Il negazionismo climatico: un'analisi testuale della letteratura pseudoscientifica e dei tweet dei complottisti

Report tecnico del progetto di Web Analytics e
Analisi Testuale

A cura di Alessio Ena

Relatore: Prof. Marco Ortu

Università degli studi di Cagliari
Anno Accademico 2022/2023

CONTENTS

1 Introduzione 2
2 Data Collection 3
2.1 Selezione dei libri . . . 3
2.2 Tweets collection . . . 3
2.3 Reverse geocoding e visualizzazione geografica dei tweet . . . 5
3 Data Cleaning 8
3.1 Pulizia dei tweet . . . 8
3.2 Pulizia del corpus estratto dai libri . . . 9
3.3 Unificazione del corpus dei libri . . . 11
4 Data Analysis 13
4.1 Data esploration . . . 13
4.2 Topic modelling . . . 17
4.3 Sentiment analysis . . . 23
5 Conclusioni 26

LIST OF FIGURES

Figure 1 Query di ricerca . . . 4
Figure 2 Distribuzione geografica dei tweet . . . 6
Figure 3 Word clouds dei libri . . . 14
Figure 4 Word clouds dei libri con riassegnazione dei colori . . . 15
Figure 5 Word cloud dei tweet . . . 16
Figure 6 Topic Coherence corpus libri . . . 18
Figure 7 Topic Perplexity corpus libri . . . 19
Figure 8 LDA corpus libri . . . 19
Figure 9 Topic Coherence corpus libri . . . 20
Figure 10 Topic Perplexity corpus libri . . . 20
Figure 11 Topic 1 LDA tweet . . . 21
Figure 12 Topic 2 LDA tweet . . . 21
Figure 13 Topic 3 LDA tweet . . . 22
Figure 14 Sentiment analysis of tweets . . . 24
Figure 15 Emotion analysis of tweets . . . 25

LIST OF TABLES

Table 1 Data cleaning results . . . 11

1 INTRODUZIONE

Le alterazioni ambientali causate dall'uomo e i sempre più rapidi cambiamenti climatici che il nostro pianeta si ritrova ad affrontare sono il maggiore pericolo attualmente esistente per il genere umano. Durante gli ultimi anni sempre più scienziati hanno studiato e monitorato questi fenomeni, confermandoli e riconoscendone la loro conseguente gravità. Parallelamente al riconoscimento dei cambiamenti climatici da parte di una schiacciante maggioranza della comunità scientifica, un altro fenomeno ha iniziato a prendere piede: la propagazione di numerose teorie complottiste nel campo ambientale. Gia durante la pandemia di COVID-19 si sono iniziate a pagare le conseguenze della diffusione della disinformazione tra i social, e la mancanza di fiducia verso le società farmaceutiche e verso i vaccini in generale ha portato diversi individui a scegliere di non farsi somministrare il vaccino, e in alcuni casi, ad andare conseguentemente incontro alla morte. Similarmente al caso del COVID-19, la disinformazione in campo climatico rischia di polarizzare eccessivamente gli individui, che potrebbero andare a rinunciare all'idea di adottare comportamenti e abitudini green. Considerata la forza crescente con la quale le teorie del complotto riescono a modificare il comportamento delle grandi masse, questo progetto ambisce ad analizzare attraverso il natural language processing (NLP), il pensiero dei negazionisti ambientali di twitter per estrapolarne le discussioni tipiche e il loro sentiment. Inoltre questo progetto si prefigge di analizzare i libri dei negazionisti ambientali più importanti e di trovare eventuali similarità con il corpus dei tweet estratti dal web.

2 DATA COLLECTION

Al fine di raggiungere gli obiettivi preposti per questo progetto la fase di data collection è stata suddivisa in due fasi. Nella prima, una volta selezionati i libri della letteratura pseudoscientifica correlata al Climate Change, sono stati processati i pdf di ciascun libro per poi riportare il loro testo in file.txt e in un file.csv. Nella seconda fase invece sono stati estratti i tweet correlati alle teorie del complotto sui cambiamenti climatici.

2.1 Selezione dei libri

I libri scelti per questo progetto non sono altro che i quattro libri pseudoscientifici più importanti nel campo dei cambiamenti climatici. Oltre ad essere i più venduti infatti, sono anche quelli che hanno suscitato più dibattito nella comunità scientifica, la quale ovviamente ha preso sin da subito le distanze dalle loro congetture e dalle loro conclusioni. I quattro libri selezionati sono riportati di seguito:

- Inhofe, James. The Greatest Hoax — How the global warming conspiracy threatens your future.
- Michaels, Pat and Balling, Robert. Climate of Extremes — Global warming science they don't want you to know.
- Bell, Larry. Climate of Corruption — Politics and power behind the global warming hoax.
- Andrew Montford. The Hockey Stick Illusion — Climategate and the corruption of science.

Un'ulteriore caratteristica rilevante di questi libri è che la loro stesura è stata verosimilmente finanziata dalle lobby dei combustibili fossili e dai gruppi di persone e società che non si vogliono adeguare alle contromisure ambientali imposte dalle politiche green. In uno studio pubblicato da Justin Farrell, professore e ricercatore della Yale University, sono state trovate evidenze circa i legami tra la filantropia americana e la diffusione di disinformazione sui cambiamenti climatici. L'organizzazione ambientalista Greenpeace inoltre, è riuscita a trovare le prove che l'industria energetica finanzia la negazione del cambiamento climatico nel suo progetto "Exxon Secrets".

Per via del fatto che il processing dei libri è intrinsecamente legato al processo di cleaning, i codici utilizzati per estrarre i corpus dai pdf sono trattati nel capitolo successivo.

2.2 Tweets collection

Per la fase di raccolta dei tweet è stata scelta la libreria snscrape, la quale è capace di restituire rapidamente i dati inerenti a ciascun tweet quali contenuto, nome dell'utente, numero di like, etc.

Prestando attenzione a non andare a raccogliere anche le risposte correlate ad ogni singolo tweet, sono stati collezionati 62518 tweet relativi al Climate Change in lingua inglese, raccolti tra il 2020-01-01 e il 2022-08-01. La query completa della ricerca effettuata è riportata nella Figura [1].

Per raccogliere i tweet dei complottisti, nella prima parte della query, che è composta dalle keyword utilizzate, sono state ricercate le parole climate e hoax, in modo tale da ottenere tutti i tweet che contengono almeno una volta ambe due le parole. Nella seconda parte della query è stato specificato di trovare solo i tweet

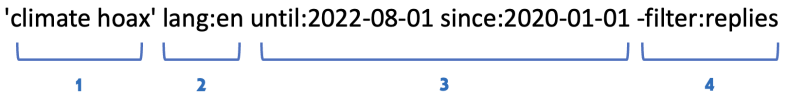


Figure 1: Query utilizzata per la ricerca dei tweet

in lingua inglese. Nella terza parte invece è stato definito il range temporale della ricerca, in questo studio si è optato per un range di 2 anni e 7 mesi (dal 01/01/2020 al 01/08/2022). Infine nell'ultima parte della query è stato inserito il filtro per le risposte ai singoli tweet, in modo tale da non raccogliere anche queste ultime.

Una volta definita e testata la query di ricerca è stato programmato il crawler destinato ad estrarre i tweet. Come primo step è stata creata la classe Tweets, la quale prende in ingresso il limite massimo di tweet che si vogliono raccogliere, la data di inizio raccolta, la data di fine raccolta e il numero minimo di retweet che ciascun tweet deve avere.[1]

```
class Tweets(object):

    def __init__(self,limit,until,since,min_retweets):
        self.limit = limit
        self.until=until
        self.since=since
        self.min_retweets=min_retweets
        print('Raccolta dei tweet in corso..\n')
```

Una volta inizializzata la classe[2] viene creata la funzione get_tweet e viene dichiarata la query appena discussa in formato stringa.

```
    def get_tweets(self):
        tweets = []
        keywords ="climate hoax"
        query = f"({keywords}) min_retweets:{self.min_retweets} lang:en" \
                f" until:{self.until} since:{self.since} -filter:replies"
```

Sempre dentro la medesima funzione vengono creati un try e un except, che svolgono la seguente funzione: se il programma non riesce ad elaborare il codice dentro il try senza incorrere in errori, viene visualizzato a video l'errore che ha interrotto l'esecuzione del codice; se invece il programma funziona correttamente, viene mandata in esecuzione la query di ricerca e vengono raccolte una serie di informazioni inerenti a ciascun tweet dentro un data frame. Una volta terminata la ricerca e conseguentemente riempito il data frame (df), si salvano i dati raccolti dentro un csv che viene chiamato 'raw_tweets'.

```
        try:
            i=0
            for tweet in sntwitter.TwitterSearchScraper(query).get_items():
                if len(tweets) == self.limit:
                    break
                else:
                    tweets.append([tweet.id, tweet.user.id,tweet.date, tweet.content,
                                   tweet.retweetCount,tweet.likeCount,tweet.replyCount,
                                   tweet.coordinates])
                    i+=1
                    if i % 100 == 0:
                        print(f'Tweet raccolti: {i} ')
```

[1] Nonostante l'aggiunta del parametro min_retweets la query risultante coincide comunque con la query vista prima in quando è stato deciso di settare questo parametro a zero per evitare di restringere troppo la ricerca.

[2] Viene inizializzata una volta compilata la richiesta nel __main__

```
            df = pd.DataFrame(tweets, columns=['ID', 'UserID' ,'Date', 'Tweet','Retweet',
                                               'Replies','Likes','Coordinate'])
            print(df)
            df.to_csv('/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/raw_tweets.csv',
                      index=False)

        except Exception as e:
            from colorama import Fore
            from colorama import Style
            print(f"{Fore.RED}UNEXPECTED ERROR:")
            exc_type, exc_obj, exc_tb = sys.exc_info()
            fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
            print(str(e), fname, exc_tb.tb_lineno,f'{Style.RESET_ALL}\n')
```

Questo processo appena descritto avviene soltanto una volta digitato il comando request.get_tweets() nel __main__ dove vengono specificati tutti i parametri della classe Tweet.

```
if __name__=='__main__':

    until = '2022-08-01'
    since = '2020-01-01'
    limit = 500000
    retweets = 0

    request = Tweets(limit,until,since,retweets)
    request.get_tweets()
    print(request)
```

2.3 Reverse geocoding e visualizzazione geografica dei tweet

Prendendo esempio dal lavoro di Biraj Dahal, Sathish A. P. Kumar e Zhenlong Li (2019) è stata riprodotta la rappresentazione grafica della distribuzione geografica dei tweet estratti. Come primo step è stata creata la classe ReverseGeo, che prende come parametro di ingresso le coordinate 'grezze' dei tweet.[3]

```
class ReverseGeo(object):

    def __init__(self, coord):
        self.coord = coord
        print('Elaborazione in corso..\n')
```

Le coordinate vengono ripulite con la funzione clean_coord() e vengono incasellate a coppie dentro una lista.

```
    def clean_coord(self):
        set_coord=[]
        for i in self.coord:
            coord=i[1]
            disallowed_characters = "()=abCdefghijklmnopqrstuvwxyz"
            for character in disallowed_characters:
                coord = coord.replace(character, "") #rimozione dei caratteri non numerici
            C = re.split(',', coord) #incasellamento delle due coordinate in una lista
            set_coord.append(C)
        return set_coord
```

Una volta ottenute le coordinate pulite viene creata una geo mappa dei tweet attraverso la funzione get_geomap() che è stata implementata sopratutto grazie alla libreria GeoPandas.

[3] Vengono definite grezze in quanto devono ancora essere ripulite dai caratteri testuali.

```
    def get_geomap(self):
        ds=pd.DataFrame(request.clean_coord(),
                 columns=["long","lat"])
        ds = ds.astype(np.float16) # converte le coordinate in float
        BBox = (-180, 180, -90, 90) # Bounding Box per delimitare i confini della mappa
        fig, ax = plt.subplots(figsize=(8, 6))
        world = gpd.read_file(
            gpd.datasets.get_path('naturalearth_lowres')) # carica la mappa 2D della Terra
        world.plot(color='whitesmoke', edgecolor='lightgrey',ax=ax)

        # plot scatter plot dei tweet
        ds.plot(x='long',y='lat', kind="scatter",zorder=1,c='deepskyblue',alpha=0.1, ax=ax)
        ax.set_xlim(BBox[0], BBox[1])
        ax.set_ylim(BBox[2], BBox[3])
        plt.show()
```

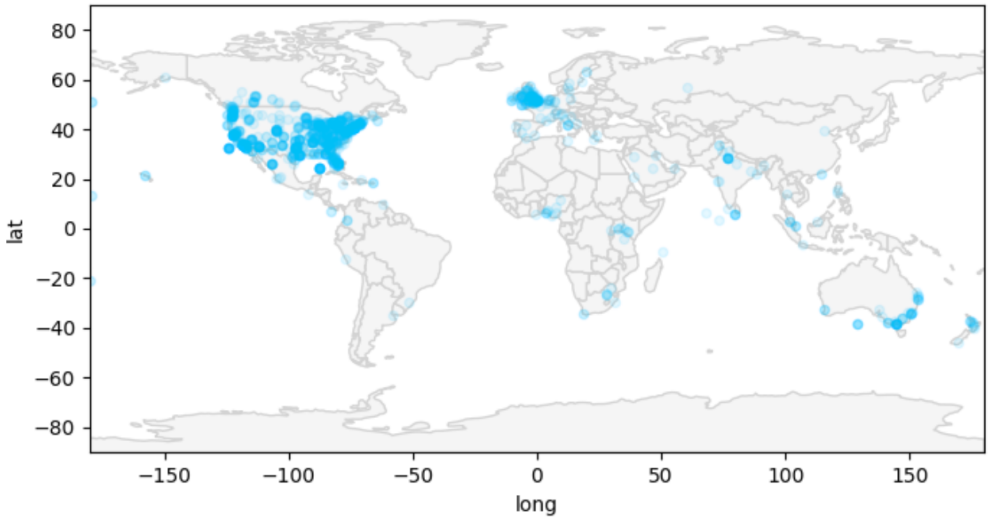


Figure 2: Distribuzione geografica dei tweet

Purtroppo i tweet rappresentati nella Figura [2] sono solo 1316 perché la restante parte dei tweet non aveva nessuna coppia di coordinate associata. Tuttavia, nonostante essi rappresentino solo un'esigua parte del dataset, i 1316 tweet geo localizzabili sono comunque sufficienti per farsi un'idea del dove provengano la maggior parte delle discussioni sul negazionismo ambientale.

Visualizzando la mappa si nota infatti che la maggioranza dei tweet provengono dagli Stati Uniti e dal Regno Unito. In primo luogo, questo fenomeno è spiegato dal fatto che sono stati raccolti esclusivamente i tweet di lingua inglese, la quale pur essendo una lingua diffusa in tutto il mondo, è comunque ovviamente più parlata in queste aree geografiche. Una conclusione un po' più azzardata invece, potrebbe consistere nel fatto che le aree con più tweet sono le aree con la maggior parte di negazionisti ambientali, ma per affermare ciò servirebbe un campione di tweet più ampio e maggiormente diversificato dal punto di vista linguistico.

L'ultima funzione della classe ReverseGeo si occupa di estrapolare il luogo esatto da dove è stato pubblicato l'i-esimo tweet.

```
    def get_location(self):
        for i in request.clean_coord():
            geolocator=Nominatim(user_agent='test/1')
            location=geolocator.reverse(f'{i[1]},{i[0]}') # 1 latitude , 2 longitude
            print(f'{location} : lat {i[1]} long {i[0]}\n')
```

Quest'ultima funzione è stata sviluppata al fine di alleggerire il lavoro dello scraper tweets.py andando a 'scrapare' solo le coordinate e non lo stato, la città e l'indirizzo per ogni tweet. Tuttavia, dopo aver effettuato qualche test è stato scoperto che l'oggetto sntwitter.TwitterSearchScraper() carica già a priori tutte le informazioni

per ogni tweet, di conseguenza la velocità di estrazione rimane invariata e si sarebbe potuto anche richiedere il luogo esatto nello scraper.

Richiesta nel __main__:

```
if __name__=='__main__':
    df = pd.read_csv(r'/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/raw_tweets.csv', encoding =
      "latin-1")
    NaN = df['Coordinate'].isna().sum() # conteggio dei NA
    valide=df['Coordinate'].notna().sum() # valido anche df['Coordinate'].count()
    print(f'Su un totale di {df.shape[0]} coordinate il numero di NaN è pari a {NaN} perciò solo {
      valide} sono valide per essere interpretate.')
    data=df[df['Coordinate'].notna()]
    del data['UserID'],data['Date'],data['Tweet'],data['Likes'],data['Retweet'],data['Replies']
    coordinate=data.values.tolist()

    request=ReverseGeo(coordinate)
    #request.get_location()
    request.get_geomap()
```

3 DATA CLEANING

3.1 Pulizia dei tweet

Anche per pulire i tweet è stata creata e inizializzata una classe. Tuttavia in questo caso, dal momento che i tweet sono stati salvati in un csv, la classe prende in ingresso un file csv.

```
class Cleaner(object):

    def __init__(self,data):
        self.data=data
        print('Cleaning del corpus in corso..\n')
```

Dopo aver inizializzato la classe viene definita la funzione clean_text. All'interno di questa funzione vengono svolti 5 step di cleaning. Nel primo step vengono inanzitutto eliminate le variabili indesiderate, dopodichè vengono utilizzate le colonne del dataset 'UserID' e 'Tweet' per riconoscere ed eliminare tutti i tweet scritti dal medisimo utente più di una volta (tweet doppi). In questo passaggio sono stati eliminati ben 1450 tweet e il numero totale di tweet del dateset è stato ridotto a 61068.[4] In conclusione, una volta espletato il suo scopo, la variabile UserID viene eliminata dal dataset.

```
    def clean_text(self):
        del self.data['Coordinate']
        ntweet=self.data.shape[0]
        df=self.data.drop_duplicates(
            subset=['UserID', 'Tweet'],
            keep='last').reset_index(drop=True)
        ntweet_finale=df.shape[0]
        print(f'Prima della pulizia dei tweet doppi il dataset era composto da  {ntweet} tweet'
              f' ora ha un numero di tweet pari a {ntweet_finale}.\n'
              f'{ntweet-ntweet_finale} tweet doppi sono stati eliminati.\n')
        df.dropna() # Pulizia dei NA risultanti dall'operazione di cleaning
        self.data=df
        del self.data['UserID']
```

Nello step successivo viene stilata una regular expression per eliminare i caratteri speciali e la punteggiatura dal testo dei tweet.

```
        self.data.Tweet.replace(regex=True, inplace=True,
                                to_replace=r'(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(https?\S+)',value=' ')
```

Completato questo breve ma fondamentale step, si procede alla tokenizzazione del testo dei tweet. Durante la tokenizzazione viene inoltre trasformato tutto il testo in carattere minuscolo.

```
        def tokenize(tweets):
            tokens = re.split('\W+', tweets)
            return tokens

        raw_tokens=self.data['Tweets_tokenized'] = self.data.Tweet.apply(lambda x: tokenize(x.lower()))
```

Tokenizzato il testo si può procedere alla eliminazione delle stopwords.

```
        stopwords = nltk.corpus.stopwords.words('english')
        keywords=['climate hoax']
        def remove_stopwords(raw_tokens):
            text = [word for word in raw_tokens if word not in stopwords and word not in keywords]
            return text

        self.data['Tokens'] = raw_tokens.apply(lambda x: remove_stopwords(x))
```

[4] Se fossero stati contati i tweet doppi a prescindere che siano stati scritti dallo stesso utente sarebbero stati eliminati 365 tweet in più.

In ragione di ridurre il caos nel corpus come ultimo passaggio di cleaning si procede alla lemmatizzazione di ogni token.

```
        wnl=WordNetLemmatizer()
        ds_intermedio=self.data.dropna() # rimozione NA

        lemmatized_tweet=[] # lista di stringhe
        for i in ds_intermedio.Tokens:
            l=[]
            for word in i:
                lem_word = wnl.lemmatize(word)
                text = "".join(lem_word)
                l.append(text)
            lemmatized_tweet.append(l)

        self.data['Token']=lemmatized_tweet

        del self.data['Tweet']
        del self.data['Tweets_tokenized']
        del self.data['Tokens']
        final_cleanedDS=self.data.dropna()

        final_cleanedDS.to_csv('/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/'
                               'cleaned_tweets.csv',index=False)
```

Richiesta nel __main__:

```
if __name__=='__main__':

    ds = pd.read_csv('/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/raw_tweets.csv')

    request = Cleaner(ds)
    request.clean_text()
```

3.2 Pulizia del corpus estratto dai libri

L'estrazione e la pulizia del corpus dei libri vengono eseguite dal programma pdf_cleaner.py il quale processa il testo del pdf dei libri convertendolo il formato .txt e pulisce lo stesso testo andando a performare gli stessi passaggi di cleaning applicati per pulire il testo dei tweet raccolti.

Una volta effettuati i dovuti import, viene definita la classe CleanerPDF, la quale prende in ingresso il path del libro, il nome del libro, la pagina da dove si vuole far partire l'analisi e la pagina alla quale in programma deve interrompere la raccolta del testo. In questo modo si può evitare di raccogliere il testo presente nelle pagine iniziali del libro, nell'indice e nei ringraziamenti finali, evitando così di 'sporcare' il testo di interesse.

```
class CleanerPDF(object):

    def __init__(self, libro,nome,inizio,fine):
        self.libro = libro
        self.nome=nome
        self.inizio=inizio
        self.fine=fine
        print('Elaborazione in corso..\n')
```

Una volta selezionato, il testo da raccogliere viene estratto attraverso la funzione get_pdf_file_content() che fa ricorso alla libreria PDFMiner. Sempre all'interno della stessa funzione viene anche creata una copia del testo in formato txt.

```
def get_pdf_file_content(self):

    resource_manager = PDFResourceManager(caching=True)
    out_text = StringIO() # string object
    laParams = LAParams() # object contenente il layout
    text_converter = TextConverter(resource_manager, out_text, laparams=laParams)

    f = open(self.libro, 'rb')
    interpreter = PDFPageInterpreter(resource_manager, text_converter)

    # grazie al seguente ciclo vengono selezionate solo le pagine incluse nel range:self.inizio-
  self.fine
    i=0
    for page in PDFPage.get_pages(f, pagenos=set(), maxpages=0, password="", caching=True,
  check_extractable=True):
        if i>=self.inizio:
            if i<=self.fine:
                interpreter.process_page(page)
                i+=1
            else:
                break
        else:
            i+=1
            pass

    text = out_text.getvalue() # contiene l'intero testo selezionato
    f.close()
    text_converter.close()
    out_text.close()
    copia = open(f'{self.nome}.txt', 'w')
    copia.write(text)
    copia.close()
    return text
```

Una volta estratto il testo dal PDF viene pulito il testo attraverso la funzione clean_text() effettuando li stessi passaggi della pulizia per i tweet fatta eccezione per la rimozione dei duplicati.

```
def clean_text(self):

    testo= request.get_pdf_file_content()
    '''
    Rimozione dei caratteri speciali e della punteggiatura
    '''
    testo = "".join([char for char in testo if char not in string.punctuation])

    '''
    Tokenizzazione
    '''
    raw_tokens = re.split('\W+', testo)
    nparole=len(raw_tokens)
    print(f'Il pdf in esame è composto da {nparole} parole. \n')

    '''
    Rimozione delle stopwords
    '''
    stopwords = nltk.corpus.stopwords.words('english')
    tokens=[t for t in raw_tokens if not t in stopwords]

    '''
    Lemmatizzazione
    '''
    wnl=WordNetLemmatizer()
    token=[]
    for w in tokens:
        token.append(wnl.lemmatize(w))
        #print(w,' : ',wnl.lemmatize(w)) # confronto

    '''
    Salvataggio in un dataframe e ritocchi finali
    '''
    df = pd.DataFrame(data=token, columns=['Token'])
    df['Token']=df['Token'].apply(str.lower)
    df['Token'] = df['Token'].replace(['u'], 'us')

    print(f'Dopo il processo di cleaning il numero di parole del libro è pari a {df.shape[0]}. \n')
    df.to_csv(f'/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/data_analysis/{self.nome}.csv'
  , index=False)
    print(df)
```

Nella tabella 1 viene riportato il conteggio delle parole per ciascun libro pre e post processo di cleaning.

Table 1: Data cleaning results

Libro	Parole pre cleaning	Parole post cleaning
The Greatest Hoax	97171	57146
Climate of Extremes	71345	44902
Climate of Corruption	95441	60283
The Hockey Stick Illusion	138441	75872

3.3 Unificazione del corpus dei libri

Ai fini di applicare una analisi di topic modelling sul corpus dei libri viene preparato un file txt con il corpus di tutti e quattro i libri.

```
def unify_corpus(self):
    file1 = open('The Greatest Hoax.txt', 'r')
    hoax = file1.read()
    file1.close()
    file2 = open('Climate of Corruption.txt', 'r')
    corruption = file2.read()
    file2.close()
    file3 = open('Climate of Extremes.txt', 'r')
    climExt = file3.read()
    file3.close()
    file4 = open('The Hockey Stick Illusion.txt', 'r')
    hockey = file4.read()
    file4.close()

    nuovotxt=hoax+corruption+climExt+hockey # unione dei corpus
    finalfile = open('corpuslibri.txt', 'w')
    finalfile.write(nuovotxt)
    finalfile.close()
```

Creato il txt dei corpus viene pulito il testo attraverso i classici passaggi di cleaning visti sopra e viene creato un csv contenente tutti i token estratti dai libri.

```
def topmod_preparation(self):
    print('Preparazione del corpus per il topic modelling in corso..')
    file = open('corpuslibri.txt', 'r')
    lines=file.readlines()

    list=[]
    for line in lines:
        if line.rstrip():
            #rimozione punteggiatura
            testo = "".join([char for char in line.strip() if char not in string.punctuation])
            testo_l=testo.lower()
            #tokenizzazione
            raw_tokens = re.split('\W+', testo_l)
            #rimozione stopwords
            stopwords = nltk.corpus.stopwords.words('english')
            tokens = [t for t in raw_tokens if not t in stopwords]
            #lemmatizzazione
            wnl = WordNetLemmatizer()
            token = []
            for w in tokens:
                if len(w)>1:
                    token.append(wnl.lemmatize(w))
            if token!=[]:
                list.append(str(token))
    file.close()

    df=pd.DataFrame(list,columns=['Token'])
    df.dropna()
    df.to_csv(f'/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/corpus_x_TM.csv', index=False)
```

Richiesta nel __main__:

```
if __name__=='__main__':
    request = CleanerPDF('./corpus_complottismo/The Greatest Hoax.pdf','The Greatest Hoax',10,292)
    #request = CleanerPDF('./corpus_complottismo/Climate of Corruption.pdf','Climate of Corruption', 9,
       286)
    #request = CleanerPDF('./corpus_complottismo/Climate of Extremes.pdf','Climate of Extremes', 7,
      252)
    #request = CleanerPDF('./corpus_complottismo/The Hockey Stick Illusion.pdf','The Hockey Stick
      Illusion', 9, 354)

    request.clean_text()
```

Preparazione al topic modelling:

```
    request.unify_corpus()
    request.topmod_preparation()
```

4 DATA ANALYSIS

Completata la fase di estrazione e di cleaning dei dati testuali è ora possibile eseguire l'analisi dei corpus estratti.

4.1 Data esploration

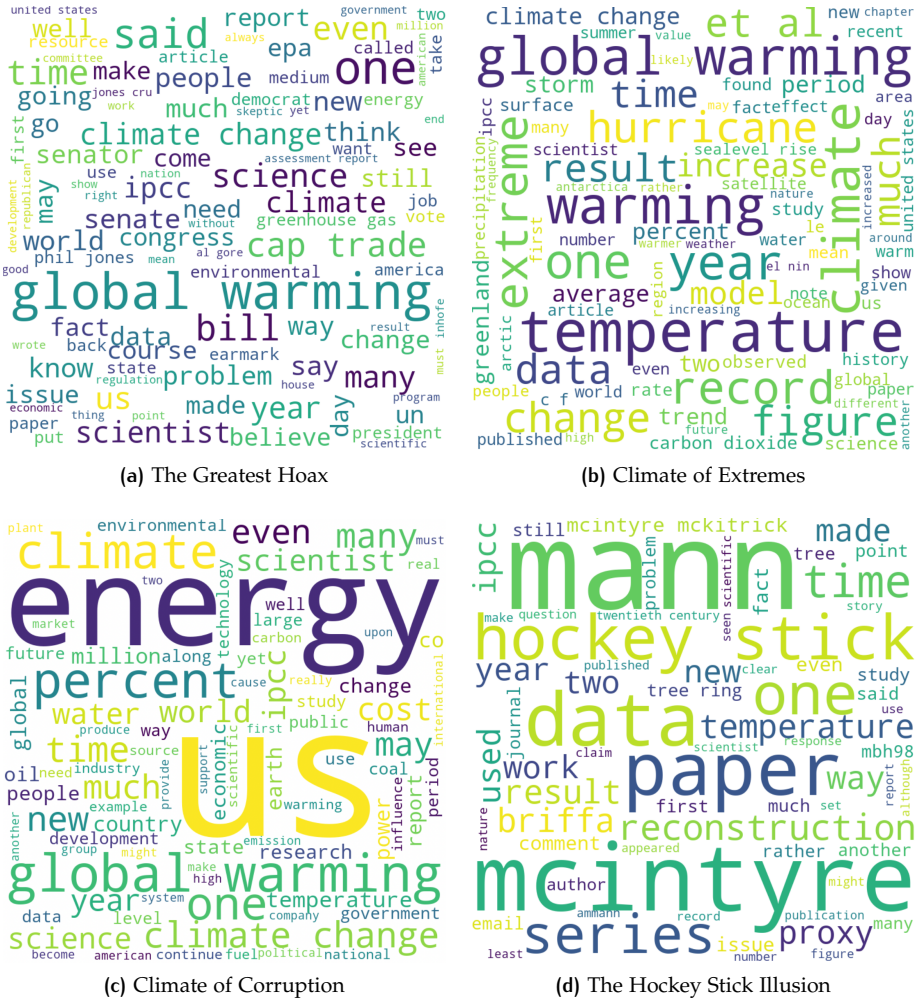
Un metodo creativo e visivamente efficace di rappresentare la distribuzione delle parole nel corpus è il ricorso alla rappresentazione tag cloud (o word cloud). Questa rappresentazione viene costruita tramite il programma word_cloud.py discusso qui di seguito. Come primo step viene costruita la classe Word_Cloud che prende in ingresso il csv dei token, i colori per la word cloud e un colore default impostato sul grigio. Questa classe è dotata di una chiamata speciale per poter costruire delle word cloud con dei colori differenti da quelli standard.

```
class Word_Cloud(object):

    def __init__(self,csv,color_to_words,default_color='grey'):
        self.csv=csv
        self.word_to_color = {word: color
                              for (color, words) in color_to_words.items()
                              for word in words}
        self.default_color = default_color
        print('Elaborazione in corso..\n')

    def __call__(self, word, **kwargs):
        return self.word_to_color.get(word, self.default_color)
```

Creazione della word cloud standard:

```
    def word_cloud(self):

        df = pd.read_csv(f'{self.csv}', encoding ="latin-1")

        #Conversione del dataframe in stringa e controllo per caratteri non ammessi
        ds=df.Token.to_string()
        disallowed_characters = "'"
        for character in disallowed_characters:
            ds = ds.replace(character, "")

        wordcloud = WordCloud(width=800, height=800,
                              background_color='white',
                              min_font_size=8).generate(ds)

        # plot the WordCloud image
        plt.figure(figsize=(8, 8), facecolor=None)
        plt.imshow(wordcloud, interpolation="bilinear")
        plt.axis("off")

        plt.show()
```

Nella Figura[3] sono riportate le word cloud per ciascun libro. Le parole più grandi sono quelle che appaiano con più frequenza nel testo mentre quelle più piccole o che non vengono proprio mostrate sono quelle meno frequenti.



(a) The Greatest Hoax

(b) Climate of Extremes

(c) Climate of Corruption

(d) The Hockey Stick Illusion

Figure 3: Word clouds dei libri.

Ai fini di enfatizzare meglio alcuni argomenti di interesse si possono riassegnare i colori delle parole attraverso un'apposita funzione alternativa.

```
def color_word_cloud(self):

        df = pd.read_csv(f'{self.csv}', encoding ="latin-1")

        wordcloud = WordCloud(width=800, height=800,
                              background_color='white',
                              min_font_size=20).generate(df.Token.to_string())

        wordcloud.recolor(color_func=request2)

        # Plot
        plt.figure()
        plt.imshow(wordcloud, interpolation="bilinear")
        plt.axis("off")
        plt.show()
```

Nella Figura[4] sono riportate le word cloud con i colori riassegnati manualmente per ciascun libro.

(a) The Greatest Hoax

(b) Climate of Extremes

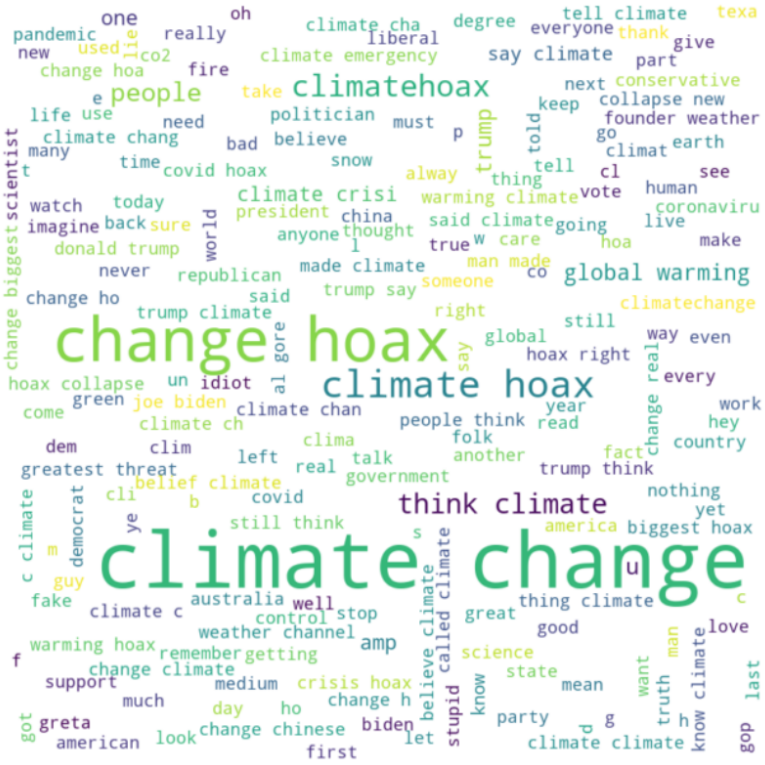
(c) Climate of Corruption

(d) The Hockey Stick Illusion

Figure 4: Word clouds dei libri con riassegnazione dei colori.

Grazie alla riassegnazione manuale dei colori è possibile notare ancora meglio che nei primi tre libri gli argomenti trattati sembrano molto simili. Specificamente, il topic 'global warming' è uno dei più affrontati. Il quarto libro invece sembra portare delle argomentazioni nettamente differenti rispetto ai primi tre. In particolare spiccano le parole 'mann','hockey stick' e 'mcintyre'. Queste parole si riferiscono al dibattito che si è generato intorno al cosidetto 'grafico a mazza da hockey', il quale rappresenta l'aumento anomalo delle temperature rilevato nella biosfera. In particolare i negazionisti ambientali hanno generato una considerevole confusione rifacendosi agli studi condotti da Michael E. Mann e Steve McIntyre i quali non riuscivano a replicare il grafico citato pocanzi. Di conseguenza si arriva a strumentalizzare questo fatto arrivando a sostenere che non esiste nessun cambiamento anomalo delle temperature.

Nella Figura[5] viene mostrata la word cloud per il corpus dei tweet.


climate change
change hoax
climate hoax
climatehoax
think climate
global warming
climate crisi
people
climatechange
climate chan
climate ch
climate emergency
climate c
climate climate
climate climate
climate climate

4.2 Topic modelling

Per eseguire il topic modelling sul corpus dei libri e dei tweet è stata scelta la Latent Dirichlet Allocation (LDA), un modello di analisi del linguaggio naturale ideato da David Blei, Andrew Ng e Michael Jordan, che permette di comprendere il significato semantico del testo analizzando la similarità tra la distribuzione dei termini del documento con quella di uno specifico topic.

Per la programmazione del modello è stata scelta la libreria pyLDAvis che oltre ad implementare la LDA, offre anche la possibilità di visualizzare iterattivamente i vari topic e di studiare quanto essi siano simili (o dissimili) tra di loro. Il primo step per la creazione del modello è la definizione della funzione topic_modelling() che prende in ingresso il csv dei token eil numero di topic (che devono essere scelti mandatoriamente a priori).

```
def topic_modelling(csv,num_top):
    ds=pd.read_csv(csv, encoding ="latin-1")
    ds.dropna(subset=['Token'], inplace=True)

    tokens=[] # lista di liste
    if type(ds.Token[1])==str: # se l'i-esima lista di token è una stringa
        lista = ds['Token'].to_list()
        for i in lista:
            import ast
            l = ast.literal_eval(i)
            l = [i.strip() for i in l]
            word=" ".join(l)
            tokens.append(word.split()) # prende ciascuna lista di token e li inserisce nella lista
     tokens
```

Una volta eseguito qualche controllo sui dati, che devono naturalmente essere compatibili per il regolare funzionamento del modello, si inizia a costruire il modello vero e proprio.

```
    try:
        dic = gensim.corpora.Dictionary(tokens) # prende in ingresso una lista di liste
        corpus_matrix= [dic.doc2bow(x) for x in tokens]

        LDA= gensim.models.ldamodel.LdaModel
        lda_model = gensim.models.ldamodel.LdaModel(corpus=corpus_matrix,
                                                    id2word=dic,
                                                    num_topics=num_top, # numero di topic
                                                    random_state=100, # seed
                                                    update_every=1,
                                                    chunksize=100,
                                                    passes=5,
                                                    alpha='auto',
                                                    per_word_topics=True)

        print(lda_model.print_topics()) # print dei topic
        coherence_model_lda = CoherenceModel(model=lda_model, texts=tokens, dictionary=dic, coherence='
     c_v')
        coherence_lda = coherence_model_lda.get_coherence()

        vis= gensimvis.prepare(lda_model,corpus_matrix,dic)#, mds='tsne') #mmds
        pyLDAvis.save_html(vis, 'LDA.html') # salvataggio in formato HTML
```

Calcolate la coherence e la perplexity del modello, due misure utilizzate per avere una misura della qualità del modello, vengono salvati i valori di questi due indici all'interno di un csv.

```
        #x salvare le performance
        perp=lda_model.log_perplexity(corpus_matrix)
        dataframe=pd.DataFrame({'NTopics':[num_top],'Coherence':[coherence_lda],'Perplexity':[perp]})
        if (os.path.exists("/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/data_analysis/
     coherence_tweet.csv")):
            dataframe.to_csv('/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/data_analysis/
     coherence_tweet.csv',
                             mode='a', index=False, header=False)
```

```
        else:
            dataframe.to_csv('/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/data_analysis/
    coherence_tweet.csv', index=False)
```

Eccezione in caso il codice non venga eseguito a buon fine:

```
    except Exception as e:
        from colorama import Fore
        from colorama import Style
        print(f"{Fore.RED}UNEXPECTED ERROR:")
        exc_type, exc_obj, exc_tb = sys.exc_info()
        fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print(str(e), fname, exc_tb.tb_lineno, f'{Style.RESET_ALL}\n')
```

Ai fini di agevolare la scelta del modello con le performance migliori è stata creata una funzione che restituisce il valore di coherence e di perplexity per ogni numero di topic scelto.

```
def performance_graph(csv):
    ds=pd.read_csv(csv)
    sns.set_theme(style="darkgrid")
    sns.lineplot(
        data=ds, x=ds.NTopics, y=ds.Coherence,
        markers=True, dashes=False
    )
    sns.scatterplot(x=ds.NTopics, y=ds.Coherence, data=ds)
    plt.show()

    sns.set_theme(style="darkgrid")
    sns.lineplot(
        data=ds, color='purple', x=ds.NTopics, y=ds.Perplexity,
        markers=True, dashes=False
    )
    sns.scatterplot(x=ds.NTopics, y=ds.Perplexity, data=ds, color='purple')
    plt.show()
```

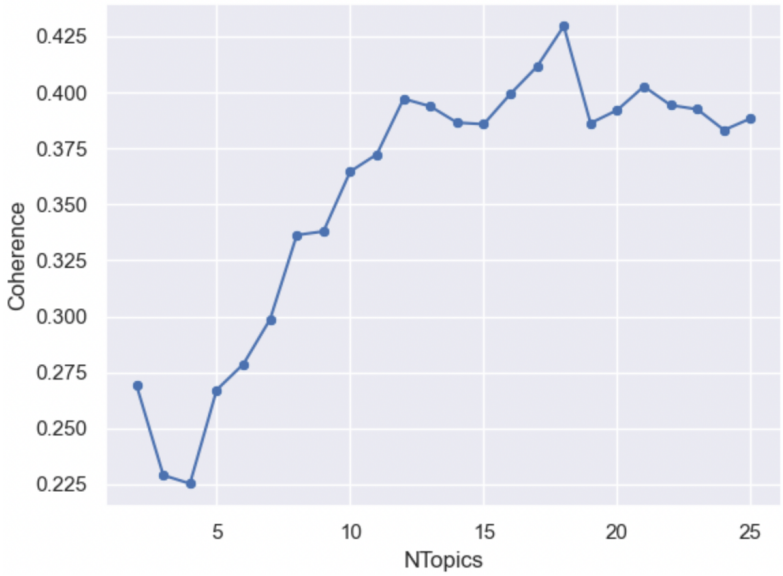


Figure 6: Topic Coherence corpus libri

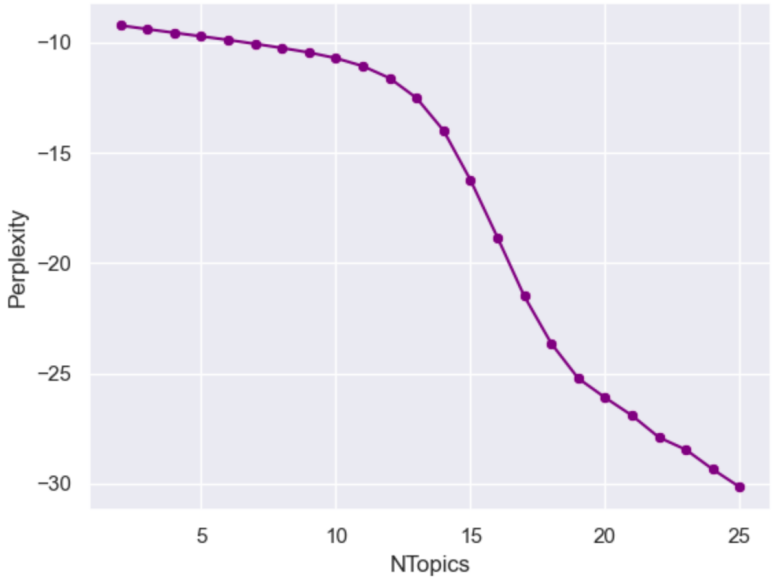


Figure 7: Topic Perplexity corpus libri

Con l'aiuto della serie di valori di coherence e di perplexity per ogni numero di topic, per l'analisi del corpus dei libri è stato scelto un numero di topic ottimale pari a 10. Grazie al modello ottenuto è possibile constatare che l'impressione avuta prima in sede di analisi esplorativa attraverso le word cloud si è rivelata esatta: esiste un topic (numero 1), appartenente al libro 'The Hockey Stick Illusion', che risulta molto diverso rispetto alle altre argomentazioni portate negli altri tre libri.

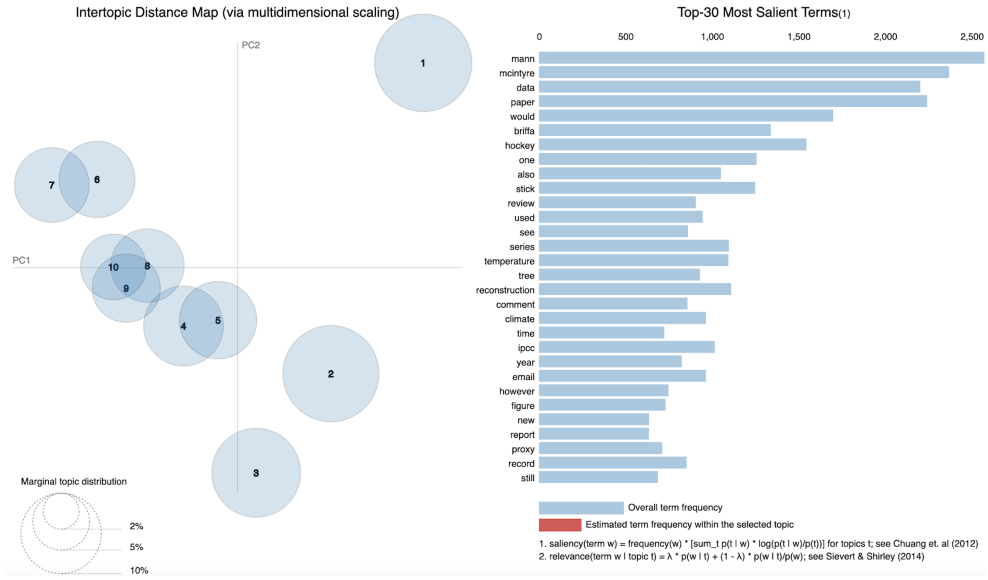


Figure 8: LDA corpus libri

Per la scelta del modello ottimale per il corpus dei tweet vengono replicati i medesimi due grafici visti sopra.

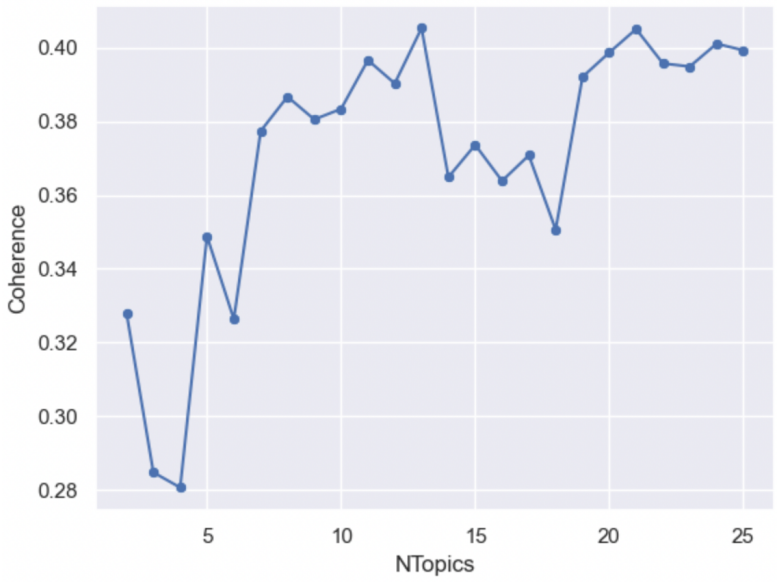


Figure 9: Topic Coherence corpus tweet

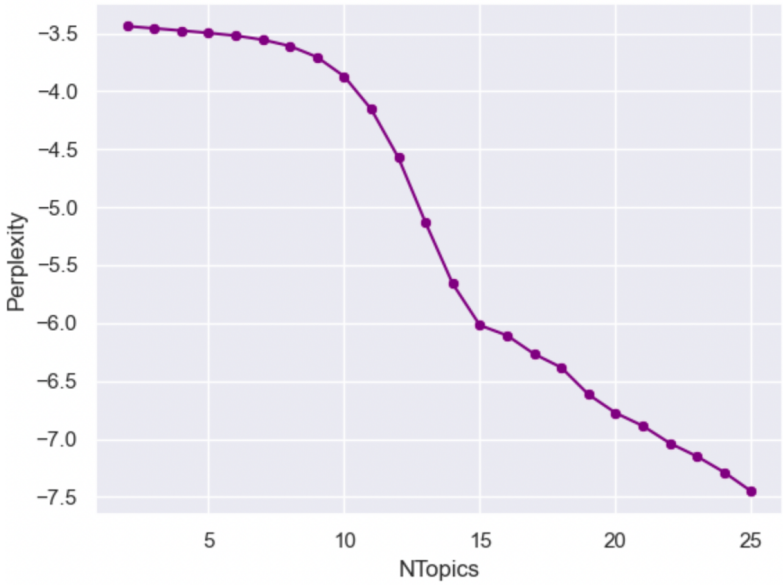


Figure 10: Topic Perplexity corpus tweet

Nonostante si dovrebbero ottenere dei risultati soddisfacenti tra i 6 e i 10 topic per non incorrere in una LDA non interpretabile è stato scelto di considerare un risultato incompleto ma ottimo dal punto di vista interpretativo. Di conseguenza il numero di topic è stato impostato a tre topic. Il primo topic (il macro topic) consiste nella negazione del cambiamento climatico (Climate change is a hoax). Il secondo topic, che è molto simile al primo ma meno frequente, consiste sempre nella negazione del cambiamento climatico, ma include anche quello che dice Trump. Il terzo topic, molto dissimile dai primi due, consiste in discorsi diffamatori rivolti all'attivista Greta Thunberg.

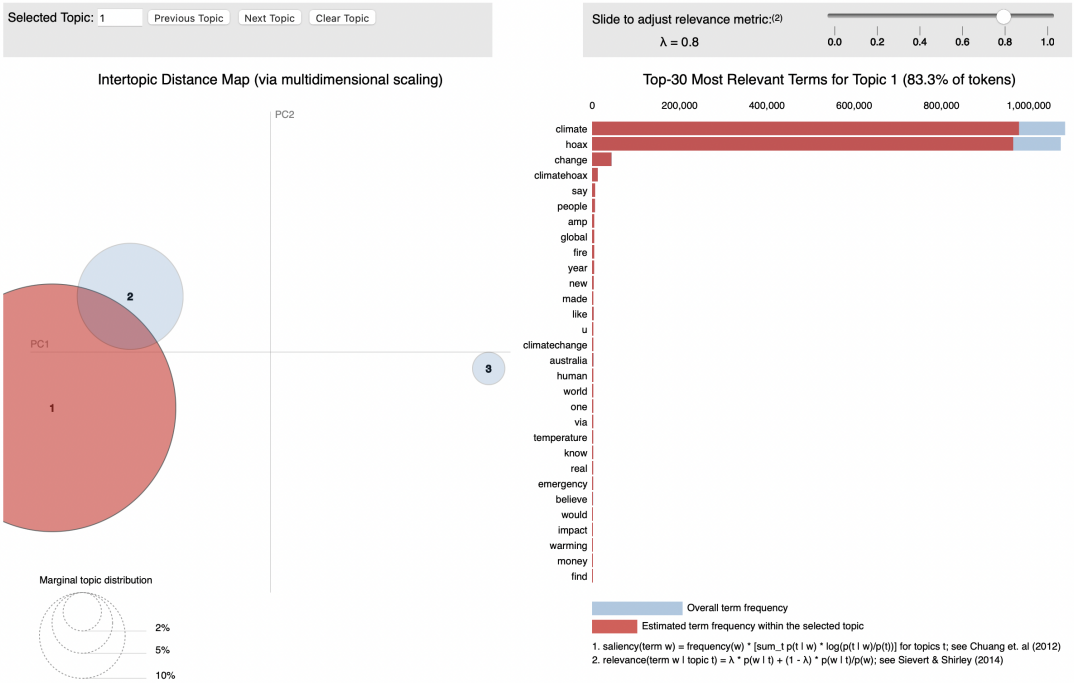


Figure 11: Topic 1

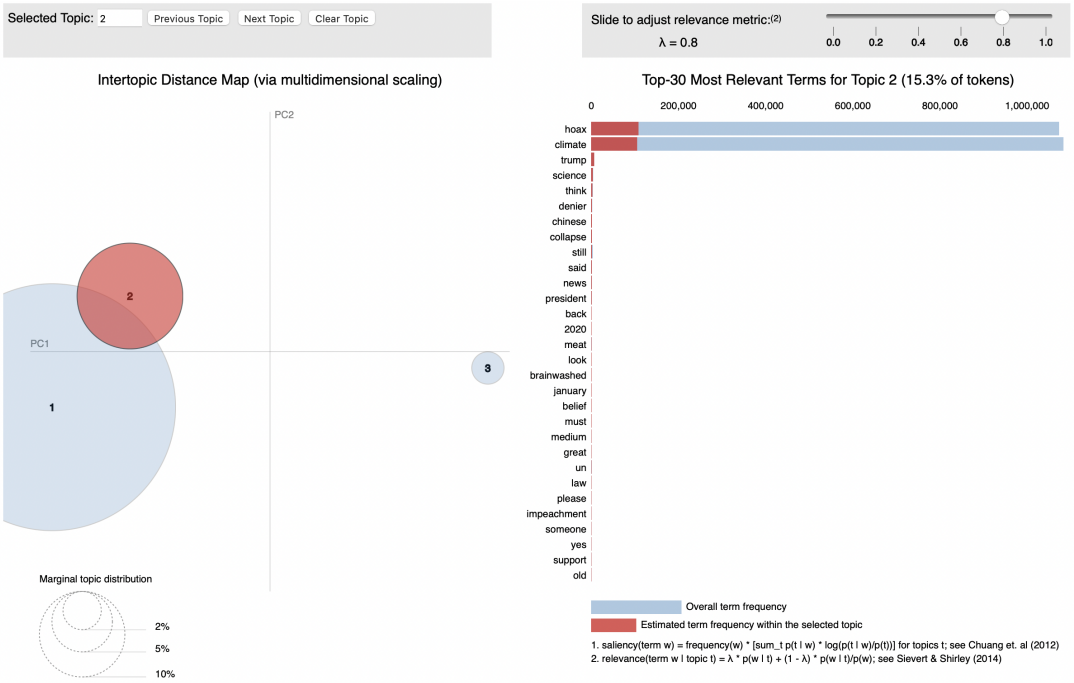


Figure 12: Topic 2

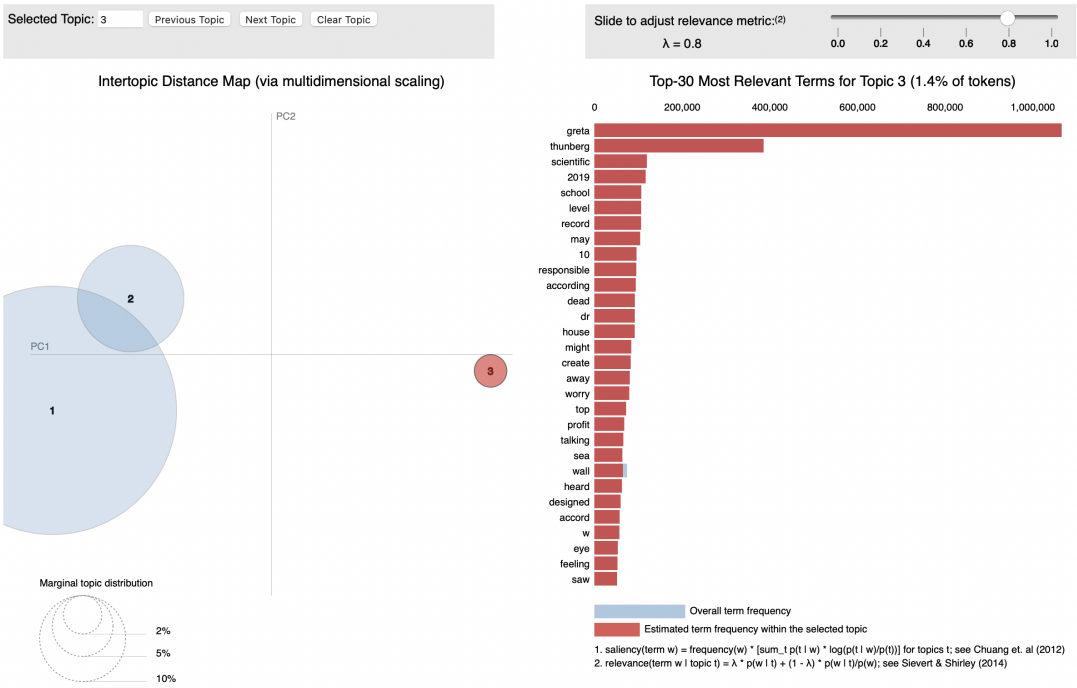


Figure 13: Topic 3

4.3 Sentiment analysis

Ai fini di misurare il grado di polarizzazione dei tweet raccolti è stata eseguita una sentiment analysis usando la libreria python vaderSentiment. Per calcolare il sentiment di ciascun tweet è stata creata la funzione get_tweet_sentiment all'interno della classe SentAnalyser (che prende come unico parametro il dataset). All'interno della funzione get_tweet_sentimet() si calcola il compound per ciascun tweet e si classifica il sentiment sulla base di tale valore. Una volta determinato quali tweet hanno un sentiment negativo, neutrale o positivo vengono salvate queste informazioni in un csv.

```
class SentAnalyser(object):

    def __init__(self, ds):
        self.ds = ds
        print('Elaborazione in corso..\n')

    def get_tweet_sentiment(self):
        '''
        Calcolo del sentiment
        '''
        text = ds['Token']
        list=[]
        numeric_sent=[]
        for i in text:
            vs = analyzer.polarity_scores(i)
            numeric_sent.append(vs['compound'])
            if vs['compound'] > 0:
                list.append('positive')
            elif vs['compound']== 0:
                list.append('neutral')
            else:
                list.append('negative')

        ds['Sentiment'] = list
        ds['Sentiment_score']=numeric_sent
        print(ds.Sentiment)
        ds.to_csv('/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/dsFin_sent.csv', index=False)
```

Per visualizzare i risultati dell'analisi viene creato un grafico a barre attraverso la funzione get_graph.

```
    def get_graph(self):
        ds = pd.read_csv('/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/dsFin_sent.csv')

        word_freq = defaultdict(int)
        sent=ds.Sentiment.tolist()
        for i in sent:
            word_freq[i] += 1
        sentiment=word_freq.keys()
        values=word_freq.values()

        print(list(sentiment))
        neg,pos,neu=word_freq['negative'],word_freq['positive'],word_freq['neutral']
        print(f'Tweet con sentiment negativo: {neg}\n'
              f'Tweet con sentiment positivo: {pos}\n'
              f'Tweet con sentiment neutrale: {neu}')

        color = ['red','lime','lightgrey']

        plt.bar(list(sentiment),list(values),
                width=0.6,linewidth=1,edgecolor='black',color=color)


        #plt.xlabel("Sentiment")
        plt.ylabel("Number of tweets")
        #plt.title("Sentiment analysis")
        plt.show()
```

L'output della Sentiment Analysis rende evidente l'estrema polarizzazione negativa dei tweet. Polarizzazione che in larga parte può essere spiegata dalla natura psicologica degli utenti protagonisti delle accese discussioni su twitter.

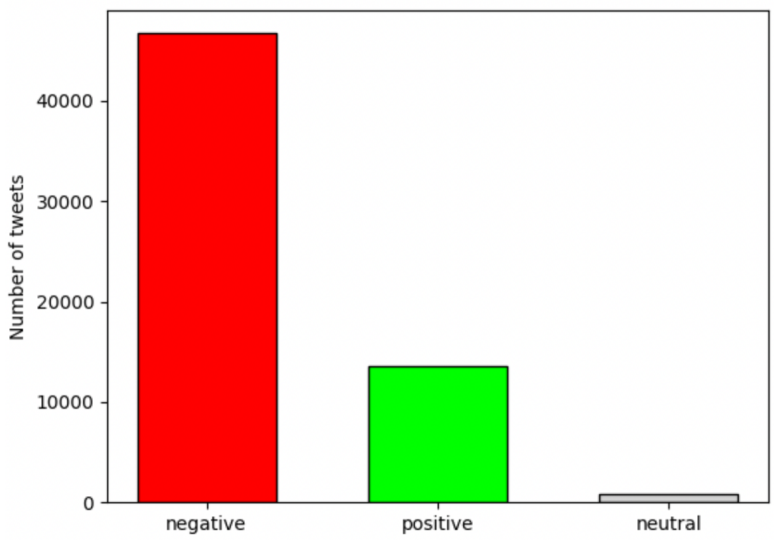


Figure 14: Sentiment analysis dei tweet

Per ottenere le emotion dei tweet viene utilizzata la funzione get_emotion, la quale utilizza la libreria NCRLex.

```
def get_emotion(self):

    list = ds['Token'].to_list()
    c=1
    if type(ds.Token[c]) == str:  # se l'i-esima lista di token è una stringa
        c+=1
        fear,anger,anticip,trust,surprise=[],[],[],[],[]
        positive, negative, sadness, disgust, joy=[],[],[],[],[]
        for i in list:
            #print(i)
            import ast
            l = ast.literal_eval(i)
            l = [i.strip() for i in l]
            word=" ".join(l)
            #print(word)
            emotion = NRCLex(word)
            #print(emotion.affect_frequencies)
            fear.append(emotion.affect_frequencies["fear"])
            anger.append(emotion.affect_frequencies["anger"])
            anticip.append(emotion.affect_frequencies["anticip"])
            trust.append(emotion.affect_frequencies["trust"])
            surprise.append(emotion.affect_frequencies["surprise"])
            positive.append(emotion.affect_frequencies["positive"])
            negative.append(emotion.affect_frequencies["negative"])
            sadness.append(emotion.affect_frequencies["sadness"])
            disgust.append(emotion.affect_frequencies["disgust"])
            joy.append(emotion.affect_frequencies["joy"])

            # Inserimento delle emotion nel dataframe:
        ds["Fear"]=pd.Series(fear)
        ds["Anger"]= pd.Series(anger)
        ds["Anticip"] = pd.Series(anticip)
        ds["Trust"] = pd.Series(trust)
        ds["Surprise"] = pd.Series(surprise)
        ds["Positive"] = pd.Series(positive)
        ds["Negative"] = pd.Series(negative)
        ds["Sadness"] = pd.Series(sadness)
        ds["Disgust"] = pd.Series(disgust)
        ds["Joy"] = pd.Series(joy)

    print(ds)
    ds.to_csv('/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/dsFin_emotion.csv', index=False
)
```

Una volta estratte le emotion viene costruito un grafico a barre per mostrare le emozioni medie percepite nei tweet.

```
def get_emotion_graph(self):
    ds = pd.read_csv('/Users/alessio/PycharmProjects/NLP_Climate/00307_ena/dsFin_emotion.csv')
    avg_emotions=[st.mean(ds.Fear),st.mean(ds.Anger),st.mean(ds.Anticip),st.mean(ds.Trust),st.mean(
  ds.Surprise),
                st.mean(ds.Positive),st.mean(ds.Negative),st.mean(ds.Sadness),st.mean(ds.Disgust),st.
  mean(ds.Joy)]
    avg_emotions.sort(reverse=True)

    index = ['Negative', 'Anger', 'Fear', 'Sadness','Surprise','Disgust','Positive','Trust','Joy','
  Anticip']

    sns.barplot(avg_emotions, index, palette='hls')
    plt.show()
```

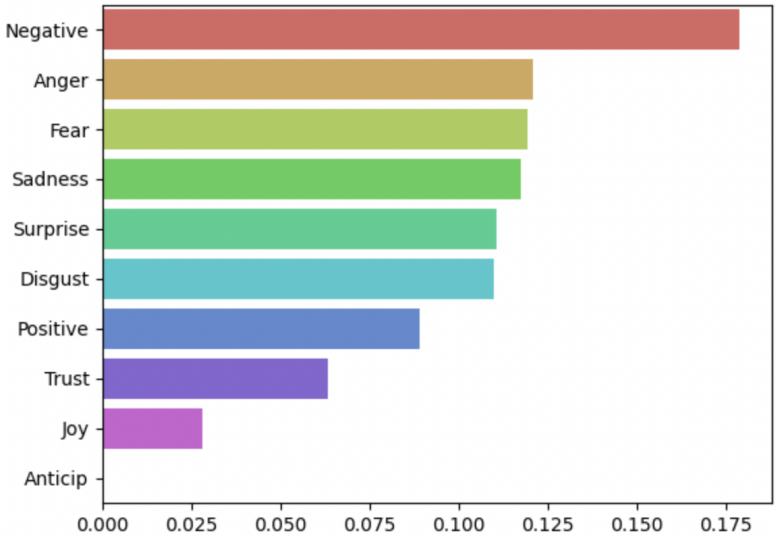


Figure 15: Emotion analysis dei tweet

Attraverso l'emotion analysis sono stati rilevati elevati livelli di rabbia, paura, tristezza e disgusto.

5 CONCLUSIONI

In questo studio si è cercato di comprendere il pensiero dei negazionisti ambientali analizzando la loro letteratura e un campione testuale delle loro discussioni sui social. Grazie alle word cloud e alla LDA si è avuto modo di identificare alcuni dei topic più acclarati dai negazionisti ambientali. Grazie alla sentiment and emotion analysis invece è stata scoperta una polarizzazione estremamente negativa nei tweet sul negazionismo ambientale. Dal punto di vista emotivo questa polarizzazione negativa è stata espressa in elevati livelli di rabbia, paura, tristezza e disgusto.

REFERENCES

[1] Biraj Dahal, Sathish A. P. Kumar, Zhenlong Li (2019). Topic modeling and sentiment analysis of global climate change tweets. *Social Network Analysis and Mining volume 9, Article number: 24.*
[Link]

[2] Fang Qiao, Jago Williams (2022). Topic Modelling and Sentiment Analysis of Global Warming Tweets: Evidence From Big Data Analysis. *Journal of Organizational and End User Computing, Volume 34, Issue 3.*
[Link]

[3] Justin Farrell (2018). The growth of climate change misinformation in US philanthropy: evidence from natural language processing. *Environmental Research Letters, Volume 14, Number 3.*
[Link]

[4] Joseph E. Uscinski, Karen Douglas, Stephan Lewandowsky (2017). Climate Change Conspiracy Theories. *Oxford Research Encyclopedia of Climate Science.*
[Link]

[5] Karen M. Douglas, Robbie M. Sutton (2015). Climate change: Why the conspiracy theories are dangerous. *Bulletin of the Atomic Scientists*, 71:2, 98-106.
[Link]

[6] Karen M. Douglas, Robbie M. Sutton (2008). The hidden impact of conspiracy theories: Perceived and actual influence of theories surrounding the death of Princess Diana. Journal of Social Psychology 148(2): 210-222.
[Link]

[7] Maria L. Loureiro, Maria Alló (2020). Sensing climate change and energy issues: Sentiment and emotion analysis with social media in the U.K. and Spain. *Energy Policy,Volume 143.*
[Link]