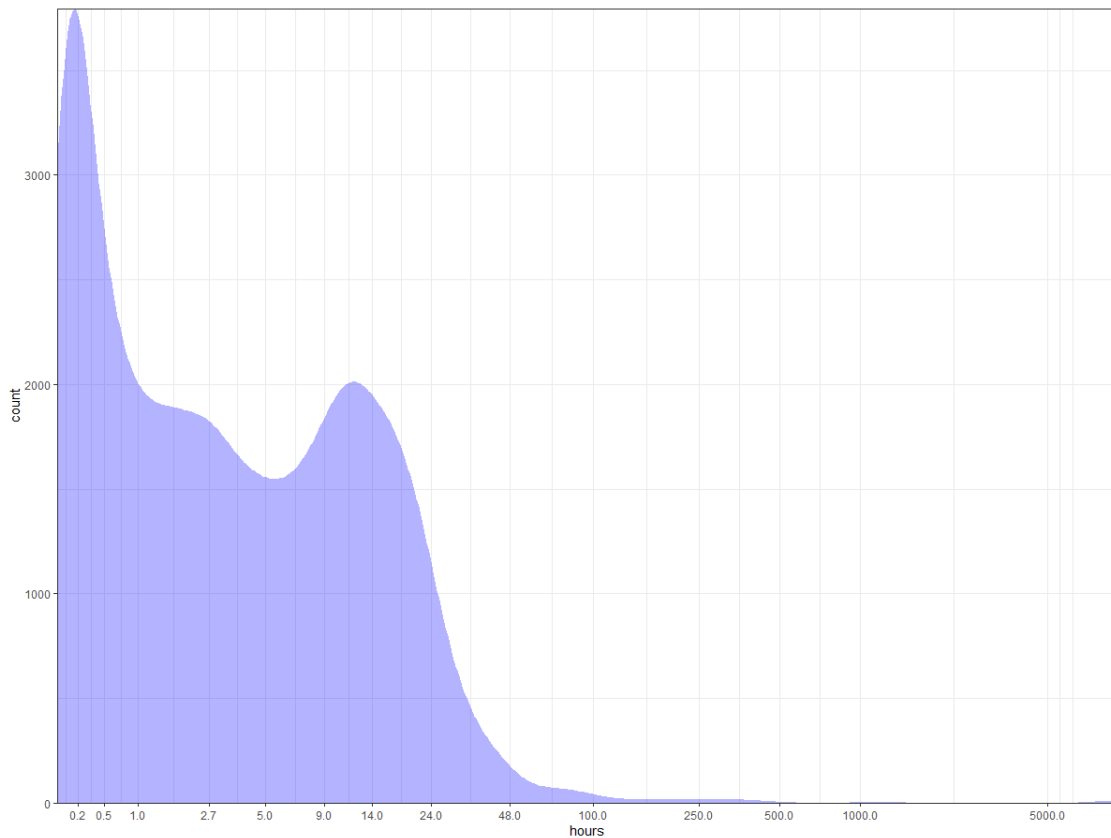


Figura 3.2: Distribuzione logaritmica del time-to-retweet espresso in ore

3.4 TF-IDF feature per i retweet più rapidi

Procediamo con la valutazione del testo dei messaggi estratti, concentrandoci su quelli che hanno manifestato un *time_to_rt* inferiore ai 10.000 secondi (circa 2,7 ore, mediana). Filtrando il dataset secondo questa condizione ed eliminando i messaggi con *retweet_id* che si ripete otteniamo 2114 osservazioni.

Come accennato nel primo capitolo, abbiamo già eseguito il pre-processing dei testi dei retweet: a differenza di quanto descritto in precedenza, abbiamo semplicemente esteso il dizionario delle stopwords includendo "RT" e non abbiamo eliminato i messaggi di nazionalità diversa da quella italiana (considerando anche che siamo rimasti con pochissimi dati da valutare).

Attraverso la funzione `bind_tf_idf()` abbiamo visualizzato quali unigrammi fossero rilevanti in termini di tf-idf nel testo dei messaggi. I risultati sono presentati in figura 3.3. Come ci aspettavamo, i termini che risaltano nel corpus si riferiscono alla scoperta dei primi due focolai in Italia (Codogno, (38)enne).

Abbiamo esteso la nostra analisi relativamente alla tf-idf valutando anche il campo

user_description degli Original Poster, attraverso la quale si dà una breve descrizione del proprio account, stavolta effettuandola sui bigrammi. I primi quindici rispetto alla tf-idf sono presentati in figura 3.4. Gli user che sono stati retweettati più velocemente si descrivono prevalentemente come account di informazione e di politica, il che è coerente con il contenuto e l'intervallo temporale dei retweet.

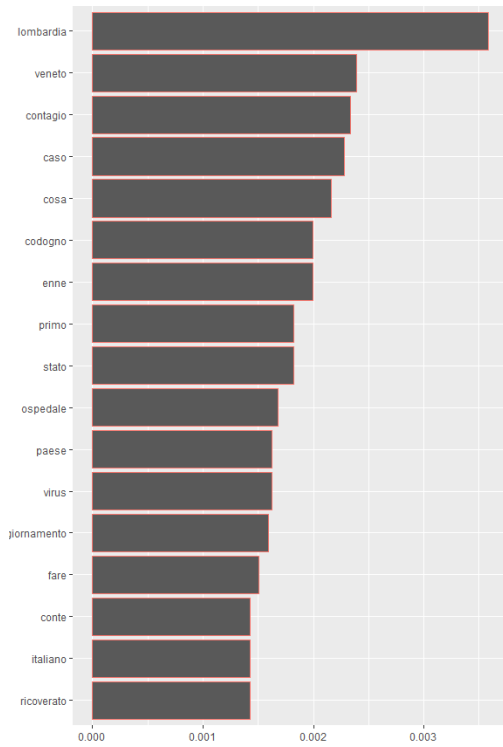


Figura 3.3: Valori di tf-idf per i testi dei retweet più rapidi

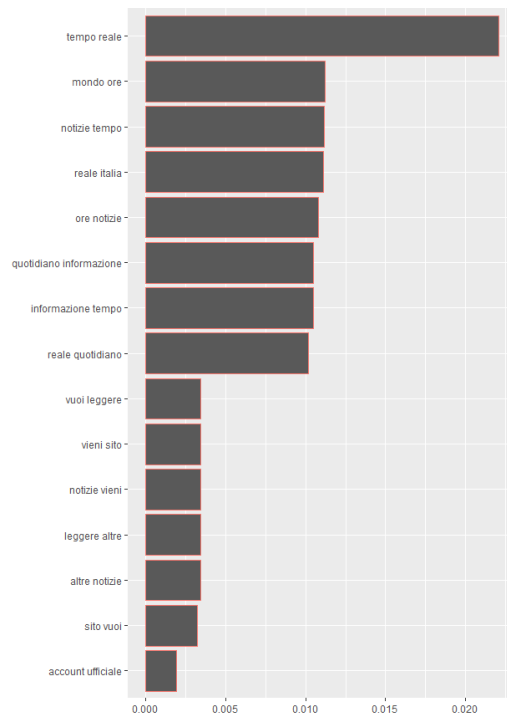


Figura 3.4: Valori di tf-idf per le descrizioni degli user più retweettati

Capitolo 4

Sentiment Analysis

Nel corso di questo capitolo, dopo aver introdotto brevemente la tecniche utilizzate e le difficoltà incontrate, applicheremo dei metodi di *sentiment analysis* alla nostra collezione di tweet originali. L'approccio che verrà utilizzato è il cosiddetto *lexicon based*, metodo che richiede l'ausilio di un dizionario per la valutazione del sentiment. Il motivo principale per cui è stato scelto di effettuare questa indagine sui tweet è l'intenzione di verificare se effettivamente intercorre una relazione tra le reazioni degli utenti sul web e l'evoluzione della pandemia nei mesi considerati, e dunque di controllare se agli eventi accaduti nel mondo reale collegati al coronavirus corrispondono degli scostamenti in termini della polarità dei messaggi analizzati.

4.1 Introduzione alla Sentiment Analysis

La *sentiment analysis* rientra nel contesto più generale dell'elaborazione del linguaggio naturale, della linguistica computazionale e dell'analisi testuale e si occupa di identificare ed estrarre, in modo automatico, opinioni e sentimenti (interscambiabili in questo contesto) contenute in un qualsiasi documento, sia esso un articolo scientifico, un commento o un post su un social network.

L'analisi del sentiment può essere ricondotta ad un problema di classificazione che si suddivide in due tipologie:

- *Sentiment polarity classification*: consiste nell'identificare l'orientamento delle frasi che compongono un documento. La classificazione può essere binaria ("positivo", "negativo"), ternaria ("positivo", "negativo", "neutro") o multiclasse ("beyond polarity"), qualora si volessero classificare anche gli stati emozionali come "felice", "triste", "arrabbiato". Più in generale, l'idea è quella che analizzando un breve testo, un tweet nel nostro caso, si riesca a determinare il sentimento (giudizio, pensiero o opinione) sottostante che esso esprime;
- *Subjectivity classification*: si occupa di classificare i documenti in due classi: "oggettivo" e "soggettivo" in base ai fatti (oggettivi) e alle opinioni (soggettive)

che vi sono contenute. Questo problema può risultare più complicato della classificazione della polarità. Ciò accade fondamentalmente per tre ragioni: la soggettività delle frasi dipende dal loro contesto; un documento oggettivo potrebbe contenere opinioni soggettive (si pensi una citazione trascritta su un articolo di giornale); i risultati sono largamente influenzati dalla definizione di soggettività utilizzata.

Le classificazioni descritte possono essere applicate su diversi livelli di granularità (intero documento, singola frase o singola parola):

- *Document level*: applica il processo di classificazione ad un intero documento, assumendo che questi contenga esclusivamente opinioni di un solo autore su una sola entità (persona, argomento, evento, prodotto in questione);
- *Sentence level*: si occupa del riconoscimento e della classificazione di singole frasi o di brevi messaggi di testo;
- *Word level*: fa utilizzo della polarità a priori delle singole parole contenute nella frase stessa. Il valore della polarità viene solitamente ricavato accedendo ad un dizionario di *opinion words* (parole intrinsecamente polarizzate), detto *lexicon*.

4.1.1 Metodi di classificazione

Come già osservato, la *sentiment analysis* può essere ricondotta ad un problema di classificazione di testi suddiviso nelle tipologie descritte nei paragrafi precedenti. Finora si sono descritti i problemi di classificazione e la granularità del testo in cui possono essere applicabili, vediamo ora quali algoritmi possono essere utilizzati. Le tecniche di classificazione, a prescindere dal livello a cui sono applicate, sono riassumibili, in base agli algoritmi e agli strumenti utilizzati, in due principali categorie:

- *Machine learning*: l'analisi del sentiment viene trattata come un problema di classificazione del testo e quindi risolta con tecniche di apprendimento automatico. A prescindere dall'algoritmo impiegato per la classificazione, tutte le tecniche includono una fase di training per un ottenere un modello su un campione di dati, ovvero di testi cui è già stato assegnato un giudizio di valore (sentiment positivo o sentiment negativo), e quindi l'impiego del modello su dati nuovi;
- *Lexicon based*: la definizione del sentiment è basata sull'analisi delle singole parole o frasi sfruttando dizionari di parole "polarizzate" cui è assegnato un peso in termini di positività o negatività; le stesse parole vengono cercate nei testi di cui si vuole indagare il sentiment e il risultato dipende dalla frequenza

di quelle parole nel testo. A differenza dell'approccio *machine learning*, quindi, il *lexicon based* richiede, nella misurazione del sentimento, un dizionario di riferimento sul quale il testo di cui si vuole estrarre il sentiment viene confrontato. Nella pratica, un dizionario potrebbe contenere la parola "buono", cui viene assegnato il punteggio +1, e la parola "cattivo", cui viene assegnato il punteggio -1. Nella più semplice implementazione di un modello basato sull'approccio *lexicon based*, dunque, tutte le parole del documento da analizzare sono confrontate con le parole già valutate dal dizionario: ogni volta che una parola del documento da analizzare corrisponde ad una presente nel dizionario, il punteggio associato a quella parola viene sommato al punteggio complessivo del sentiment del documento. Il risultato finale sarà nient'altro che una somma di tutti i punteggi.

Entrambi gli approcci presentati hanno vantaggi e svantaggi: il *lexicon based* non richiede una precedente operazione di *labeling* dei testi in entrata, ma si basa su dizionari già esistenti e perciò ha una portata di analisi più ampia. Questo può tuttavia rappresentare un duplice limite per l'indagine: da un lato la non scontata disponibilità di dizionari sufficientemente complessi nella lingua del testo su cui l'indagine è svolta, dall'altro la difficoltà di tenere conto del contesto dell'indagine e quindi dell'adeguatezza del dizionario che viene utilizzato, poiché lo stesso dizionario potrebbe non avere la stessa efficacia su contesti d'indagine differenti. Inoltre, le collezioni di documenti nel caso dell'approccio basato su dizionario vengono valutate allo stesso modo (assegnando gli stessi punteggi) indipendentemente dall'importanza che assume ogni termine all'interno dei testi.

Dall'altro lato, l'approccio *machine learning* non richiede solitamente il ricorso ad un dizionario e sfrutta l'accuratezza dei metodi di classificazione utilizzati anche in altri strumenti di indagine. Tuttavia, questa accuratezza dipende fortemente da una corretta etichettatura dei testi utilizzati per il training e da una attenta selezione delle feature considerate dall'algoritmo.

Vale la pena sottolineare che i due approcci possono essere utilizzati individualmente o in combinazione tra loro tramite l'approccio ibrido, che potrebbe essere una soluzione per ovviare alle problematiche intrinseche dei due metodi appena illustrate.

4.1.2 Problematiche nella Sentiment Analysis

Rispetto ai problemi di classificazione classici, troviamo nella *sentiment analysis* diverse problematiche a dispetto del ridotto numero di classi da considerare, generalmente due o tre. Questo è dovuto soprattutto alla sottile distinzione che esiste tra sentimento positivo e negativo (distinzione che risulta spesso difficile anche per un essere umano). Non sempre le opinioni sono espresse tramite l'uso di *opinion words*; in molti casi il sentimento è espresso in modo indiretto o tramite artifici linguistici

quali metafore o altre figure retoriche, o con l'uso di espressioni informali e slang non appartenenti al vocabolario linguistico. Proviamo poi a pensare ad una frase contenente ironia o sarcasmo, dove l'interpretazione del significato è strettamente soggettiva. Ulteriori problematiche sono dovute alle ambiguità sintattiche e alla difficoltà di circoscrivere il dominio trattato: si osserva, infatti, come alcune opinioni siano strettamente dipendenti dal contesto e dall'ambito di riferimento. Aggettivi e avverbi che risultano essere positivi per alcune entità possono risultare negativi per altre; in proposito al contesto trattato, si pensi proprio all'aggettivo "positivo", chiaramente classificato con sentiment maggiore di zero da qualsiasi dizionario, ma che nel nostro caso assume valenza neutra nella maggior parte dei casi, qualora il messaggio in questione sia di carattere informativo e riferisca il numero dei nuovi contagi, o tutt'al più negativa se il tweet fa riferimento alla positività al test per il coronavirus dell'autore o di suoi amici/parenti.

4.1.3 La lingua italiana e il contesto di riferimento

Se le definizioni, gli approcci e le tecniche per la *sentiment analysis* finora descritte sono valide a prescindere dalla lingua utilizzata ciò non è altrettanto vero per gli strumenti, le regole grammaticali ed i modelli linguistici. Ogni lingua è diversa dall'altra e pensare di condurre una *sentiment analysis* per la lingua italiana utilizzando gli stessi strumenti e le stesse regole che funzionano per quella inglese, che prendiamo come riferimento dato che è quella più studiata in letteratura, porterebbe solamente a risultati inaccurati. C'è poi da considerare il grande sbilanciamento di risorse online presente appunto tra la lingua inglese e la lingua italiana. La disparità di risorse è uno dei principali motivi per i quali le problematiche presentate non fanno altro che acuirsi se consideriamo il nostro dominio di applicazione, vale a dire una collezione di tweet pubblicati in Italia. Basti pensare che mediante la libreria `tidyverse` si può accedere a tre dizionari, sviluppati per la lingua inglese, che assegnano punteggi agli unigrammi secondo diversi criteri, in particolare abbiamo: AFINN assegna ai termini un punteggio compreso tra -5 (più negativo) e +5 (più positivo), Bing si limita ad una classificazione della polarità positiva o negativa e NRC opera una classificazione che va oltre le due polarità, includendo appunto categorie emozionali quali tristezza, rabbia e paura. Torniamo con i piedi per terra se consideriamo gli strumenti disponibili per la lingua italiana. Il dizionario utilizzato durante la nostra indagine è quello fornito dalla libreria `TextWiller`, grazie al quale è possibile assegnare ad ogni token tre possibili valori: +1 per i termini positivi (es. "meglio"), 0 per i termini neutri (es. "Lombardia") e -1 per quelli negativi (es. "morti").

Il contesto di riferimento è stato scelto come terreno su cui applicare la *sentiment analysis* fondamentalmente per la peculiarità della situazione italiana rispetto alla

diffusione del virus. L'Italia è stato infatti il primo grande paese ad essere colpito nel mondo occidentale ed è riuscita in un tempo relativamente breve (si veda la gestione della pandemia negli Stati Uniti) a debellare il virus, il che rende interessante l'analisi da un punto di vista temporale proprio per verificare se intercorre una relazione tra gli eventi più drammatici che hanno caratterizzato i mesi considerati (così come quelli più positivi) e l'andamento del sentiment dei tweet. Inoltre, se restringiamo il contesto alla sola dimensione virtuale, sono state numerose le campagne lanciate su Twitter per sensibilizzare la popolazione a restare a casa (si pensi all'hashtag #andràtuttobene) e a donare quanto possibile per sostenere economicamente ospedali, operatori sanitari e lavoratori essenziali (ad esempio la raccolta fondi per il San Raffaele). Ancora, durante le fasi più concitate della gestione dell'emergenza, hanno acquisito sempre più rilievo le conferenze stampa del Presidente del Consiglio in merito alle direttive imposte alla popolazione; è interessante quindi vedere che ruolo hanno giocato le rispettive chiusure e riaperture in termini di polarità dei messaggi.

Alla luce di queste considerazioni, l'analisi dei tweet sul COVID-19 in Italia da un punto di vista del sentiment coinvolge tre diversi livelli di discussione:

- il sentiment degli italiani in relazione all'evoluzione della pandemia, dallo scoppio dei primi focolai in Lombardia e Veneto alla riapertura del 4 maggio;
- l'impatto avuto dall'adesione alle campagne online sul sentiment;
- il sentiment degli italiani rispetto alle conferenze stampa del Presidente del Consiglio.

4.2 Classificazione degli unigrammi

Iniziamo la nostra analisi con una semplice valutazione degli unigrammi appoggiandoci al dizionario fornito dalla libreria `TextWilla`, la quale ci permette di assegnare ai termini ottenuti tramite la tokenization un determinato valore attraverso la funzione `sentiment()`. Considerando che non tutti i token sono riconosciuti dalla funzione e che la nostra analisi si limita ad una valutazione della polarità, dagli oltre 800.000 termini iniziali estraiamo un dataframe contenente circa 200.000 parole etichettate con la rispettiva classe di appartenenza ("positive" o "negative"). Confrontando questo dataset (contenente le classi) con la totalità dei token presenti nel corpus si perviene alla figura 4.1, nella quale vengono presentati i termini più frequenti suddivisi per polarità.

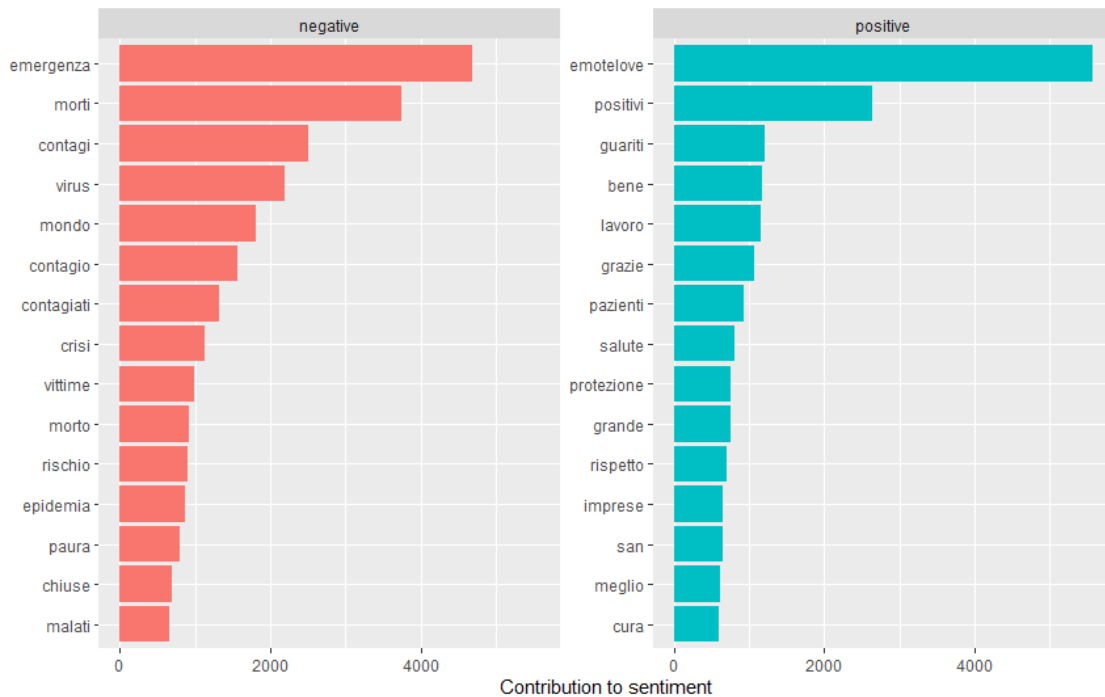


Figura 4.1: Primi 15 termini suddivisi per polarità

Dall'immagine è evidente che il contesto di riferimento gioca un ruolo non indifferente nella *sentiment analysis* e porta ad accentuare le problematiche di cui sopra. Fra i termini catalogati come "positive", è proprio il termine "positivi" che si trova in primo piano, secondo solo all'emoticon del cuore. Altri termini che sono considerati positivi ma che nel contesto di riferimento assumono valenza neutra sono "pazienti", "ordinanza", "influenza". Per quanto riguarda le parole negative si segnalano meno ambiguità, seppur il dizionario non ne sia completamente esente (ad esempio "mondo" è considerata negativa).

Prima di proseguire nell'indagine, propendiamo per la rimozione delle parole più frequenti che in relazione alla pandemia non sono polarizzate. Ovviamente questa è una soluzione semplice ad un problema ben più complesso, è plausibile infatti che siano presenti diversi altri termini caratterizzati da una polarità "sbagliata" rispetto al contesto trattato, motivo per cui un'analisi più accurata dovrebbe prevedere una classificazione a livello dei tweet e non delle singole parole. In ogni caso, con la rimozione di alcuni dei termini più frequenti eliminiamo alcune alterazioni semantiche che potrebbero distorcere i risultati.

4.3 Andamento del sentiment nel tempo

Passiamo ora alla principale domanda di ricerca che coinvolge il sentiment dei tweet collegati al coronavirus, ovvero la relazione che intercorre fra il sentimento che traspare dei messaggi e l'evoluzione della pandemia in Italia. A questo scopo, costruiamo un dataframe contenente per ogni giorno di rilevazione il relativo sentiment, calcolato nel seguente modo:

$$sentiment = positive - negative$$

Poiché il dizionario utilizzato assegna i valori +1 e -1, il risultato finale non sarà altro che la differenza tra il numero di parole positive e negative calcolate in una giornata. Dal momento che non tutti i giorni di rilevazione presentano lo stesso numero di tweet estratti, cosa che potrebbe causare una distorsione verso i giorni più prolifici in termini di rilevazioni, effettuiamo una normalizzazione dei valori ottenuti secondo la formula:

$$x_i^{norm} = \frac{x_i - \max(|x_i|)}{\max(|x_i|) - \min(|x_i|)}$$

dove x_i indica il sentiment giornaliero.

A questo punto visualizziamo l'andamento del sentiment suddividendo l'intervallo temporale in quattro sezioni, in modo tale da vedere nel dettaglio in corrispondenza di quali giorni sono avvenute le principali fluttuazioni.

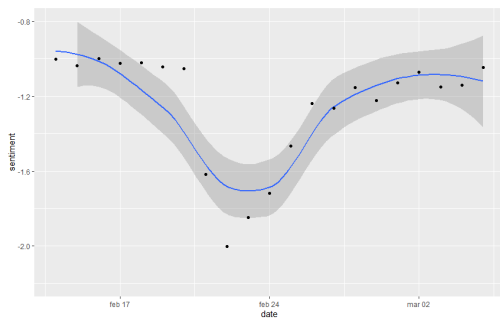


Figura 4.2: Andamento sentiment dal 14/02 al 5/03

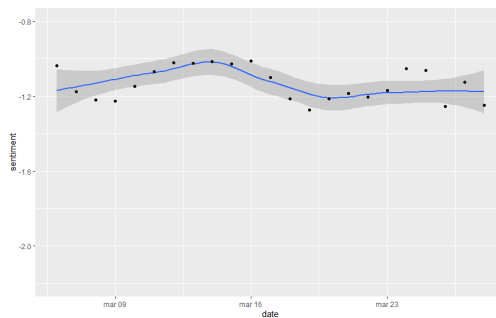


Figura 4.3: Andamento sentiment dal 6/03 al 28/03

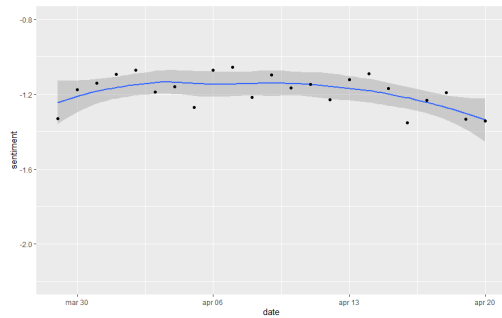


Figura 4.4: Andamento sentiment dal 29/03 al 20/04

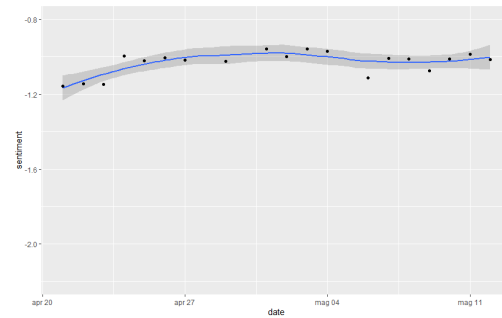


Figura 4.5: Andamento sentiment dal 21/04 al 12/05

I valori ottenuti sono prevalentemente negativi anche senza la normalizzazione, indice del fatto che i termini catalogati come "negative" sono in quantità superiore ai positivi. Come era prevedibile, il limite inferiore della curva viene raggiunto in corrispondenza della scoperta dei primi focolai localizzati in Lombardia e Veneto, dei quali abbiamo già parlato nei capitoli precedenti sottolineando come abbiano scatenato una reazione di sdegno e paura sul web. Si assiste successivamente ad un livello pressoché costante fino all'8-9 marzo, giorni in cui sono stati decretati i primi lockdown a livello regionale dell'intera Lombardia e di alcune province del Piemonte e dell'Emilia Romagna, chiusure che, ancora una volta, sono state ampiamente protagoniste dei tweet e che hanno fatto prendere coscienza alla popolazione della gravità della situazione. Non è chiaro perché a seguito di queste chiusure e delle conseguenti conferenza stampa del Presidente del Consiglio si veda un andamento crescente del sentiment fino al 16 marzo. Una possibile spiegazione è il lancio in quella settimana delle campagne per sensibilizzare i cittadini a rispettare le misure imposte e a effettuare donazioni per sostenere il Sistema Sanitario Nazionale con hashtag quali #andràtuttobene e #restiamoacasa, che hanno causato un aumento dei termini con polarità positiva come "bene", "solidarietà", "grazie". È curioso inoltre notare che l'inversione di tendenza si assista proprio in corrispondenza di una delle giornate più drammatiche vissute dall'inizio dell'emergenza: a Bergamo, una delle città più colpite dalla pandemia, nella serata di mercoledì 18 marzo, sono arrivati i mezzi dell'esercito per trasportare le bare di alcune delle persone morte di COVID-19 dal cimitero ai forni crematori di altre città. Sono di più difficile interpretazione i dati del mese di aprile: nonostante il 10 aprile sia stato prolungato il lockdown fino al 4 maggio e conseguentemente siano state annullate le festività pasquali non si nota un'inversione del trend che rimane stabile fino al 19-20-21 aprile, giornate in cui si assiste ad una brusca discesa. Al contrario, il rialzo in corrispondenza del 23 e 24 aprile potrebbe essere associato alla notizia diffusa dalla stampa del parziale allentamento delle misure imposte a partire dal 4 maggio, poi ufficializzato durante la serata del 26/04 con il consueto discorso in diretta nazionale. L'andamento rimane simile fino al termine dei giorni considerati, con un unico

outlier in corrispondenza del 5/05.

4.4 Limiti e possibili sviluppi

L'analisi eseguita si è limitata ad un approccio *lexicon based* basato sugli unigrammi, scelta che da un lato ha di gran lunga semplificato il processo di valutazione del sentiment ma che dall'altro, come espresso in precedenza in merito alle problematiche di questo approccio, ne ha accentuato i limiti e gli errori che si possono commettere. Nonostante si siano trovati risultati interessanti e in parte compatibili con la realtà di riferimento, l'approccio utilizzato è intrinsecamente fallace, poiché, banalmente, la maggiore presenza di termini negativi all'interno di un testo non indica affatto che questo esprima un'opinione negativa.

Uno dei possibili sviluppi del lavoro svolto potrebbe essere l'estensione da una *word level classification* ad una *sentence level classification*, che dunque preveda l'attribuzione della polarità ai singoli tweet e non alle parole che contengono, basata sui metodi di *machine learning*. A tal proposito, fra i classificatori più semplici da implementare figura il Naive Bayes Classifier. Attraverso di esso si può calcolare la probabilità che, data una parola, questa appartenga ad una certa classe sulla base di un training set, all'interno del quale sono contenuti dei messaggi già correttamente classificati da esseri umani. Tuttavia, anche questo metodo presenta delle criticità, fondamentalmente per una ragione: il contesto di riferimento dei messaggi già classificati gioca un ruolo fondamentale per la previsione di eventuali osservazioni non catalogate; riagganciandoci ancora all'esempio fatto in precedenza, è quasi certo che il termine "positivo" all'interno di un tweet pubblicato al di fuori del contesto di una pandemia mondiale abbia un significato diverso da quello in esame. Se a questo aggiungiamo la scarsità di risorse disponibili per la lingua italiana, allora anche un approccio di questo tipo potrebbe presentare delle gravi mancanze.

Ecco quindi che la soluzione migliore sarebbe la classificazione manuale di qualche centinaia di tweet in lingua italiana relativi al coronavirus, anche in vista dello sviluppo di modelli di apprendimento automatico molto più complessi (un riferimento per i tweet in lingua inglese è [22]).

Terminata questa sezione, proseguiamo nell'ultimo capitolo con un'applicazione di topic modeling, un'altra delle aree del Natural Language Processing esplorate in questa tesi.

Capitolo 5

Topic Modeling

Arrivati al termine dell'elaborato, applichiamo il modello statistico generativo Latent Dirichlet Allocation (LDA) ai tweet in esame, non prima di averne fornito una rigorosa descrizione metodologica.

Similmente a quanto visto per la sentiment analysis, anche l'idea di applicare questo algoritmo di topic modeling nasce dalla natura dei dati a disposizione: è risaputo che i risvolti drammatici della pandemia hanno coinvolto quasi ogni attività umana, dall'economia ai sistemi sanitari, passando per il turismo e l'istruzione; ci chiediamo quindi se questa suddivisione tematica traspare anche dai messaggi collezionati e in particolare se contestualmente all'evolversi dell'emergenza si è verificata una variazione nelle proporzioni di tweet appartenenti ad un determinato argomento.

5.1 Introduzione al Topic Modeling

Nell'ambito del machine learning e del Natural Language Processing, un topic model è un modello statistico utilizzato per identificare i topic che si presentano in una collezione di documenti, ovvero i macroargomenti che sottendono i testi considerati. Il topic modeling è di conseguenza una tecnica usata spesso nel text mining per scoprire la struttura semantica nascosta all'interno di un testo. Intuitivamente, considerando l'ambito di ricerca in esame, dato un messaggio riguardante il coronavirus ci aspettiamo che parole come "contagiati", "malati", "terapie intensive" facciano riferimento a messaggi riguardanti l'aspetto sanitario della pandemia, mentre parole come "crisi", "economia", "recessione" siano più presenti in testi che esplorano le conseguenze economiche dell'emergenza. I topic generati dalle tecniche di topic modeling sono cluster di parole simili. Un topic model cattura questa intuizione in un contesto matematico, il che permette di esaminare un insieme di documenti per scoprire quali potrebbero essere i topic che vi sono contenuti e soprattutto qual è la proporzione di ogni topic all'interno dei documenti.

Si fa talvolta riferimento ai topic model anteponendogli l'aggettivo *probabilistic*,

che fa riferimento ai metodi utilizzati per scoprire la struttura semantica latente di un corpus testuale. Tali tecniche possono svolgere questo compito in maniera quasi autonoma, senza l'ausilio di esseri umani, difatti non sono necessari training set con categorie predefinite per "etichettare" i topic, poiché spesso l'unica operazione specificata dal ricercatore è il numero di topic da identificare. L'algoritmo genera il numero di topic specificato e restituisce le probabilità che ogni parola appartenga ad un determinato topic, così come la distribuzione dei topic all'interno del corpus considerato. Ovviamente l'algoritmo non è in grado di dirci a quale argomento facciano riferimento i topic individuati; sarà compito del ricercatore trovare una similarità semantica tra le parole inserite all'interno degli specifici cluster che vengono creati, ognuno dei quali corrisponde ad un topic a cui sono associati i termini con probabilità più alta. Tutto ciò posiziona il topic modeling all'interno della sfera di algoritmi di apprendimento automatico che prende il nome di *unsupervised learning*.

L'assunto fondamentale alla base di questi modelli è che i documenti sono una miscela di topic, e dunque includono più argomenti con diverse proporzioni, mentre a loro volta i topic hanno una specifica distribuzione di probabilità sulle parole di un vocabolario predefinito.

Un topic model può essere altresì visto quindi come un modello per la generazione di documenti: per generare un nuovo testo si estrae un topic, e successivamente un termine dalla distribuzione sul vocabolario corrispondente; il processo va iterato per tutta la lunghezza del documento. Il processo può essere a sua volta invertito attraverso tecniche statistiche, allo scopo di fare inferenza sull'insieme di topic che ha generato il documento. Sono stati proposti nel corso degli anni svariati modelli per l'analisi dell'informazione contenuta nei documenti, i quali si differenziano principalmente per le assunzioni statistiche alla loro base.

Sebbene non siano infallibili, tali metodi dimostrano di possedere un'alto grado di affidabilità e una buona interpretazione semantica dei testi, per questo hanno svariate applicazioni in diversi campi del machine learning: dal riconoscimento delle immagini alla bioinformatica. In figura 5.1 si può osservare un primo esempio applicato ad un articolo di genetica.

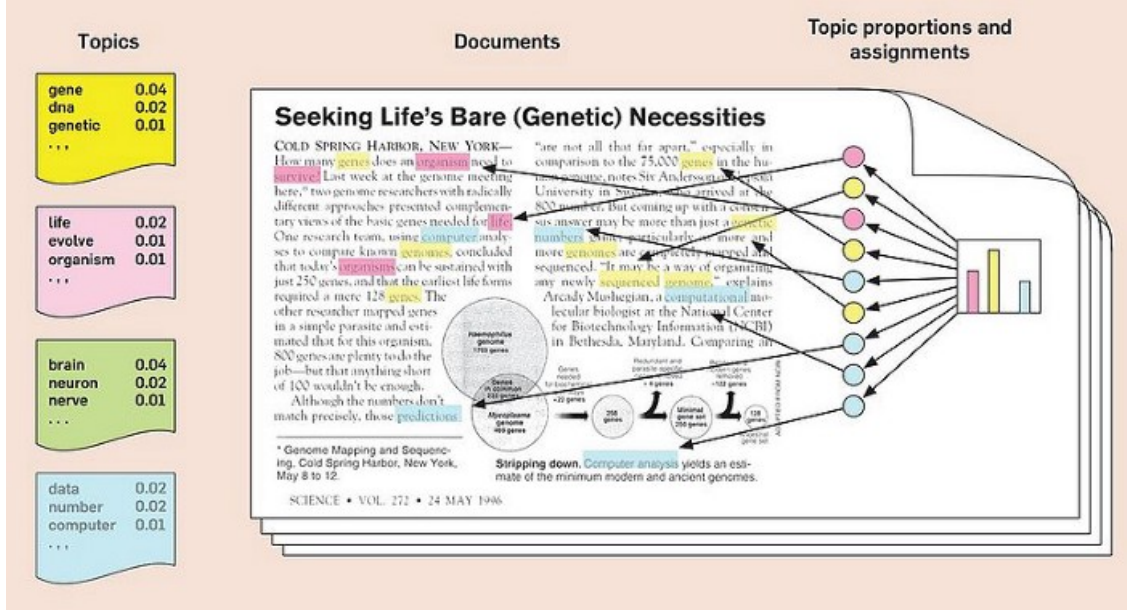


Figura 5.1: Esempio introduttivo di topic model

5.2 Latent Dirichlet Allocation: il modello teorico

La Latent Dirichlet Allocation è un modello statistico di tipo generativo in grado di creare un corpus di documenti (introdotta per la prima volta da David Blei nel 2003 [23]). Partendo dagli assunti di base appena descritti, LDA rappresenta ogni documento come una mistura di topic, dove ogni topic è una distribuzione multinomiale sulle parole del vocabolario. Al fine di effettivamente determinare i topic, immaginiamo di generare i documenti attraverso il seguente processo, in modo tale da fare inferenza sulla struttura degli argomenti latenti:

1. Si estrae $\theta_i \sim \text{Dirichlet}(\alpha)$, dove $i \in \{1, \dots, M\}$ (M numero di documenti) e $\text{Dirichlet}(\alpha)$ è la distribuzione di Dirichlet di parametro α
2. Si estrae $\phi_k \sim \text{Dirichlet}(\beta)$, dove $k \in \{1, \dots, K\}$ (K numero di topic)
3. Date le distribuzioni, per ogni parola in posizione (i, j) , j -esima parola dell' i -esimo documento (dove $j \in \{1, \dots, N_i\}$ e $i \in \{1, \dots, M\}$):
 - (a) si estrae un topic da $z_{i,j} \sim \text{Multinomiale}(\theta_i)$, dove $z_{i,j}$ è il topic per la j -esima parola nel documento i
 - (b) si estrae una parola da $w_{i,j} \sim \text{Multinomiale}(\phi_{z_{i,j}})$, dove $w_{i,j}$ è una parola associata al topic

θ descrive le cosiddette *document-topics distributions*, ovvero le probabilità che una dato documento d appartenga ad un certo topic k : i documenti presentano dunque una distribuzione sui topic latenti. Questa distribuzione è rappresentata dalla distribuzione di Dirichlet.

ϕ rappresenta le cosiddette *topic-words distributions*, ovvero le probabilità che hanno le parole di essere estratte per uno specifico topic.

In altre parole, per ogni documento e per ogni parola che gli appartiene, un topic è scelto sulla base della document-topics distribution (*Multinomiale*(θ)), generando in questo modo una mistura di topic per ogni documento. Successivamente, una parola è estratta dal vocabolario, considerando le topic-words distributions per ogni topic all'interno della mistura di documenti.

Un esempio potrebbe essere il seguente. Supponiamo che un documento abbia una distribuzione sui topic di questo tipo: 50% politica, 40% economia, 8% sport, 2% tecnologia. Ogni topic ha a sua volta una distribuzione sulle parole del vocabolario: per il topic "sport", alla parola "calcio" è associata una probabilità più alta rispetto al termine "iPhone"; per il topic "tecnologia" vale esattamente il contrario, e così via. La generazione di un documento viene eseguito in questo modo: si parte scegliendo la distribuzione del documento sui topic, come quella appena descritta; si prosegue scegliendo casualmente uno dei topic, ipotizziamo "economia"; si sceglie casualmente una parola basandosi sulla specifica topic-words distribution. Quest'ultima sarà la prima parola del documento generato. Il processo generativo deve essere ripetuto per ogni documento all'interno del corpus.

Dati α e β , la distribuzione multivariata finale è data da:

$$Pr(\mathbf{W}, \mathbf{Z}, \phi, \theta; \alpha, \beta) = \prod_{i=1}^K Pr(\phi_i; \beta) \prod_{j=1}^M Pr(\theta_j; \alpha) \prod_{t=1}^N Pr(Z_{j,t}|\theta_j) Pr(W_{j,t}|\phi_{Z_{j,t}})$$

La distribuzione congiunta viene invertita per ottenere la distribuzione condizionata, detta anche distribuzione a posteriori. Nel caso di LDA, le variabili osservate sono le parole dei documenti, mentre ciò che è ignoto è la struttura latente dei topic, quindi le distribuzioni di θ e ϕ . Il modello LDA può essere rappresentato graficamente nello schema riportato in figura 5.2. Come si può notare, sono presenti tre diversi livelli, ognuno dei quali rappresenta rispettivamente il corpus, i documenti e i termini. Nel rettangolo inferiore si può vedere come per ogni termine venga estratto sia z che w : ciò significa che per ogni parola all'interno del documento viene estratto un topic, il che implica che il singolo documento viene descritto da più topic contemporaneamente. Le variabili θ_i si estraggono per ogni documento, mentre i parametri α e β sono estratti una sola volta per l'intero processo generatore.

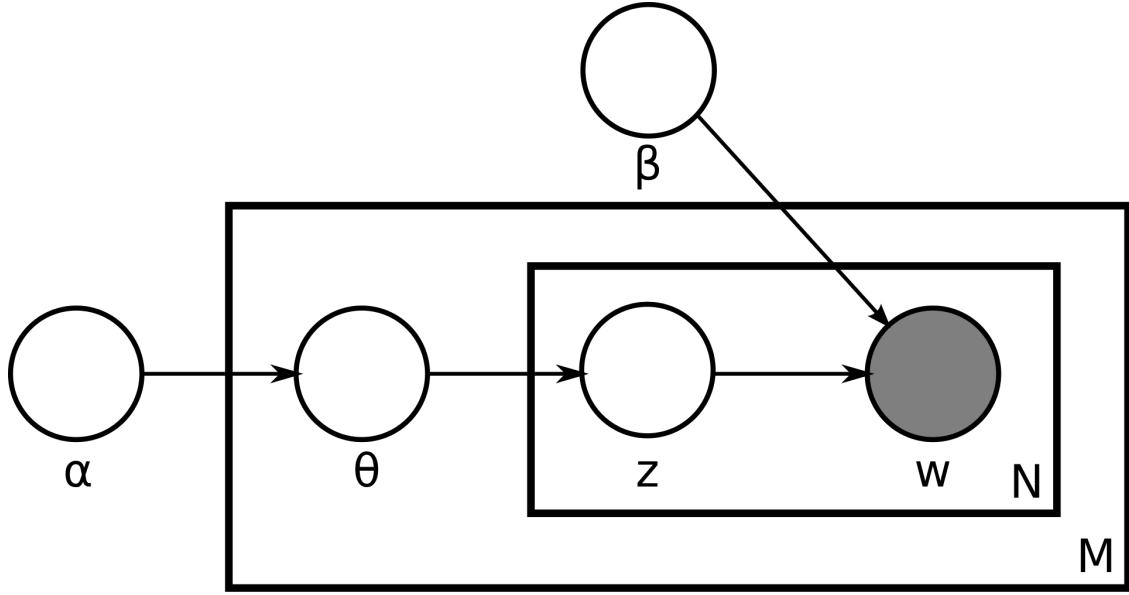


Figura 5.2: Modello grafico per LDA

5.2.1 Stima dei parametri: il campionamento di Gibbs

Dopo aver presentato dal punto di vista teorico il modello LDA, passiamo alla procedura di stima dei parametri. Le variabili di interesse sono la distribuzione dei topic per ogni documento (θ) e la distribuzione delle parole nei topic (ϕ). Non risulta invece oggetto di inferenza il numero di parametri K , che viene dato per fissato.

Sono stati proposti in letteratura diversi approcci che si basano sia sull'algoritmo EM (Expectation-Maximization) che su simulazioni basate sui metodi Monte Carlo (MCMC) come il campionamento di Gibbs. In questo paragrafo verrà presentato quest'ultimo, implementato per la fase di stima nella funzione `LDA()`.

Dato l'alto numero di parametri c'è il rischio che l'algoritmo rimanga "bloccato" in massimi locali della distribuzione a posteriori. Una possibile soluzione è di non fare inferenza su θ e ϕ , bensì stimare la distribuzione a posteriori di z (le assegnazioni delle parole ai topic), condizionando rispetto alle parole osservate w . In particolare, vale la pena utilizzare il campionamento di Gibbs per generare dalla a posteriori di z , marginalizzando rispetto a θ e ϕ .

Griffiths e Steyvers [24] hanno dimostrato che la distribuzione condizionata può essere calcolata come segue:

$$Pr(z_i = j | \mathbf{z}_{-i}, w_i, d_i, \cdot) \propto \frac{C_{w_i j}^{WT} + \beta}{\sum_{w=1}^W C_{w j}^{WT} + W\beta} \frac{C_{d_i j}^{DT} + \alpha}{\sum_{t=1}^T C_{d_i t}^{DT} + T\alpha}$$

dove C^{WT} e C^{DT} sono matrici di conteggi di dimensioni $W \times T$ e $D \times T$. C^{WT} contiene il numero di volte che la parola w è assegnata al topic j , senza includere il passo corrente i ; C^{DT} contiene il numero di volte che il topic j è assegnato al documento d , senza includere il passo corrente i . Il quoziente a sinistra dell'equazione rappresenta la probabilità della parola w sul topic j , mentre quello a destra la probabilità del topic j sul documento d . Ogni volta che una parola è assegnata al topic j , la probabilità di assegnare altre parole specifiche a questo topic aumenta. Allo stesso tempo, se il topic j è usato più volte nello stesso documento, aumenta la probabilità che le parole del documento vengano assegnate ad esso. Quindi le parole sono assegnate ai topic più verosimili come ai topic predominanti in un documento. L'algoritmo fornisce stime dirette di z per ogni parola; rimane comunque di interesse conoscere le stime per θ e ϕ . Queste possono essere ottenute come segue:

$$\phi_i^{(d)} = \frac{C_{ij}^{WT} + \beta}{\sum_{k=1}^W C_{kj}^{WT} + W\beta} \quad \theta_j^{(d)} = \frac{C_{dj}^{DT} + \alpha}{\sum_{k=1}^T C_{dk}^{DT} + T\alpha}$$

I valori corrispondono rispettivamente alle distribuzioni di probabilità predettive di estrarre una nuova parola i dal topic j , e di estrarre un nuovo topic j nel documento d ; in particolare sono le medie a posteriori di queste quantità condizionate ad un particolare valore di z .

5.3 Implementazione

La libreria di riferimento per l'implementazione dell'algoritmo è `topicmodels` [25], all'interno della quale troviamo la funzione `LDA()`. Questa richiede in input la Document Term Matrix dei testi (DTM), il numero di topic K e il metodo scelto per stimare i parametri (utilizziamo Gibbs). Prima di effettivamente applicare la funzione al corpus in esame è doveroso fare alcune precisazioni sul numero K di topic che, come abbiamo specificato nei paragrafi precedenti, deve essere noto a priori.

5.3.1 Selezione del numero di topic

Il numero di topic ha una grande influenza sui risultati prodotti dall'algoritmo, ma spesso l'interpretazione di questi ultimi è soggettiva e difficile da valutare. Un numero di topic troppo basso porterebbe ad avere un modello approssimativo, e dunque inadeguato per cogliere la struttura latente degli argomenti sottostanti al corpus; d'altro canto, un numero elevato renderebbe l'interpretazione troppo difficile, senza considerare che aumenterebbero i casi di termini che si presentano in più di un topic.

Nonostante l'approccio soggettivo sia di gran lunga il migliore per determinare qual è l'effettivo numero di topic che meglio cattura gli argomenti all'interno dei documenti, esistono alcune metriche che permettono di valutare la bontà del numero

scelto. Fra queste troviamo il Rate of Perplexity Change (RPC) [26], metrica basata sulla verosimiglianza, l'Entropy Optimized Latent Dirichlet Allocation (En-LDA) [27], metrica basata sull'entropia e il Coherence Score [28], utilizzata nel nostro caso.

Quest'ultima è definita come:

$$CoherenceScore = \sum_{i < j} Score(w_i, w_j)$$

con:

$$Score(w_i, w_j) = \log \frac{D(w_i, w_j) + 1}{D(w_i)}$$

dove w_i e w_j sono le parole con la più alta probabilità di essere estratte per ogni topic considerato.

Il numeratore della funzione $D(w_i, w_j)$ è il numero di documenti in cui le parole w_i, w_j compaiono contemporaneamente. Si noti che viene aggiunto l'uno per evitare di avere $\log(0)$ qualora le due parole non apparissero mai insieme nello stesso documento. Si divide per il numero di documenti in cui w_i appare, evitando in questo modo che parole con un'alta frequenza non abbiano uno score troppo elevato. Lo score calcolato sarà tanto più alto quanto più w_i e w_j appaiono negli stessi documenti, rispetto a quanto spesso w_i appare singolarmente. Questo ci dà un'idea della coerenza dei topic estratti, poiché se due parole di un topic hanno veramente affinità semantica allora ci aspettiamo che compaiono spesso negli stessi documenti. Più alto è il valore per un determinato numero di K, meglio saranno collegate fra loro le parole. Ad esempio, se confrontiamo {cane, televisione, libro, articolo} con {cane, osso, abbaire, palla}, quest'ultimo otterrà un Coherence Score maggiore per via del collegamento semantico fra i termini.

Partendo dalla DTM, al fine di ridurre l'onere computazionale, consideriamo solo i termini che compaiono almeno nell'1% e al massimo nel 60% dei documenti. Eseguiamo l'algoritmo facendo variare il valore di K da 1 a 20, ipotizzando che un numero troppo elevato di argomenti sarebbe andato a scapito dell'interpretabilità. Calcoliamo per ogni modello il corrispondente Coherence Score tramite la funzione `CalcProbCoherence()`, disponibile tramite la libreria `textmineR` [29] e visualizziamo l'andamento della metrica in figura 5.3. Notiamo che il valore massimo viene raggiunto in corrispondenza di K=7. Sebbene il valore associato non sia particolarmente alto, propendiamo per mantenere il modello con K=7 siccome ci dà buoni riscontri in termini di interpretabilità dei topic. Va precisato che la scelta del numero di topic non è mai semplice e che avremmo in ogni caso deciso di percorrere la strada per arrivare più facilmente all'identificazione dei macroargomenti che sottendono il dataset considerato, anche se si fosse manifestato un Coherence Score minore.

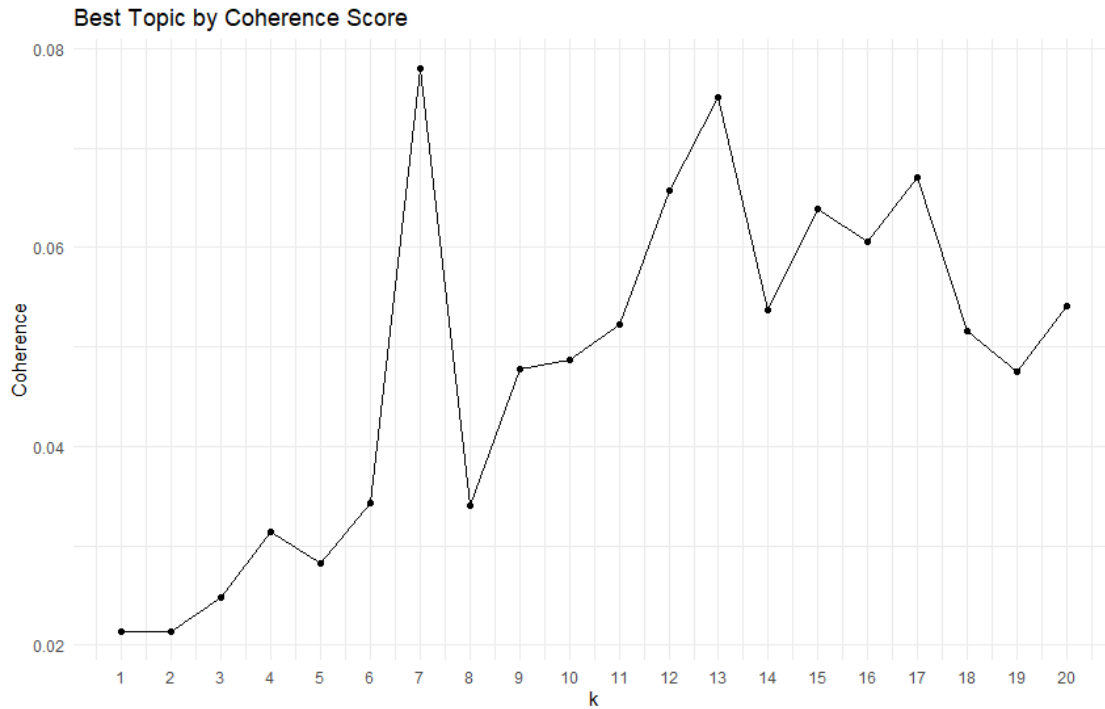


Figura 5.3: Coherence Score per K da 1 a 20

5.4 Risultati

Se applichiamo il metodo `posterior()` al modello calcolato tramite `LDA()` otteniamo le stime per θ , distribuzione dei tweet sui K topic e per ϕ , distribuzione di ogni topic sulle parole del vocabolario. Vediamo i 10 termini a cui sono associate le probabilità più alte all'interno di ciascun topic.

La composizione individuata (in ordine decrescente rispetto alla probabilità assegnata) è presentata in tabella 5.1.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7
1	governo	casa	casi	persone	fare	morti	emergenza
2	conte	quarantena	lombardia	ancora	stato	positivi	mascherine
3	tempo	fatto	mondo	virus	fatto	oggi	regione
4	fare	cina	essere	milano	tempo	decessi	situazione
5	fase	essere	contagi	ospedale	sempre	dati	medici
6	sempre	casa	marzo	nuovo	bene	tamponi	lavoro
7	regione	bene	contagio	positivo	roma	guariti	grazie
8	misure	senza	giorno	test	qui	contagi	aggiornamento
9	oggi	fare	aggiornamento	anni	emergenza	altri	qui
10	altri	virus	paese	video	italiani	giorno	altri

Tabella 5.1: Composizione dei topic individuati

Nonostante il basso numero di topic, non è comunque immediato notare una suddivisione semantica fra gli argomenti individuati. Risulta più semplice catalogare i topic 1, 2, 6, e 7, mentre per i topic 3, 4, e 5 l'interpretazione diventa più ostica. In particolare, i macroargomenti che possiamo individuare sono i seguenti:

1. **POLITICA:** il cluster di termini individuato per questo topic mostra un'associazione semantica evidente, soprattutto se ci concentriamo su quelli con probabilità più alte (i primi 5). Durante l'emergenza, i governi nazionali di tutto il globo hanno acquisito una popolarità e un'attenzione sempre maggiore, essendo costantemente sotto la luce dei riflettori per la gestione dell'emergenza. Ciò si è visto anche in Italia, specialmente se guardiamo alle conferenze stampa del Presidente del Consiglio; non sorprende dunque che Conte sia al secondo posto per probabilità assegnata. Come vedremo successivamente, l'interesse verso la politica è andato crescendo man mano che l'emergenza sanitaria è rientrata, motivo per cui all'interno di questo topic troviamo anche il termine "fase", sempre più utilizzato da aprile in poi in vista del parziale allentamento delle misure di contenimento del virus;
2. **PRIMI CONTAGI E LOCKDOWN:** come abbiamo osservato nei capitoli precedenti, l'incremento dell'attività su Twitter a febbraio è coinciso con la scoperta dei primi focolai in Lombardia e Veneto, i quali hanno aperto gli occhi ai cittadini sulla situazione in cui versava l'Italia e hanno reso l'imminente pericolo tangibile ("cina"). A seguito delle misure prese tra il 7 e il 10 marzo (lockdown nazionale, Italia "zona arancione") sono andati in tendenza messaggi di sensibilizzazione ai rischi della propagazione del virus e di invito al rispetto delle regole ("quarantena"). Sfortunatamente, in questo topic, sembra che questi due argomenti siano stati condensati assieme, cosa che avremmo preferito evitare per poterli valutare separatamente;

3. LOMBARDIA: l'epicentro dell'epidemia è stata oggetto di discussione per svariate ragioni, in particolare nei mesi di marzo e aprile per la drammaticità della situazione in cui versavano gli ospedali e per l'immane numero di vite umane perdute (si pensi alla vicenda di Bergamo);
4. Similmente al precedente i termini potrebbero fare riferimento a tematiche legate alla Lombardia, stavolta ponendo l'accento su questioni legate alla gestione sanitaria dell'epidemia: "nuovo", "ospedale", "milano" dovrebbero appunto riferirsi all'ospedale realizzato a tempo record in Fiera;
5. Il topic numero 5 purtroppo non fornisce alcun livello di interpretazione. Sembra siano stati aggregati termini generici con un'alta frequenza senza alcun collegamento semantico;
6. ANDAMENTO EPIDEMIOLOGICO: un'altra delle costanti che ci ha accompagnato durante i mesi di lockdown (e che prosegue ancora oggi, seppur generi meno interesse) è il consueto appuntamento delle 18 per la conferenza stampa della Protezione Civile, durante la quale viene diffuso il cosiddetto "bollettino", all'interno del quale sono riportati i numeri dei nuovi contagiati, dei decessi, dei guariti e dei tamponi effettuati;
7. SANITÀ: considerando che il periodo analizzato è stato quello più critico dal punto di vista sanitario, era lecito aspettarsi un topic che includesse termini riguardanti la crisi affrontata dal Sistema Sanitario Nazionale: dalla mancanza dei dispositivi di protezione individuale ("mascherine") ai contagi tra medici e infermieri. Questi ultimi, inoltre, sono stati protagonisti dei messaggi di solidarietà profusi dai cittadini per la dedizione e il coraggio mostrato nel corso delle settimane più difficili dell'emergenza.

5.4.1 Trend temporale dei topic

Proseguiamo l'analisi costruendo un dataframe all'interno del quale, per ogni giorno di rilevazione e per ogni topic individuato, calcoliamo la media delle document-topics distributions (ricavate dal parametro θ), visualizzando in questo modo qual è l'andamento delle probabilità stimate nel periodo in esame. Assegniamo, in più, ad ogni topic un nome sulla base dei primi 5 termini che lo caratterizzano.

Ci concentriamo sui topic numero 1, 2 e 7, più significativi dal punto di vista del trend generato, escludendo i topic 3, 4, e 5 per motivi di interpretabilità dei risultati e il topic 6 per la costante presenza del bollettino nei mesi considerati.

Per suddividere temporalmente i dati abbiamo riportato tre linee verticali che, come visualizzato precedentemente per la sentiment analysis, corrispondono al 5/03, al 28/03 e al 20/04.

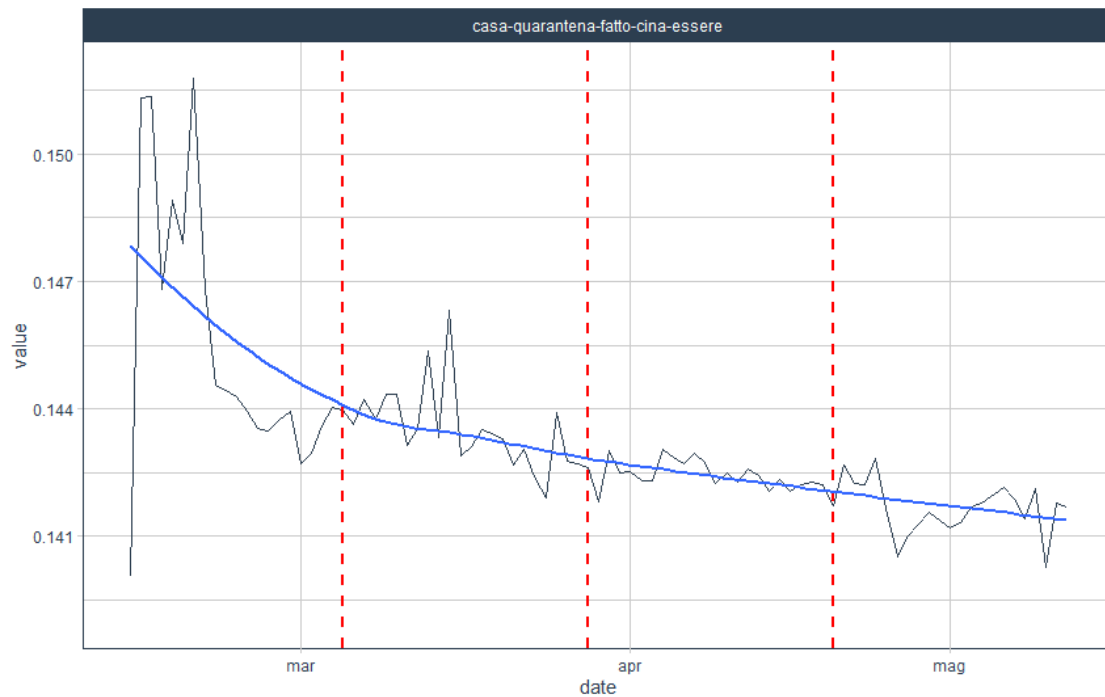


Figura 5.4: Trend topic 2

L'andamento temporale del topic 2 risente particolarmente dell'ondata di tweet seguita alla scoperta dei primi focolai. Si può comunque notare il trend discendente tra marzo, aprile e, in modo più consistente, maggio, indice del fatto che termini come "casa" e "quarantena" si concentrano principalmente tra il 10 e il 25 di marzo (trending dell'hashtag #iorestoacasa) e hanno avuto un effetto nell'alzare il valore della probabilità stimata.

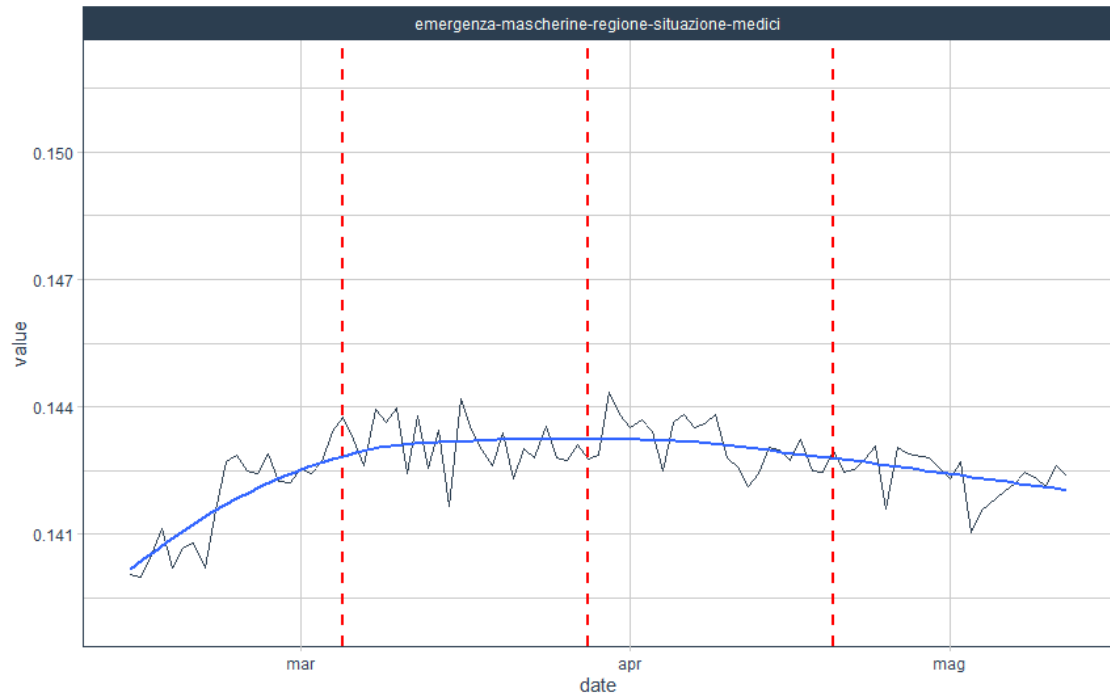


Figura 5.5: Trend topic 7

Per quanto riguarda il topic legato alla sanità notiamo un andamento altalenante ma complessivamente decrescente. Viene raggiunto un primo picco in corrispondenza del 13-14 marzo, il che è da attribuire alla centralità del tema sanitario in quei giorni: pensiamo ai messaggi di ringraziamento rivolti verso medici e infermieri, agli sforzi fatti per trovare nuovi posti letto e alla ricerca forsennata di mascherine e materiale medico. Si osserva, inoltre, un secondo picco tra la fine di marzo e i primi di aprile, ovvero proprio in quelle settimane in cui è stato raggiunto il numero massimo di malati attualmente positivi e la saturazione delle terapie intensive. Il trend va progressivamente decrescendo dal 20 aprile in seguito, proprio in corrispondenza dei giorni in cui il tema sanitario è entrato in secondo piano.

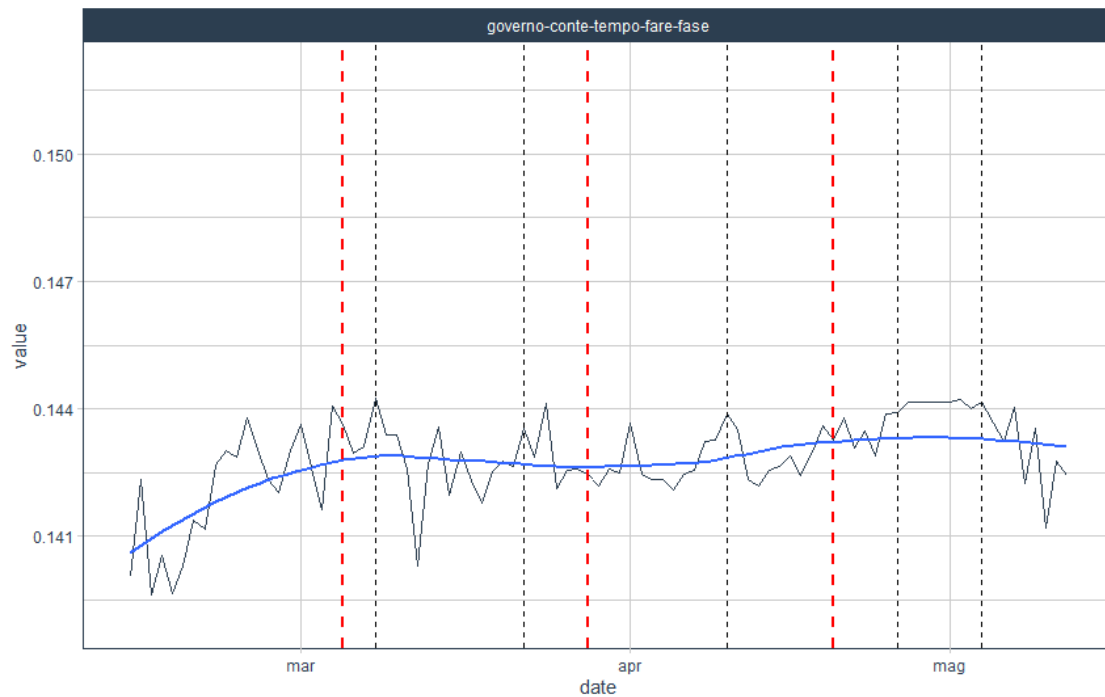


Figura 5.6: Trend topic 1

Infine, consideriamo il topic che ci dà più spunti per un confronto con quanto accaduto nel corso dei mesi, in particolare in relazione alle decisioni politiche che sono state prese tramite i famigerati DPCM. A tal proposito, abbiamo riportato sul grafico 5.6 delle linee che indicano momenti di svolta nella gestione dell'emergenza da parte del governo, a cui sono si collegano eventi che hanno generato un aumento dell'attività su Twitter, come ad esempio le conferenze stampa del Presidente del Consiglio. Vediamo nel dettaglio a cosa fanno riferimento e se possiamo trovare qualche associazione con le probabilità assegnate ai topic:

- 8 marzo: istituzione della zona arancione a livello regionale (Lombardia, Piemonte, Emilia-Romagna), la quale prevede delle limitazioni alla mobilità dei cittadini e la sospensione di alcune attività. Notiamo un picco della probabilità media stimata proprio in corrispondenza di questo giorno;
- 21 marzo: conferenza stampa del Presidente del Consiglio e entrata in vigore del nuovo DPCM. Vengono attuate misure più stringenti che prevedono la chiusura di tutte quelle attività non ritenute necessarie per la filiera produttiva italiana in relazione alla situazione contingente. Oltre a questo viene istituito il divieto di spostamento fra comuni se non per motivi di lavoro, salute o in caso di necessità. Graficamente sembra che abbia avuto più impatto la conferenza stampa del 24 marzo a seguito del Consiglio dei Ministri;

- 10 aprile: conferenza stampa del Presidente del Consiglio che proroga al 3 maggio le disposizioni per il contenimento del contagio. Si può vedere un innalzamento della probabilità stimata;
- 26 aprile: conferenza stampa del Presidente del Consiglio in cui viene annunciato il parziale allentamento delle misure di contenimento del virus a partire dal 4 maggio. Ancora una volta, l'innalzamento della spezzata coincide con il giorno della conferenza stampa. In generale, notiamo che nel mese di aprile, soprattutto dalla seconda metà del mese in poi, il tema politico è diventato sempre più centrale, com'era da aspettarsi visto il calo dei contagi e l'incombere della crisi economica;
- 4 maggio: inizio della Fase 2. Primo allentamento delle misure attuate con conseguente riapertura di alcune attività e maggiori libertà di spostamento. Viene raggiunto il livello maggiore proprio tra il 26 aprile e il 4 maggio, cosa che molto probabilmente è da attribuire alle attenzioni rivolte dai media verso l'uscita dal periodo più buio dell'emergenza e all'inizio della fase di convivenza col virus.

Conclusioni

Nel corso del presente scritto abbiamo applicato alcuni metodi afferenti al processo di elaborazione del linguaggio naturale su un corpus composto da tweet in lingua italiana riguardanti il coronavirus. La scelta di analizzare dei messaggi pubblicati durante le fasi più critiche della pandemia di COVID-19 in Italia è stata dettata fondamentalmente da due ragioni: in primis la volontà di effettuare un confronto tra i risultati ottenuti e la realtà di riferimento, che come abbiamo visto ci offre parecchi spunti di comparazione; in secondo luogo per presentare un'analisi che approfondisse tecniche di *text mining* che in letteratura, in riferimento al dominio trattato, sono state applicate prevalentemente su testi in lingua inglese.

Siamo partiti da un'intensa attività di *tweet mining*, portata a termine sia appoggiandoci alle API di Twitter che risalendo ad ulteriori osservazioni trasformando degli identificativi di tweet disponibili online, la quale ci ha permesso di costruire due dataset: il primo composto da circa 100.000 messaggi originali, il secondo con al suo interno oltre 130.000 retweet. Dopo aver eseguito le principali operazioni di pre-processing sui tweet in questione, abbiamo eseguito alcune analisi preliminari al fine di esplorare in superficie il contenuto informativo dei messaggi originali: fra queste troviamo, ad esempio, i termini e gli hashtag più utilizzati, gli utenti più citati e quelli più attivi, gli unigrammi più significativi per tf-idf, analisi quest'ultima che ci ha dato i primi riscontri dal punto di vista semantico rispetto all'evoluzione dell'epidemia nel nostro paese.

Ci siamo in seguito concentrati sui retweet, valutando il time-to-retweet, statistica che misura il tempo impiegato dall'autore del retweet a ripubblicare il corrispondente il messaggio originale, trovando un valore medio e uno mediano in linea con quanto calcolato da altri ricercatori su un corpus di retweet postati negli Stati Uniti. Indagando il contenuto dei retweet più rapidi abbiamo inoltre notato come questi ultimi si concentrassero nel periodo più sconvolgente per l'Italia, ovvero quello seguito alla scoperta dei primi focolai in Lombardia e Veneto. Ciò potrebbe essere tuttavia frutto di una delle prime complicazioni della nostra indagine, in quanto per risalire al tempo di pubblicazione dei messaggi retweettati abbiamo dovuto ridurre considerevolmente il numero delle osservazioni a nostra disposizione. I limiti dell'analisi si sono accentuati quando abbiamo cercato di applicare degli strumenti più complessi alla collezione di testi oggetto di studio: la *sentiment analysis* e il

topic modeling.

Per quanto riguarda la prima ci siamo limitati ad un approccio di tipo *lexicon based* applicato sugli unigrammi e abbiamo costruito una serie temporale che ci ha permesso di valutare l'andamento del sentiment durante i mesi considerati, riscontrando un sentimento prevalentemente negativo. Nonostante l'approccio utilizzato sia più "grezzo" rispetto ai sofisticati metodi di *machine learning*, abbiamo comunque notato delle connessioni tra gli eventi riguardanti la pandemia e le variazioni del livello calcolato, ad esempio il picco negativo è stato raggiunto in concomitanza con la scoperta dei primi contagi a Codogno e Vo'. L'analisi eseguita può essere sicuramente ampliata ed approfondita mediante l'approccio *machine learning*, per il quale sarebbe utile avere un training set di tweet precedentemente classificati.

L'applicazione del modello statistico generativo LDA ha invece portato a risultati più soddisfacenti: sebbene non tutti i topic individuati all'interno dei tweet siano di facile interpretazione, abbiamo notato come tramite questo algoritmo siano emersi dei macroargomenti distinti collegati alla pandemia: la sanità, l'andamento epidemiologico, i primi contagi e il lockdown, la gestione dell'emergenza da parte del governo. In merito a quest'ultima, la visualizzazione del trend temporale della probabilità stimata ci mostra chiaramente come questa aumenti in corrispondenza dei giorni in cui l'attenzione verso la compagine governativa è maggiore, si pensi per esempio alle conferenze stampa del Presidente del Consiglio e alle conseguenti misure prese per gestire l'emergenza.

Concludiamo così questa dissertazione, nella speranza di aver illustrato con chiarezza le caratteristiche dei metodi utilizzati e di aver contribuito, nel nostro piccolo, alla ricerca sul coronavirus.

Appendice A

Codice

CAPITOLO 1

```
1
2 libraries=c("streamR", "ROAuth", "twitterR", "tidyverse", "
  TextWiller", "tidytext", "textcat", "tm", "tau", "
  wordcloud2", "quanteda", "igraph", "ggraph", "hrbrthemes",
  "scales", "topicmodels", "tidyquant")
3 lapply(libraries, require, character.only = TRUE)
4
5 #API connection
6 consumerKey='****'
7 consumerSecret='****'
8 accessToken='****'
9 accessTokenSecret= '****'
10 setup_twitter_oauth(consumerKey, consumerSecret, accessToken,
  accessTokenSecret)
11
12 search = function(searchterm) {
13   list = searchTwitter(searchterm, n=2000, lang="it")
14   df = twListToDF(list)
15   df = df[, order(names(df))]
16   df$created <- strptime(df$created, '%Y-%m-%d')
17   if (file.exists(paste(searchterm, '_stack.csv'))==FALSE)
18     write.csv(df, file=paste(searchterm, '_stack.csv'), row.
19       names=F)
20
21   stack = read.csv(file=paste(searchterm, '_stack.csv'))
22   stack = rbind(stack, df)
23   stack = subset(stack, !duplicated(stack$text))
24   write.csv(stack, file=paste(searchterm, '_stack.csv'), row.
25     names=F)
26 }
```



```

24 search("coronavirus")
25
26
27 #Hydrator
28 for (i in 1:length(vec)){
29   temp=read.csv(vec[i], sep=",") #vector containing names
    of files to be hydrated
30   temp=as_tibble(temp)
31
32   it=temp %>% filter(Language == "it")%>% select((Tweet_ID))
33   it=as.data.frame(it)
34   out=write.table(format(it, digits=19), vec[i], col.names =
    F, row.names = F, quote=F, sep="\t")
35 }
36
37 for (i in 1:length(vec2)){
38   temp=read.csv(vec2[i], sep=",") #vector containing names
    of hydrated files
39   temp=as_tibble(temp)
40   corpus=bind_rows(corpus, temp)
41
42 }
43
44 corpus=as_tibble(corpus)
45 tweet=corpus %>% filter(lang=="it")
46 tweet=tweet%>% distinct(id, .keep_all = T)
47 sort(unique(strftime(tweet$created_at, format="%Y-%m-%d"))) #
    range from 14-02 to 12-05
48
49 #convert dates to UTC format
50 chr=tweet%>%select(created_at)
51 created_day=vector()
52 created_hour=vector()
53 convert = function(chr, day, hour) {
54 for (i in 1:dim(chr)[1]){
55   data=as.character(chr[i,])
56   s1 = unlist(strsplit(data, split=' ', fixed=TRUE))
57   if (s1[2] == "Feb"){
58     month="02"
59   }
60   if (s1[2] == "Mar"){
61     month="03"
62   }
63   if (s1[2] == "Apr"){
64     month="04"
65   }

```

```

66     if (s1[2] == "May"){
67         month="05"
68     }
69     day[i]=paste( s1[6], month,s1[3], sep="-")
70     hour[i]=s1[4]
71 }
72 }
73 convert(chr, created_day, created_hour)
74
75 tweet$created_hour=created_hour
76 tweet$created_day=created_day
77 tweet=tweet %>% select(id, favorite_count, retweet_count,
78     text, screen_name, created_at, latitude, longitude,
79     created_hour, created_day)
80 tweet=bind_rows(tweet, coronavirus_stack)
81
82 retweet=corpus%>%filter(!is.na(retweet_id))
83 retweet=retweet%>% distinct(id, .keep_all = T)
84
85 #pre-processing
86
87 tweet$text=normalizzaTesti(as_vector(tweet$text),
88     normalizzacaratteri=TRUE,tolower=TRUE,perl=FALSE,fixed=
89     FALSE)
90
91 tweet$text=gsub("'", " ", tweet$text)
92 Textprocessing <- function(x)
93 {gsub("http[[:alnum:]]*",'', x)
94   gsub('http\\S+\\s*', '', x) ## Remove URLs
95   gsub('#\\S+', '', x) ## Remove Hashtags
96   gsub('@\\S+', '', x) ## Remove Mentions
97   gsub('[:cntrl:]', '', x) ## Remove Controls and special
98     characters
99   gsub("\\d", '', x) ## Remove Controls and special
100     characters
101   gsub('[:punct:]', '', x) ## Remove Punctuations
102   gsub("^[:space:]*","",x) ## Remove leading whitespaces
103   gsub("[:space:]*$","",x) ## Remove trailing whitespaces
104   gsub(' +',' ',x) ## Remove extra whitespaces
105 }
106 tweet$text=Textprocessing(tweet$text)
107

```

```

102 tweet$text=normalizzaTesti(tweet$text, remove=c("coronavirus"
, "covid", "fef", "ffb", "ddc", "wwwurlwww", "covid", "covid19",
, "dfc", "ffa", "fda", "fbc", "def", "coronavirusitaly", "
coronavirusitalia", "covid2019", "covid19italia", "
coronaviriusitalia", "coronaviruslombardia", "covid19", "
coronavirusitalla", "coronavirusitalia", "
coronarvirusitalia"))
103
104
105 tweet$text=iconv(tweet$text, "latin1", "ASCII", sub="")
106 tweet$text=gsub(" *\\b[[:alpha:]]{1,2}\\b *", " ", tweet$text
)
107 tweet = tweet %>% filter(text != "")
108
109
110 tweet$lan=textcat(tweet$text, ECIMCI_profiles)
111 table(tweet$lan)
112 tweet=tweet %>%filter(lan %in% c("it", "la"))

```

CAPITOLO 2

```

1
2 corpus=Corpus(VectorSource(tweet$text))
3 dtm=DocumentTermMatrix(corpus, control = list( stemming =
FALSE, stopwords=itastopwords, minWordLength = 3,
removeNumbers = FALSE, removePunctuation = TRUE, bounds=
list(local = c(1, Inf)) ))
4
5 text_df=tweet %>% unnest_tokens(word, text)
6
7 #top retweets
8 tweet %>%
9   arrange(-retweet_count) %>%
10   select(screen_name, text, retweet_count) %>%
11   top_n(10)
12
13 #top mentions
14 tweet %>%
15   unnest_tokens(mentions, text, "tweets", to_lower = FALSE) %
>%
16   filter(str_detect(mentions, "^@")) %>%
17   count(mentions, sort = TRUE) %>%
18   top_n(10)
19
20 #top hashtags

```

```

21 tweet %>%
22   unnest_tokens(hashtag, text, "tweets", to_lower = TRUE) %>%
23   filter(str_detect(hashtag, "^#"), !hashtag %in% c("#
     coronavirus", "#covid", "#covid19", "#coronarvirusitalia"
     , "#coronavirussitalia", "#coronvirusitalia", "#
     coronavirusitaly", "#coronavirusitalia", "#covid2019", "
     #covid19italia", "#covid???19", "#coviditalia")) %>%
24   count(hashtag, sort = TRUE) %>%
25   top_n(20)
26
27 #keywords
28 tweet %>% filter(str_detect(text, "contagi "))
29 tweet %>% filter(str_detect(text, "mascherine"))
30
31 #users
32 tweet %>% group_by(screen_name) %>% count(sort=T) %>% top_n
     (20)
33
34 #tokens
35 wordcloudDF=text_df %>% count(word, sort=T)
36 wordcloud2(wordcloudDF, color = "skyblue")
37 bigrams=textcnt(tweet$text, method="string", n=2L, split="[:
     blank:]")
38 sort(bigrams, decreasing=TRUE)[1:30]
39 trigrams=textcnt(tweet$text, method="string", n=3L, split="[:
     blank:]")
40 sort(trigrams, decreasing=TRUE)[1:30]
41
42 #tf-idf
43 feb=tweet %>% filter(as.Date(created_day) > "2020-02-13" & as
     .Date(created_day) < "2020-03-08")
44 mar=tweet %>% filter(as.Date(created_day) > "2020-03-07" & as
     .Date(created_day) < "2020-04-02")
45 apr=tweet %>% filter(as.Date(created_day) > "2020-04-01" & as
     .Date(created_day) < "2020-04-25")
46 mag=tweet %>% filter(as.Date(created_day) > "2020-04-24" & as
     .Date(created_day) < "2020-05-15")
47
48 testo1=tweet %>% inner_join(feb) %>% mutate(month= "feb")
49 testo2=tweet %>% inner_join(mar) %>% mutate(month= "mar")
50 testo3=tweet %>% inner_join(apr) %>% mutate(month= "apr")
51 testo4=tweet %>% inner_join(mag) %>% mutate(month= "mag")
52
53 tweet_words = bind_rows(testo1, testo2, testo3, testo4) %>%
54   unnest_tokens(word, text) %>%
55   count(month, word, sort = TRUE)

```

```

56
57 total_words <- tweet_words %>%
58   group_by(month) %>%
59   summarize(total = sum(n))
60 tweet_words <- left_join(tweet_words, total_words)
61
62 tweet_words <- tweet_words %>%
63   bind_tf_idf(word, month, n)
64
65 idf=tweet_words %>%
66   select(-total) %>%
67   arrange(desc(tf_idf))
68
69 tweet_words %>%
70   arrange(desc(tf_idf)) %>%
71   mutate(word = factor(word, levels = rev(unique(word)))) %>%
72   group_by(month) %>%
73   top_n(15) %>%
74   ungroup() %>%
75   ggplot(aes(word, tf_idf)) +
76   geom_col(show.legend = FALSE) +
77   labs(x = NULL, y = "tf-idf") +
78   facet_wrap(~month, ncol = 2, scales = "free") +
79   coord_flip()

```

CAPITOLO 3

```

1
2 #original messages extraction
3 retweet=retweet %>% rename(screen_name=user_screen_name.1, OP=
4   retweet_id) %>% filter (!is.na(OP)) %>% distinct()
5
6 id=select(retweet, OP)
7 id$OP=as.character(id$OP)
8 id=as.data.frame(id)
9 write.table(format(id, digits=19), "retweet.txt", col.names =
10   F, row.names = F, quote=F, sep="\t")
11
12 originals=as_tibble(hydrated_retweet)
13 originals=originals %>% distinct(id, .keep_all = TRUE)
14 originals=originals %>% rename( OP = id)
15 originals$OP=as.double(originals$OP)
16 join=inner_join(originals,retweet, by = "OP")
17 join %>% select(created_at.x, created_at.y)

```

```

17 join %>% select(text.x, text.y)
18
19
20 #response time
21 created_orig=join %>% select(created_at.x)
22 created_ret=join %>% select(created_at.y)
23
24 created_day_orig=vector()
25 created_hour_orig=vector()
26 created_day_ret=vector()
27 created_hour_ret=vector()
28 convert(created_orig, created_day_orig, created_hour_orig)
29 convert(created_ret, created_day_ret, created_hour_ret)
30 join$created_day_ret=created_day_ret
31 join$created_hour_ret=created_hour_ret
32 join$created_day_orig=created_day_orig
33 join$created_hour_orig=created_hour_orig
34 join$diff=as.numeric(difftime(strptime(paste(join$created_day
35 _orig, join$created_hour_orig),"%Y-%m-%d %H:%M:%S"),
                                strptime(paste(join$created_day
                                                _ret, join$created_hour_ret)
                                ,"%Y-%m-%d %H:%M:%S"))))
36 median(join$diff, na.rm=T) #9820 seconds, 2.7h
37 mean(join$diff, na.rm=T) #50432 secpnds, 14h
38 join2=join%>%filter(is.na(diff) == FALSE)
39
40 ggplot(data=join2, mapping=aes(x=(abs(diff)/3600)))+
41   stat_density(aes(y=..count..), fill="blue", alpha=0.3) +
42   scale_x_continuous(breaks=c
43     (0,0.2,0.5,1,2.7,5,14,24,48,100,250,500,1000,5000,10000)
44     , trans="log1p", expand=c(0,0)) +
45   scale_y_continuous(breaks=c(0, 1000, 2000,3000,4000,10000),
46     expand=c(0,0)) +
47   theme_bw()+ xlab("hours")
48
49 #rapid retweets
50 less_than=join%>% filter(abs(diff) < 10000) %>% distinct(OP,
51   .keep_all = TRUE)
52 feb=less_than %>% filter(as.Date(created_day_ret) == "
53   2020-02-19")
54 mar=less_than %>% filter(as.Date(created_day_ret) > "
55   2020-02-19" & as.Date(created_day_ret) < "2020-02-25")
56 testo1=less_than %>% filter(user_description.x != "") %>%
57   inner_join(feb) %>% mutate(month= "1")
58 testo2=less_than %>% filter(user_description.x != "")%>%inner
59   _join(mar) %>% mutate(month= "2")

```

```

52
53 tweet_words=bind_rows(testo1,testo2) %>% unnest_tokens(word,
    text.x) %>% count(month, word,sort=T)
54 total_words <- tweet_words %>%
55   group_by(month) %>%
56   summarize(total = sum(n))
57 tweet_words <- left_join(tweet_words, total_words)
58 tweet_words <- tweet_words %>%
59   select(-total) %>%
60   bind_tf_idf(word, month,n)
61 idf=tweet_words %>%
62   arrange(desc(tf_idf))
63
64 tweet_words %>%
65   arrange(desc(tf_idf)) %>%
66   mutate(word = factor(word, levels = rev(unique(word)))) %>%
67   group_by(month) %>%
68   top_n(15) %>%
69   ungroup() %>%
70   ggplot(aes(word, tf_idf, col = "blue")) +
71   geom_col(show.legend = FALSE) +
72   labs(x = NULL, y = "tf-idf") +
73   facet_wrap(~month, ncol = 2, scales = "free") +
74   coord_flip()
75
76
77 bigrams=bind_rows(testo1,testo2) %>% unnest_tokens(bigram,
    user_description.x, token="ngrams",n=2)
78 bigrams_separated <- bigrams %>%
79   separate(bigram, c("word1", "word2"), sep = " ")
80
81 bigrams_united <- bigrams_separated %>%
82   unite(bigram, word1, word2, sep = " ")
83
84 bigram_tf_idf <- bigrams_united %>%
85   count(month, bigram) %>%
86   bind_tf_idf(bigram, month, n) %>%
87   arrange(desc(tf_idf))
88
89 bigram_tf_idf %>%
90   arrange(desc(tf_idf)) %>%
91   mutate(word = factor(bigram, levels = rev(unique(bigram))))
    %>%
92   group_by(month) %>%
93   top_n(15) %>%
94   ungroup() %>%

```

```

95 ggplot(aes(x= reorder(bigram,tf_idf),tf_idf, col="blue")) +
96 geom_col(show.legend = FALSE) +
97 labs(x = NULL, y = "tf-idf") +
98 facet_wrap(~month, ncol = 2, scales = "free") +
99 coord_flip()
100
101 #visualization
102
103 top_orig=originals%>% group_by(user_screen_name) %>% count(
104   sort=T) %>% top_n(50)
105
106 top_ret=join%>% group_by(user_screen_name.y) %>% count(sort=T
107   ) %>% top_n(50)
108
109 join %>% filter(user_screen_name.x != "") %>%
110   filter(user_screen_name.y != "") %>% filter(user_screen_
111     name.x %in% top_orig[2:100,]$user_screen_name) %>%
112   filter(user_screen_name.y %in% top_ret[1:100,]$user_screen_
113     name.y) %>%
114   select(user_screen_name.x, user_screen_name.y) %>%
115   filter(!is.na(user_screen_name.y)) %>%
116   graph_from_data_frame() -> rt_g
117
118 V(rt_g)$node_label <- unname(names(V(rt_g)))
119
120 # Size of node
121 V(rt_g)$node_size <- unname(ifelse(degree(rt_g)[V(rt_g)] > 1,
122   degree(rt_g), 1))
123
124 # Adjust angle of label based on position
125 nIds <- length(V(rt_g))
126 V(rt_g)$Id <- seq(1:nIds)
127 V(rt_g)$label_angle <- 90 - 360 * V(rt_g)$Id / nIds
128 V(rt_g)$hjust <- ifelse(V(rt_g)$label_angle < -90, 1, 0)
129
130 # Flip text depending on what side of the plot it is on
131 V(rt_g)$angle <- ifelse(V(rt_g)$label_angle < -90, V(rt_g)$
132   label_angle+180, V(rt_g)$label_angle)
133
134 p <- ggraph(rt_g, layout = 'linear', circular = TRUE) +
135   geom_edge_arc(aes(alpha=..index..)) +
136   geom_node_point(aes(x = x*1.07, y=y*1.07, size=node_size,
137     alpha=0.2)) +
138   geom_node_text(aes(x=x*1.15, y=y*1.15,label=node_label,
139     angle=angle, hjust=hjust),

```



```

132         color="dodgerblue", size=2.7, family=font_rc
133         ) +
134     coord_fixed() +
135     labs(title="#coronavirus Relationships", subtitle="Darker
136           edges == more retweets. Node size == larger degree") +
137     theme_graph(base_family=font_rc) +
138     theme(legend.position="none") +
139     expand_limits(x = c(-1.3, 1.3), y = c(-1.3, 1.3))
p

```

CAPITOLO 4

```

1
2 #lexicon based unigram classification
3 sent=sentiment(as.vector(text_df$word))
4 pos=subset(sent,sent==1)
5 temp=bind_rows(pos)
6 pos=gather(temp, key="word")
7
8 neg=subset(sent,sent==-1)
9 temp=bind_rows(neg)
10 neg=gather(temp, key="word")
11
12 pos_df=text_df %>%
13   inner_join(pos) %>%
14   count(word, sort = TRUE)%>%
15   mutate(sentiment="positive")
16 neg_df=text_df %>%
17   inner_join(neg) %>%
18   count(word, sort = TRUE)%>%
19   mutate(sentiment="negative")
20 word_counts=full_join(pos_df, neg_df) %>% arrange(desc(n))
21
22 word_counts %>%
23   group_by(sentiment) %>%
24   top_n(30, n) %>%
25   ungroup() %>%
26   mutate(word = reorder(word, n)) %>%
27   ggplot(aes(word, n, fill = sentiment)) +
28   geom_col(show.legend = FALSE) +
29   facet_wrap(~sentiment, scales = "free_y") +
30   labs(y = "Contribution to sentiment",
31        x = NULL) +
32   coord_flip()

```

```

33
34
35 #daily sentiment index
36 date=sort(unique(tweet$created_day))
37 vec=data.frame(sentiment=integer(), date=as.Date(character()),
38               ,stringsAsFactors=FALSE)
39 wordcounts=select(word_counts, word, sentiment)
40
41 for (i in 1:length(date)){
42   temp = tweet %>% filter(as.Date(created_day) == date[i])
43   tokens <- data_frame(temp) %>% unnest_tokens(word, text)
44
45   sentiment <- tokens %>%
46     inner_join(wordcounts) %>%
47     count(sentiment) %>% #
48     spread(sentiment, n, fill = 0) %>%
49     mutate(sentiment = positive - negative)
50
51   df=data.frame(sentiment$sentiment,as.Date(temp$created_day
52     [1]))
53   colnames(df) = c("sentiment", "date")
54   vec=rbind(vec, df)
55 }
56
57 vec$sentiment=(vec$sentiment-max(vec$sentiment))/(max(vec$
58   sentiment)-min(vec$sentiment))
59 ggplot(vec, aes(x = date, y = sentiment)) +
60   geom_smooth(method = "auto") +
61   geom_point()+
62   ylim(-2.2, -0.8)

```

CAPITOLO 5

```

1 tweet_corpus <- corpus(testo$text)
2
3
4 corpus_tokens <- tweet_corpus %>%
5   tokens(remove_punct = TRUE, remove_numbers = TRUE, remove_
6     symbols = TRUE) %>%
7   tokens_tolower()
8
9 tweet_collocations <- textstat_collocations(corpus_tokens,
10   min_count = 10)
11 tweet_collocations <- tweet_collocations[1:300, ]

```

```

11 corpus_tokens <- tokens_compound(corpus_tokens, tweet_
    collocations)
12
13 DTM <- corpus_tokens %>%
14   tokens_remove("") %>%
15   dfm() %>%
16   dfm_trim(min_docfreq = 0.01, max_docfreq = 0.6, docfreq_
     type = "prop")
17
18 dim(DTM)
19 top10_terms <- c( "non", "italia", "emotelove", "emergenza",
    "ora", "solo", "ecco", "cosa", "perch", "cos", "dopo", "
    prima")
20
21 DTM <- DTM[, !(colnames(DTM) %in% top10_terms)]
22 sel_idx <- rowSums(DTM) > 0
23 DTM <- DTM[sel_idx, ]
24 textdata <- tweet[sel_idx, ]
25
26 topicModel <- LDA(DTM, k=7, method="Gibbs", control=list(iter
    = 500, verbose = 25))
27 tmResult <- posterior(topicModel)
28 attributes(tmResult)
29
30 beta <- tmResult$terms
31 theta <- tmResult$topics
32
33 terms(topicModel, 10)
34
35 df=data.frame(id=names(topics(topicModel)),
36   date=as.Date(textdata$created_day))
37 dft <- cbind(df,posterior(topicModel)$topics)
38
39 M <- gather(dft,topic,value,-id,-date) %>%
40   group_by(topic,date) %>%
41   summarize(value=mean(value))
42
43 top5termsPerTopic <- terms(topicModel, 5)
44 topicNames <- apply(top5termsPerTopic, 2, paste, collapse="-"
    )
45
46 nomi=as_tibble(topicNames) %>% transmute(topic=row_number(),
    nome=value)
47
48 nomi=nomi%>%mutate(topic = as.character(topic))
49 M=M%>%left_join(nomi)

```

```

50 M1=M%>%filter(topic==1)
51
52
53 p=ggplot(M1,aes(x=date,y=value)) +
54   geom_line(color = palette_light()[[1]], size=.6) +facet_
55     wrap(~nome)+ theme_tq()
56 p + geom_vline(xintercept = as.Date("2020-03-08"), linetype="
57   dashed") +
58   geom_vline(xintercept = as.Date("2020-03-05"), linetype="
59     dashed", color="red", size=1) +
60   geom_vline(xintercept = as.Date("2020-03-28"), linetype="
61     dashed", color="red", size=1) +
62   geom_vline(xintercept = as.Date("2020-04-20"), linetype="
63     dashed", color="red", size=1) +
64   geom_vline(xintercept = as.Date("2020-03-22"), linetype="
65     dashed") +
66   geom_vline(xintercept = as.Date("2020-04-10"), linetype="
67     dashed") +
68   geom_vline(xintercept = as.Date("2020-04-26"), linetype="
69     dashed") +
70   geom_vline(xintercept = as.Date("2020-05-04"), linetype="
71     dashed") + geom_smooth(method="loess", se=F, size=1)+
72   ylim(0.139,0.152)
73
74 M2=M%>%filter(topic==2)
75
76
77 pp=ggplot(M2,aes(x=date,y=value)) +
78   geom_line(color = palette_light()[[1]], size=.6) +facet_
79     wrap(~nome)+ theme_tq()
80
81 pp+
82   geom_vline(xintercept = as.Date("2020-03-05"), linetype="
83     dashed", color="red", size=1) +
84   geom_vline(xintercept = as.Date("2020-03-28"), linetype="
85     dashed", color="red", size=1) +
86   geom_vline(xintercept = as.Date("2020-04-20"), linetype="
87     dashed", color="red", size=1) +
88   geom_smooth(method="loess", se=F, size=1)+   ylim
89     (0.139,0.152)
90
91
92 M3=M%>%filter(topic==7)
93
94
95 pp=ggplot(M3,aes(x=date,y=value)) +

```

```
80 geom_line(color = palette_light()[[1]], size=.6) +facet_
    wrap(~nome)+ theme_tq()
81
82 pp+
83 geom_vline(xintercept = as.Date("2020-03-05"), linetype="
    dashed", color="red", size=1) +
84 geom_vline(xintercept = as.Date("2020-03-28"), linetype="
    dashed", color="red", size=1) +
85 geom_vline(xintercept = as.Date("2020-04-20"), linetype="
    dashed", color="red", size=1) +
86 geom_smooth(method="loess", se=F, size=1)+ ylim
    (0.139,0.152)
```


Appendice B

Italian Stopwords

	1	2	3	4
1	abbia	è	ne	stavamo
2	abbiamo	ebbe	nè	stavano
3	abbiano	ebbero	né	stavate
4	abbiate	ebbi	negl	stavi
5	ad	ecc	negli	stavo
6	agl	ed	nei	stemmo
7	agli	era	nel	stesse
8	ai	erano	nell	stessero
9	al	eravamo	nella	stessi
10	all	eravate	nelle	stessimo
11	alla	eri	nello	steste
12	alle	ero	noi	stesti
13	allo	essendo	nostra	stette
14	anche	etc	nostre	stettero
15	anziche	fa	nostri	stetti
16	anziché	faccia	nostro	stia
17	avemmo	facciamo	ogni	stiamo
18	avendo	facciano	per	stiano
19	avesse	facciate	perche	stiate
20	avessero	faccio	perché	sto
21	avessi	facemmo	però	su
22	avessimo	facendo	po	sua
23	aveste	facesse	pò	sue
24	avesti	facessero	poi	sugl
25	avete	facessi	può	sugli
26	aveva	facevamo	qual	sui

	1	2	3	4
27	avevamo	faceste	quale	sul
28	avevano	facesti	quali	sull
29	avevate	faceva	quando	sulla
30	avevi	facevamo	quanta	sulle
31	avevo	facevano	quante	sullo
32	avrà	facevate	quanti	suo
33	avrai	facevi	quanto	suoi
34	avranno	facevo	quell	ti
35	avrebbe	fai	quella	tra
36	avrebbero	fanno	quelle	tu
37	avrei	farà	quelli	tua
38	avremmo	farai	quello	tue
39	avremo	faranno	quest	tuo
40	avreste	farebbe	questa	tuoi
41	avresti	farebbero	queste	tutti
42	avrete	farei	questi	tutto
43	avrò	faremmo	questo	un
44	avuta	faremo	sa	una
45	avute	fareste	sarà	uno
46	avuti	faresti	sarai	vabbè
47	avuto	farete	saranno	vi
48	che	farò	sarebbe	via
49	chi	fece	sarebbero	voi
50	chissà	fecero	sarei	vostra
51	ci	feci	saremmo	vostre
52	ciò	fosse	saremo	vostri
53	cmq	fossero	sareste	vostro
54	coi	fossi	saresti	xche
55	col	fossimo	sarete	xché
56	come	foste	sarò	xké
57	comunque	fosti	se	a
58	con	fu	sei	b
59	contro	fui	si	c
60	cose	fummo	sia	d
61	così	furono	siamo	e
62	cui	già	siano	f
63	da	gli	siate	g
64	dà	ha	siete	h
65	dagl	hai	sono	i
66	dagli	hanno	st	j

	1	2	3	4
67	dai	ho	sta	k
68	dal	il	stai	l
69	dall	in	stando	m
70	dalla	io	stanno	n
71	dalle	la	starà	o
72	dallo	le	starai	p
73	degl	lei	staranno	q
74	degli	li	starebbe	r
75	dei	lo	starebbero	s
76	del	loro	starei	t
77	dell	lui	staremmo	u
78	della	ma	staremo	v
79	delle	mi	stareste	w
80	dello	mia	staresti	x
81	dello	mie	starete	y
82	dello	miei	starò	z
83	dello	mio	stava	

Bibliografia

- [1] Catherine Ordun et al. Covid resources. <https://github.com/nudro/covid-resources>, 2020.
- [2] B. Wang and J. Zhuang. Crisis information distribution on twitter: a content analysis of tweets during hurricane sandy, 2017.
- [3] Matteo Redaelli Dario Solari, Livio Finos. Textwiller: Collection of functions for text mining, specially devoted to the italian language, 2013.
- [4] Emily Chen et al. Covid-19-tweetids. <https://github.com/echen102/COVID-19-TweetIDs>, 2020.
- [5] Christian Lopez and Malolan Vasu. Covid-19 multilanguage tweets dataset. https://github.com/lopezbec/COVID19_Tweets_Dataset, 2020.
- [6] Kurt Hornik Patrick Mair Johannes Rauch Wilhelm Geiger Christian Buchta and Ingo Feinerer. The textcat package for n-gram based text categorization in r. <https://www.jstatsoft.org/article/view/v052i06>, 2013.
- [7] Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D’Agostino McGowan, Romain François, Garrett Grolmund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohnske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani. Welcome to the tidyverse, 2019.
- [8] Julia Silge and David Robinson. tidytext: Text mining and analysis using tidy data principles in r. <http://dx.doi.org/10.21105/joss.00037>, 2016.
- [9] Hadley Wickham, Romain François, Lionel Henry, and Kirill Müller. dplyr: A grammar of data manipulation. <https://CRAN.R-project.org/package=dplyr>, 2018. R package version 0.7.6.

- [10] Hadley Wickham Lionel Henry. tidy: Easily tidy data with 'spread()' and 'gather()' functions. <https://CRAN.R-project.org/package=tidy>, 2018. R package version 0.8.1.
- [11] Hadley Wickham. ggplot2: Elegant graphics for data analysis. <https://ggplot2.tidyverse.org>, 2016.
- [12] Ingo Feinerer and Kurt Hornik. tm: Text mining package. <https://CRAN.R-project.org/package=tm>, 2019. R package version 0.7-7.
- [13] Guan-tin Chien Dawei Lang. wordcloud2: Create word cloud by 'htmlwidget'. <https://github.com/lchiffon/wordcloud2>, 2018.
- [14] Christian Buchta Kurt Hornik Ingo Feinerer and David Meyer. tau: Text analysis utilities. <http://cran.r-project.org/package=tau>, 2012. R package version 0.0-15.
- [15] K. Lee D. Palsetia R. Narayanan M. M. A. Patwary A. Agrawal and A. Choudhary. Twitter trending topic classification, 2011. IEEE, 2011, pp. 251–258.
- [16] P. Barnaghi P. Ghaffari and J. G. Breslin. Opinion mining and sentiment polarity on twitter and correlation between events and sentiment, 2016. IEEE, 2016, pp. 52–57.
- [17] Gabor Csardi and Tamas Nepusz. The igraph software package for complex network research, 2006.
- [18] Thomas Lin Pedersen. An implementation of grammar of graphics for graphs and networks, 2020.
- [19] E. Spiro C. Irvine C. DuBois and C. Butts. Waiting for a retweet: modeling waiting times in information propagation, 2012. NIPS workshop of social net-works and social media conference. <http://snap.stanford.edu/social2012/papers/spiro-dubois-butts.pdf>. Accessed, vol. 12.
- [20] Catherine Ordun Sanjay Purushotham and Edward Raff. Exploratory analysis of covid-19 tweets using topic modeling, umap, and digraphs. <https://arxiv.org/abs/2005.03082>, 2020.
- [21] L. Zhang L. Xu and W. Zhang. Social media as amplification station: factors that influence the speed of online public response to health emergencies, 2017. AsianJournal of Communication, vol. 27, no. 3, pp. 322–338.
- [22] Martin Müller Marcel Salathé and Per E Kummervold. Covid-twitter-bert: A natural language processing model to analyse covid-19 content on twitter., 2020. arXiv preprint arXiv:2005.07503.

- [23] Andrew Y Ng David M Blei and Michael I Jordan. Latent dirichlet allocation, 2003.
- [24] M. Steyvers and T. Griffiths. Latent semantic analysis: A road to meaning, 2007.
- [25] Bettina Grün and Kurt Hornik. topicmodels: An R package for fitting topic models, 2011.
- [26] W. Zhao J. J. Chen R. Perkins Z. Liu W. Ge Y. Ding and W. Zou. A heuristic approach to determine an appropriate number of topics in topic modeling, 2015.
- [27] W. Zhang Y. Cui and T. Yoshida. En-lda: An novel approach to automatic bug report assignment with entropy optimized latent dirichlet allocation, 2017.
- [28] M. Röder A. Both and A. Hinneburg. Exploring the space of topic coherence measures, 2015.
- [29] Tommy Jones and William Doane. textminer: Functions for text mining and topic modeling. <http://cran.r-project.org/package=textmineR>, 2019. R package version 3.0.2.
- [30] C. Angioni. Machine learning e fattore umano nella sentiment analysis. il caso starbucks a milano, 2017.
- [31] P. Di Gennaro. Due approcci alla sentiment polarity classification di tweet per la lingua italiana, 2016.
- [32] F. Ferraccioli. Topic modeling, dietro le quinte: modelli grafici diretti e indiretti, 2016.