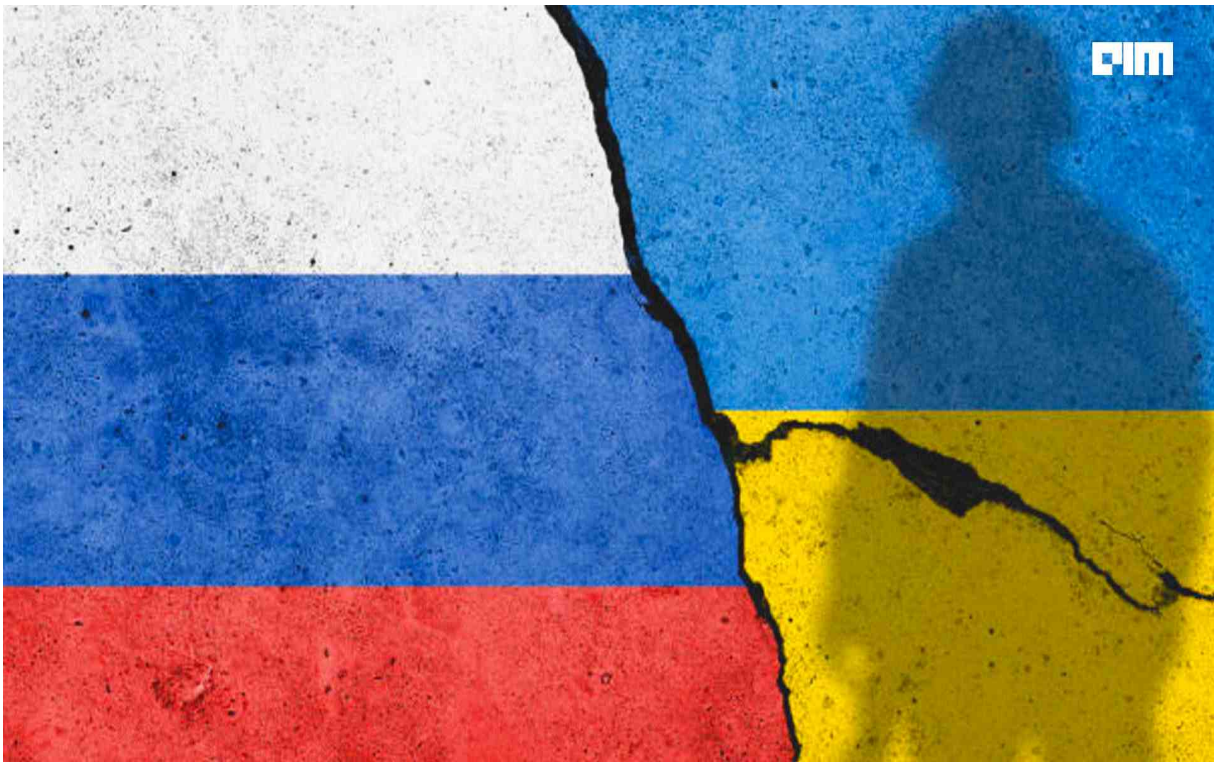
Twitter Sentiment Analysis

Analisi emotiva della comunità italiana tramite Twitter, sulla guerra russo-ucraina

Reti Geografiche: Struttura, Analisi e Prestazioni

Studente	Matricola
Francesco Pio Covino	0522501315



Docente: Prof.ssa Delfina Malandrino
Durata progetto: 03/2022 - 04/2022
Facoltà: Magistrale in Informatica: Cloud Computing

Abstract

Data l'odierna centralità che hanno i Social Network nelle nostre vite, la maggior parte delle persone e quindi degli utenti pubblica online il proprio pensiero riguardo la tematica in quel momento in cima alle tendenze. L'idea alla base del progetto è quella di analizzare nel tempo il pensiero comune della popolazione italiana riguardo uno degli eventi storici recenti più impattante: la guerra tra Russia e Ucraina.

L'analisi verrà svolta utilizzando come principale piattaforma Twitter, la cui funzionalità più conosciuta riguarda la raccolta di tweet in topic. Verranno scelti diversi hashtag che permetteranno la raccolta dei dati e la creazione di un dataset utile allo scopo finale. Il dataset sarà creato raccogliendo i tweet in fasce temporali diverse e per diversi giorni cosi da spalmare in modo equo i diversi avvenimenti. Con tecniche di Topic Modeling verranno divise in cluster le parole dei testi in base al topic. Un'analisi più specifica servirà per capire la divisione a livello di emozioni, parole più usate e cosi discorrendo...

Il progetto ha lo scopo di analizzare in modo imparziale il pensiero comune, etichettare i tweet in modo positivo o negativo e fare un'analisi emotiva degli stessi grazie all'uso di più tecniche e librerie. Sarà inoltre utile per capire nel profondo come la popolazione italiana si schiera o pensa in un momento di alta tensione internazionale come una guerra in Europa.

Contents

1

Contesto e Raccolta dati

Il conflitto tra Russia e Ucraina inizia e continua tuttora dal 24 febbraio 2022. Dopo aver avallato ufficialmente l'indipendenza delle due regioni separatiste di Donetsk e Luhansk, infatti, il presidente russo Putin ha ordinato l'invasione del Donbass e di altre aree dell'Ucraina, dando inizio a una guerra interna all'Europa.

Mentre la capitale Kiev viene sempre più accerchiata e colpita da razzi russi, continuano i combattimenti e i bombardamenti in diverse città dell'Ucraina, tra cui Mariupol, primo obiettivo strategico di Mosca, dove la resistenza ucraina sta opponendo un'opposizione disperata. Dall'inizio dell'invasione l'Unhcr ha stimato circa 3,5 milioni di profughi in fuga dalla guerra verso Stati vicini (la maggior parte in Polonia) e migliaia di morti tra civili e forze militari da ambo le parti.

In risposta alle azioni militari del governo russo, Stati Uniti ed Unione Europea stanno percorrendo la strada delle sanzioni economiche, evitando l'intervento militare della NATO e nel tentativo di scongiurare una Terza Guerra Mondiale. Il conflitto si sta estendendo anche al cyberspazio: il collettivo Anonymous ha dichiarato guerra alla Russia di Putin, mettendo in atto diversi attacchi hacker ai siti governativi e ai canali televisivi russi.

Una simile situazione rappresenta un evento storico di enorme portata, l'attenzione mondiale è concentrata costantemente verso gli ultimi accadimenti ed è perciò importante capire i vari schieramenti e quindi i vari pensieri. Il progetto si pone quindi come uno degli obiettivi quello di fare **Sentiment Analysis**. Più nello specifico la Sentiment Analysis si riferisce all'uso dell'elaborazione del linguaggio naturale, dell'analisi del testo, della linguistica computazionale e della biometria per identificare, estrarre, quantificare e studiare sistematicamente gli stati affettivi e le informazioni soggettive. Viene ampiamente applicata a campi quali recensioni e risposte a sondaggi, media online, social e materiali sanitari per applicazioni che vanno dal marketing al servizio clienti fino alla medicina clinica. (Fonte: Wikipedia).

Un caso specifico da citare è quello dell'**Hate Speech**. L' hate speech (o incitamento all'odio) è un evento purtroppo comune su Internet. Spesso i siti di social media come Facebook e Twitter affrontano il problema di identificare e censurare i post problem-

atici mentre soppesano il diritto alla libertà di parola. L'importanza di rilevare e moderare l'incitamento all'odio è evidente dalla forte connessione tra esso e i crimini d'odio effettivi.

L'identificazione precoce degli utenti che promuovono l'incitamento all'odio potrebbe consentire programmi di sensibilizzazione che tentano di prevenire un'escalation dal discorso all'azione. Siti come Twitter e Facebook hanno cercato di combattere attivamente l'incitamento all'odio. Nonostante queste ragioni, la ricerca della **NLP(Natural Language Processing)** sull'incitamento all'odio è stata molto limitata, principalmente a causa della mancanza di una definizione generale dello stesso o di un'analisi delle sue influenze demografiche o delle sue caratteristiche principali.

1.1. Obiettivi

Il fine ultimo del progetto è quello di analizzare, nel modo più attinente alla realtà, l'opinione pubblica italiana in un evento storico quale la guerra tra Russia e Ucraina.

- Collezionare i tweet tramite le API di Twitter e costruire un dataset che sia affidabile.
- Classificare nel modo più affidabile i tweet in 2 categorie: positivi o negativi.
- Classificare i tweet secondo l'emozione che traspare dal testo. Le emozioni base sono 4: gioia, rabbia, paura e tristezza.
- Topic Modeling: nell'apprendimento automatico e nell'elaborazione del linguaggio naturale, è uno dei tipi di modello statistico per scoprire quali sono gli "argomenti" che occorrono maggiormente in un testo. E' anche frequentemente usato per scoprire le strutture semantiche presenti nel testo.
- Rappresentazione grafica e schematica della distribuzione dei dati in base all'emozione. Verranno utilizzate librerie python che si occupano di plotting dei dati quali *pandas* e *matplotlib*.
- Analisi dei dati raccolti e dei risultati finali ottenuti.

1.2. Raccolta dei dati

Per la raccolta dei dati sono state utilizzate le API di Twitter, in quanto permettono di raccogliere in maniera gratuita i dati dai tweet con query di ricerca semplici e veloci. L'accesso ai dati è concesso ottenendo la dicitura di "sviluppatore", approvazione che si ottiene da Twitter stesso dimostrando di utilizzare i dati concessi senza intaccare la privacy degli utenti in alcun modo.

Per una raccolta dei dati coerente con l'obiettivo sono stati individuati e scelti una serie di hashtag che riguardano gli stessi o lo stesso topic: in questo caso la guerra russo-ucraina. Gli hashtag scelti sono di seguito elencati:

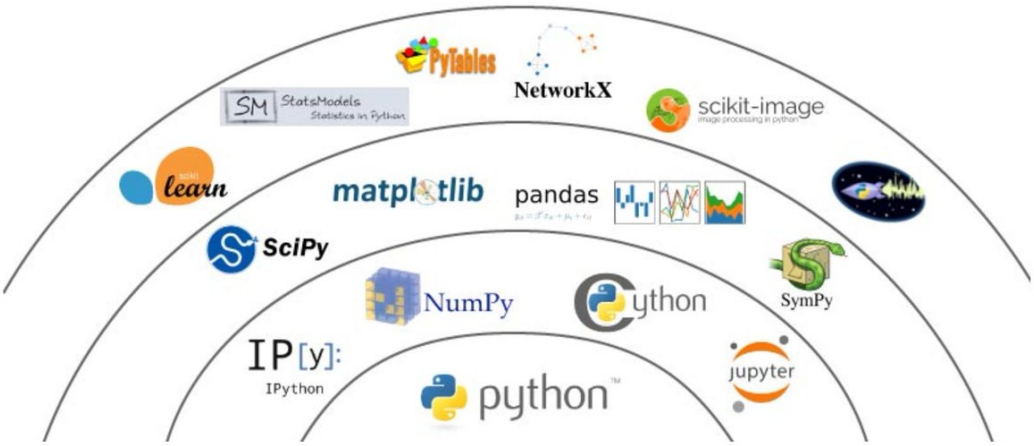
#putin, #ukraine, #ucraina, #russia, #UkraineWar, #ukrainerussiawar, #UkraineRussiaWar, #ukrainevsrussia, #russiavsukraine, #worldwar3, #PeaceInUkraine, #guerra, #guerraUcraina, #terzaguerramondiale

La raccolta dei dati è stata divisa in un lasso temporale che va da marzo 2022 ad

aprile 2022 in quanto non si può sapere con precisione, dato un tale evento, quanto tempo ancora possa durare e quali siano gli sviluppi futuri.

1.3. Tecnologie utilizzate

Per lo sviluppo del progetto è stato utlizzato Python. Python è un linguaggio di programmazione dinamico orientato agli oggetti utilizzabile per molti tipi di sviluppo software. Offre un forte supporto all'integrazione con altri linguaggi e programmi, è fornito di una estesa libreria standard e di numerose librerie adatte all'intelligenza artificiale e al machine learning.



Source

- **0. Jupyter Notebook**: Jupyter Notebook è un'applicazione Web open source che permette di creare e condividere documenti testuali interattivi, contenenti oggetti quali equazioni, grafici e codice sorgente eseguibile. Jupyter è diventato uno standard de-facto per data scientist perché offre la possibilità di realizzare, documentare e condividere analisi di dati all'interno di un framework che supporta più linguaggi e possibilità.
- **1. Tweepy**: Libreria open-source che facilita la comunicazione con le API di Twitter attraverso vari metodi, per la ricerca, l'invio di richieste HTTP, l'impaginazione dei risultati, l'autenticazione e cosi via...
- **2. Feel-it**: Libreria made in Italy adatta alla sentiment analysis in lingua italiana. Di semplice utilizzo, consente di etichettare un testo secondo 4 emozioni base: gioia, tristezza, rabbia, paura. Una seconda funzione permette di trovare se l'accezione del testo è positiva o negativa.
- **3. Pandas**: Pandas è una libreria Python usata per lavorare con gli insiemi di dati.Ha funzioni per analizzare, pulire, esplorare e manipolare i dati. Il nome "Pandas" ha un riferimento sia a "Panel Data", sia a "Python Data Analysis" ed è stato creato da Wes McKinney nel 2008.
- **4. Matplotlib**: Matplotlib è una libreria completa per creare visualizzazioni statiche, animate e interattive in Python.

- **5. Sklearn**: Sklearn è un modulo del linguaggio python con funzioni di scikit-learn utili per il machine learning e l'apprendimento automatico. La libreria contiene diversi algoritmi di apprendimento, classificatori o regressori (decision tree, regressione lineare, percetron, ecc.) e diversi dataset didattici di addestramento. Con poche righe di codice si può addestrare la macchina a svolgere un determinato compito.
- **6. WordCloud**: Libreria che consente la visualizzazione e la creazione di schemi di parole in base all'occorrenza.
- **7. hdbscan**: La libreria hdbscan è una suite di strumenti per utilizzare l'apprendimento non supervisionato per trovare cluster, o regioni dense, di un set di dati. L'algoritmo principale è HDBSCAN* proposto da Campello, Moulavi e Sander. La libreria fornisce un'implementazione ad alte prestazioni di questo algoritmo, insieme a strumenti per analizzare il clustering risultante.
- **8. umap**: (Uniform Manifold Approximation and Projection) è una tecnica di riduzione delle dimensioni che può essere usata per la visualizzazione in modo simile a t-SNE, ma anche per la riduzione generale delle dimensioni non lineari.

2

Sviluppo

2.1. Dataset

Il dataset ottenuto dalla fase di raccolta dati contiene all'incirca 6 mila tweet in lingua italiana. La raccolta è avvenuta in giorni diversi in modo da avere un insieme di opinioni più varie e da diversi utenti. Ogni riga del dataset ha i seguenti campi:

- **row**: indice del tweet, lo identifica.
- **text**: testo del tweet.
- **retweet**; numero di volte che il tweet è stato citato.
- **likes**: numero di mi piace ottenuti.
- **sentiment**: specifica il sentimento all'origine del testo (gioia, tristezza, rabbia, paura).
- **target**: indica se il tweet è positivo, negativo o neutrale.

Prima dell'utilizzo sono state svolte alcune operazioni sul testo dei tweet come la pulizia dei caratteri. Principalmente sono stati rimossi caratteri speciali, link ed emoji. Tutte parti del testo che non contribuivano all'analisi del sentimento. I dati raccolti sono stati uniti in un unico dataset finale generato come file formato csv.

Tutti i file.csv raccolti nei vari giorni sono stati poi in un unico dataset finale a cui è stata fatta la rimozione dei duplicati.

2.2. Twitter API

I tweet, come già accennato, sono stati raccolti tramite le API di Twitter. L'accesso è stato ottenuto creando un account twitter, nello specifico un account *twitter developer*. L'accesso ai dati, per motivi di sicurezza, viene fornito solo dopo una richiesta ufficiale in cui viene specificato l'uso che ne verrà fatto. Dato che l'account è gratuito ci sono delle limitazioni sull'uso delle API come per esempio il numero massimo giornaliero di richieste inviabili o il totale di tweet collezionabili per ogni richiesta.
Per accedere alle richieste è necessario autenticarsi, per questione di semplicità, la libreria **Tweepy** ha a disposizione metodi appositi che si occupano di comunicare con le API. Twitter fornisce all'utente i token di accesso da inviare con ogni richiesta. Un esempio di autentificazione con Tweepy nel codice sottostante:

```
api_key = "XXXX"
api_key_secret = "XXXX"
access_token = "XXXX"
access_token_secret = "XXXX"

auth = tweepy.AppAuthHandler(api_key, api_key_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth, wait_on_rate_limit=True)
```

Completata l'autenticazione, l'accesso alle API diventa possibile tramite la variabile *api*, che rappresenta l'interfaccia tra utente e API vere e proprie. Tutta la logica d raccolta tweet è stata racchiusa nella funzione *createdataset*. Un array di stringhe contenente tutti gli hashtag scelti viene esplorato elemento per elemento e Tweepy si occupa di fare una ricerca dove la query contiene l'hashtag corrente. Completata la raccolta di dati, la lista di tweet viene ritornata come valore e trasformata in un dataframe, una struttura dati ottimizzata alla gestione degli stessi. Dal dataframe sono stati poi rimossi tutti i duplicati ed è stato memorizzato il tutto in formato *csv* come mostrato nel codice sottostante:

```
final_list = create_dataset()
df = pd.DataFrame(final_list)
df.drop_duplicates(inplace=True, subset=['text'])
df.to_csv('dataset.csv', encoding='utf-8')
```

2.3. Etichettamento

La fase di etichettamento prevede innanzitutto di aggiungere due colonne al dataset, la prima nominata "sentiment", la seconda "target".La seconda parte si occupa di analizzare ogni tweet del dataset con due classificatori, Il primo classificatore aggiunge nella colonna "sentiment" il sentimento percepito (gioia, rabbia, tristezza, paura), il secondo classificatore invece assegna un valore alla colonna target (positivo, negativo, neutrale).

Per questa fase si è fatto uso di **feel-it**, libreria python made in italy che offre due classificatori, entrambi usano tecniche di deep learning. Il testo in input viene etichettato con 4 emozioni base quali rabbia, paura, tristezza e gioia nel caso dell'EmotionClassifier mentre è etichettato con positivo o negativo nel caso del SentimentClassifier.

Il dataset viene letto riga per riga e ogni testo dato in pasto al classificatore che ritorna l'emozione e il sentimento associati. La funzione nel codice sottostante:

```
def label_dataset():
    sentiment_classifier = SentimentClassifier()
    emotion_classifier = EmotionClassifier()

    # riga per riga vengono aggiornate le nuove colonne
```

```
    op = open("tweet_dataset.csv", "r")
    dt = csv.DictReader(op)
    up_dt = []
    count = 0
    for r in dt:
        # modifica della riga
        val_em = emotion_classifier.predict([r['text']])
        val_tar = sentiment_classifier.predict([r['text']])
        row = {'row': count,
                'text': r['text'],
               'retweet': r['retweet'],
               'likes': r['likes'],
               'sentiment': val_em[0],
               'target': val_tar[0]}
        up_dt.append(row)
        count += 1
    op.close()
    op = open("tweet_dataset.csv", "w", newline='')
    headers = ['row','text', 'retweet', 'likes', 'sentiment', 'target']
    data = csv.DictWriter(op, delimiter=',', fieldnames=headers)
    data.writerow(dict((heads, heads) for heads in headers))
    data.writerows(up_dt)
    op.close()
    print('label complete')
```

2.4. Topic Modeling

La fase di Topic Modeling prevede la modellazione del dataset etichettato per capire quali sono i topic, come sono distribuiti e le parole chiave per ogni cluster che si viene a formare. La più comune delle tecniche di Topic Modeling è la Latent Dirichlet Allocation, introdotta da Blei, Ng e Jordan nel 2002. Questa tecnica ha sicuramente il pregio di essere piuttosto versatile: non tiene conto della lingua e delle sue strutture, ma si basa sul modello bag-of-words. In parole semplici: viene creata una matrice avente per colonna tutte le singole parole dei testi e per righe i singoli documenti. I valori al suo interno corrispondono al numero delle occorrenze della parola in colonna nel documento in riga.

2.4.1. I problemi

Quali sono i problemi derivanti?

- 1. La rimozione delle parole non portatrici di significato è fatta in modo piuttosto grezzo: semplicemente prima di inserire i dati nella matrice di riferimento vengono eliminati quelli più frequenti di un certo valore ai quali si aggiunge una lista di cosiddette ”stopwords” - congiunzioni, articoli, preposizioni, ... - delineata a priori. Se questa pulizia non è fatta bene i topic individuati sono di difficile interpretazione.

- 2. il modello si basa su un numero di topic pre-inserito. Nel caso di un corpus poco variegato, in cui cioè si può rintracciare un numero limitato di topic, LDA funziona piuttosto bene. In presenza di molti topic diversi si perde efficienza.
- 3. Il modello forza ogni documento ad un topic. O meglio, attribuisce ad ogni documento una probabilità che appartenga ad un certo topic il cui totale dà 100. Ci sarà sempre un topic con una probabilità maggiore di altri, ma non è detto che in quel documento si parli di alcuno dei topic individuati.

2.4.2. BERT e c-TF-IDF

Il primo step è la conversione dei documenti in dati numerici. Per fare questo usaimo BERT in quanto è in grado di estrarre da un testo dei raggruppamenti (embeddings) basati sul contesto della parola. Un punto a favore è che ci sono già molti modelli pre-allenati.

Per generare gli embedding utilizzeremo la library sentence-trasformers. Gli embedding che genera infatti sono di buona qualità e funzionano abbastanza bene a livello di singolo documento. Utilizzo come modello **Distilbert**, un modello pre allenato multilingue che dovrebbe garantire un buon trade-off tra velocità e performance.

2.4.3. Clustering

A questo punto i testi con gli stessi argomenti saranno clusterizzati insieme in modo da riuscire a trovare i topic all'interno di questi cluster.

Prima di fare questo però è necessario diminuire la dimensionalità degli embeddings in quanto molti algoritmi di clustering sono in difficoltà nel gestire una ampia dimensionalità.

Nel progetto verrà usato HDBSCAN. Viene ridotta la dimensionalità degli embeddings dei tweet. Il processo verrà fatto con UMAP in quanto a differenza di altri algoritmi riesce a mantenere una buona porione della struttura dimensionale originale locale anche in dimensionalità minori.

Nel nostro caso si riduce la dimensionalità a 5 e si mantiene la dimensione del local neighborhood a 15. Il primo passo per leggere i topic è la creazione di un dataframe che raccolga i topic stessi in maniera separata. Verrà applicata la TF-IDF su tutti i testi con in comune un certo topic. Trovati i topic e divisi per cluster verranno prese le top 20 parole per ogni topic basate sul loro c-TF-IDF score.

```
1  STOP_WORDS =get_stop_words('italian')
2
3  filename = 'tweet_dataset.csv'
4  data_raw = pd.read_csv(filename)
5
6  #creo una lista con tutti i testi
7  data = list(data_raw['text'])
8
9  model = SentenceTransformer('distilbert-multilingual-nli-stsb-quora-ranking')
10 embeddings = model.encode(data, show_progress_bar=True)
11
```

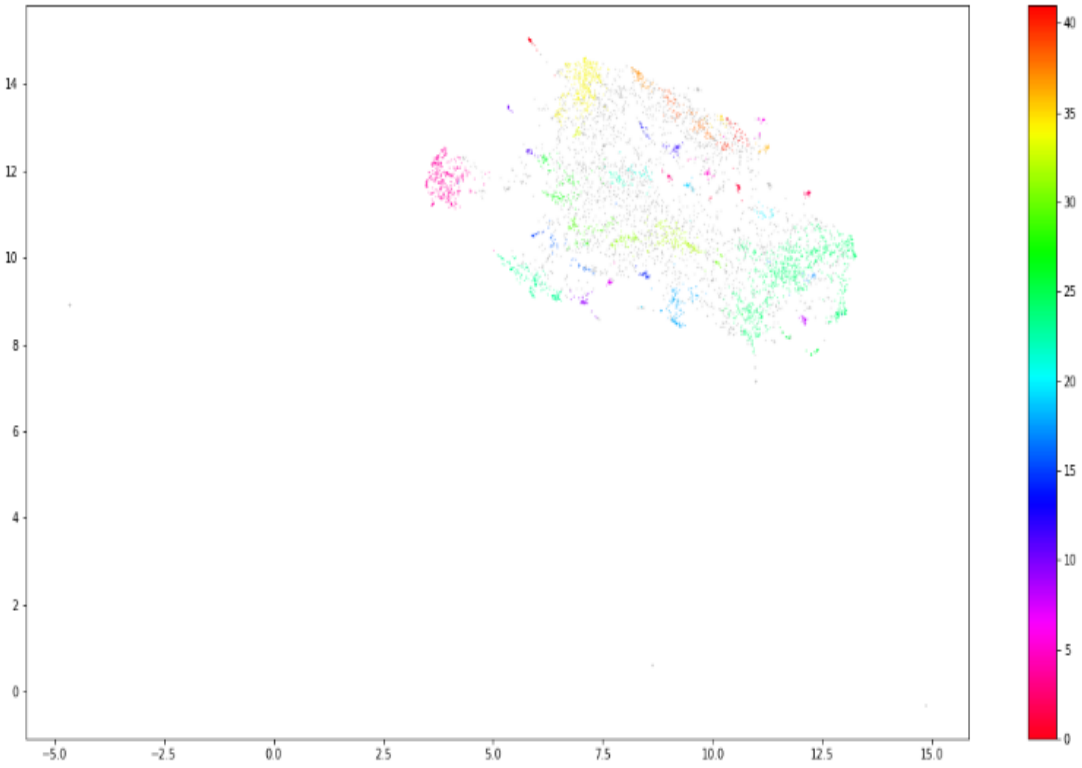
```
umap_embeddings = umap.UMAP(n_neighbors=15,
                            n_components=5,
                            metric='cosine').fit_transform(embeddings)

cluster = hdbscan.HDBSCAN(min_cluster_size=15,
                          metric='euclidean',
                          cluster_selection_method='eom').fit(umap_embeddings)

# DATA VISUALIZATION
# Prepare data
umap_data = umap.UMAP(n_neighbors=15, n_components=2, min_dist=0.0, metric='cosine').
result = pd.DataFrame(umap_data, columns=['x', 'y'])
result['labels'] = cluster.labels_

# Visualize clusters
fig, ax = plt.subplots(figsize=(20, 10))
outliers = result.loc[result.labels == -1, :]
clustered = result.loc[result.labels != -1, :]
plt.scatter(outliers.x, outliers.y, color='#BDBDBD', s=0.05)
plt.scatter(clustered.x, clustered.y, c=clustered.labels, s=0.05, cmap='hsv_r')
plt.colorbar()


# CREAZIONE TOPIC
# creo un nuovo dataframe solo per i topic

docs_df = pd.DataFrame(data, columns=["Doc"])
docs_df['Topic'] = cluster.labels_
docs_df['Doc_ID'] = range(len(docs_df))

# raccolta topic
# uniamo i documenti aventi lo stesso topic
docs_per_topic = docs_df.groupby(['Topic'], as_index=False).agg({'Doc': ' '.join})


# definiamo la funzione c_tf_idf:
def c_tf_idf(documents, m, ngram_range=(1, 1)):
    count = CountVectorizer(ngram_range=ngram_range, stop_words=STOP_WORDS).fit(docum
    t = count.transform(documents).toarray()
    w = t.sum(axis=1)
    tf = np.divide(t.T, w)
    sum_t = t.sum(axis=0)
    idf = np.log(np.divide(m, sum_t)).reshape(-1, 1)
    tf_idf = np.multiply(tf, idf)

    return tf_idf, count
```

```
58
59
60 # procediamo
61 tf_idf, count = c_tf_idf(docs_per_topic.Doc.values, m=len(data))
62
63 def extract_top_n_words_per_topic(tf_idf, count, docs_per_topic, n=20):
64     words = count.get_feature_names_out()
65     labels = list(docs_per_topic.Topic)
66     tf_idf_transposed = tf_idf.T
67     indices = tf_idf_transposed.argsort()[:, -n:]
68     top_n_words = {label: [(words[j], tf_idf_transposed[i][j]) for j in indices[i]][:
69     return top_n_words
70
71 def extract_topic_sizes(df):
72     topic_sizes = (df.groupby(['Topic'])
73                       .Doc
74                       .count()
75                       .reset_index()
76                       .rename({"Topic": "Topic", "Doc": "Size"}, axis='columns')
77                       .sort_values("Size", ascending=False))
78     return topic_sizes
79
80 top_n_words = extract_top_n_words_per_topic(tf_idf, count, docs_per_topic, n=20)
81 topic_sizes = extract_topic_sizes(docs_df); topic_sizes.head(15)
```

Completata la fase di Topic Modeling sarà possibile visualizzare tutte le parole più usate dei vari topic / cluster. Nell'esempio sottostante si può vedere il risultato del clustering, il grafico dei cluster individuati divisi per colore. Nel caso un elemento non viene identificato in un cluster, viene colorato di grigio. Sopra il grafico sono mostrati invece i 15 topic individuati dai tweet, ogni topic identifica un cluster e viceversa.

Nella seconda immagine viene invece mostrata la stampa delle prime 15 parole, le 15 con più occorrenze che appaiono in alcuni dei cluster generati. E' evidente come ogni cluster abbia un preciso argomento, nel cluster 3 per esempio si parla di negazionismo, finzione e attori per la guerra...

```
[('gas', 0.13737160443938937), ('rubli', 0.06741674435803695), ('russia', 0.0
4244810170346292), ('russo', 0.0364861639063066), ('putin', 0.030634568878101
058), ('petrolio', 0.030034392866089235), ('europa', 0.027964546262410257),
('rt', 0.026175302557194285), ('pagare', 0.024400122859816093), ('forniture',
0.02431453410442751), ('euro', 0.02411862268111105), ('pagamento', 0.02396873
9524747165), ('sanzioni', 0.02284886648210016), ('germania', 0.02104949560636
3735), ('gazprom', 0.019869415257238763)]
```



In [12]:
```
# mostra le 15 parole più importanti del topic 4
print(top_n_words[4][:15])
```

```
[('gas', 0.13737160443938937), ('rubli', 0.06741674435803695), ('russia', 0.0
4244810170346292), ('russo', 0.0364861639063066), ('putin', 0.030634568878101
058), ('petrolio', 0.030034392866089235), ('europa', 0.027964546262410257),
('rt', 0.026175302557194285), ('pagare', 0.024400122859816093), ('forniture',
0.02431453410442751), ('euro', 0.02411862268111105), ('pagamento', 0.02396873
9524747165), ('sanzioni', 0.02284886648210016), ('germania', 0.02104949560636
3735), ('gazprom', 0.019869415257238763)]
```

In [13]:
```
#mostra le 15 parole più importanti del topic 3
print(top_n_words[3][:15])
```

```
[('mariupol', 0.18124926265827584), ('bombardamento', 0.13882697003999842),
('sentito', 0.1270769067075368), ('attori', 0.11770806359964839), ('recitat
a', 0.11741149502305719), ('finto', 0.10286882496734456), ('freccero', 0.1015
4334168171492), ('dire', 0.08372637405336748), ('teatro', 0.078472042399765
6), ('piega', 0.07553818230053852), ('parrucchiere', 0.07553818230053852),
('propaganda', 0.07135520395595117), ('servizio', 0.0680699249346542), ('vist
o', 0.06135769969801625), ('negazionisti', 0.05824979968869694)]
```

In [14]:
```
#mostra le 15 parole più importanti del topic 2
print(top_n_words[2][:15])
```

```
[('vaccini', 0.14318389047477784), ('vaccino', 0.13961971290651753), ('covi
d', 0.09221174594819875), ('virus', 0.054475314068501285), ('pandemia', 0.049
626680040520045), ('coronavirus', 0.04380923482169961), ('alieni', 0.04380923
482169961), ('guerra', 0.043081106115870224), ('san', 0.04207116830793019),
('microchip', 0.03813970156729657), ('giorgio', 0.03813970156729657), ('covid
19', 0.033139716379512726), ('draghistan', 0.032561199278027274), ('nogreenpa
ss', 0.03229229745354813), ('invenzione', 0.030768642513534474)]
```

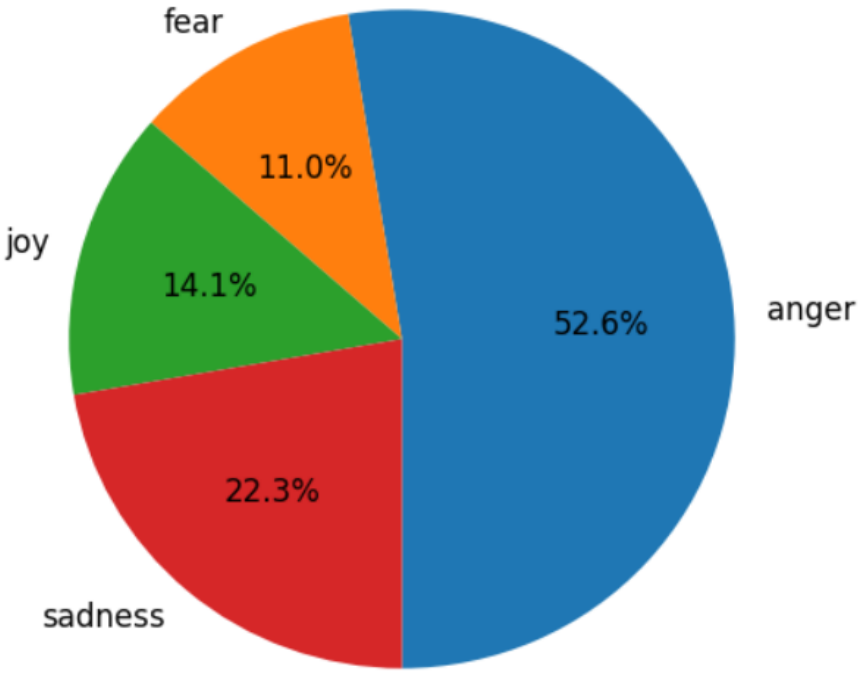
3

Risultati

Completata la fase di Topic Modeling, sono stati sintetizzati alcuni schemi per mostrare la distribuzione dei dati. Nel primo grafico a torta viene mostrata la distribuzione dei dati in base all'emozione. Dal grafico si evince quanto il sentimento più condiviso sia la rabbia.

```
sentiment
anger      3087
fear        645
joy         830
sadness    1308
dtype: int64
```

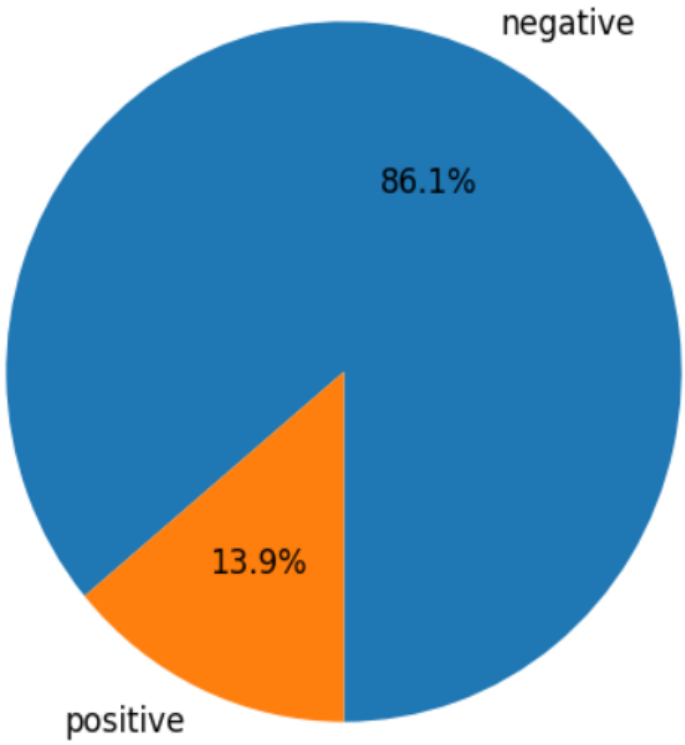
```
[5]: <AxesSubplot:>
```



Di conseguenza, molti dei testi hanno accezioni negative, prevedibile in quanto l'evento della guerra è solo negativo, per chiunque.

```
target
negative    5052
positive     818
dtype: int64
```

Out[6]: <AxesSubplot:>



In base alle etichette sono stati poi elaborati dei grafici Cloud of Words che mostrano le parole utilizzate, più i termini hanno un carattere grande più sono le occorrenze sono presenti.

Il grafico per le emozioni di gioia e rabbia.

Il grafico per le emozioni di tristezza e paura.


Sadness Tweets - Wordcloud
Name
inviati
L'orrore
interrotto
Gazprom
cittadino
decreto
sempre
difficoltà
Conte
text
p
Putin
j'accuse
così
ta
intanto
dtype
Tornare
Gas
Bucha
regge
meno
Leggo
spe
primavera
Length
object
Usa
storici
russo
andata
Draghi
Russia
dell'attore
Europa
mol
fianco
origini
claudio
GuerraUcraina


Fear Tweets - Wordcloud
minaccia
Mosca
indagine
subito
fa
vicina
Satan2
Name
andare
terzaguerramondiale
figlio
Putin
fargli
minacciando
viene
p
papa
Speriamo
text
Length
Secondo
sa
bisogna
Beh
Podcast
obiettivi
fake
comb
successo
object
Ucraina
NATO
Proekt
dignità
etype
GuerraUcraina
paura
implicato
sapere
Bi den
Ormai
nessuno
cosa
investigativa
atomica
forse
certo
brut

4

Conclusione

Dai risultati ottenuti si evince dunque un sentimento popolare abbastanza negativo rispetto la situazione, nonchè un sentimento condiviso di rabbia e paura rispetto il futuro della guerra e la possibilità di una guerra mondiale o dell'uso del nucleare. Una simile situazione ha messo in ginocchio parte della popolazione mondiale, istigandola a momenti di rabbia e frustrazione, spesso sfogata sui social. La seguente esperienza progettuale ha aiutato ad acquisire conoscenze sul campo della Social Media Analysis toccando con mano tutte le difficoltà riguardanti analisi dei dati, dei sentimenti. Sono state sperimentate nuove tecnologie, librerie e tecniche apposite.

Si spera che l'esito del progetto, seppur minimale e per certi versi banale, possa dare un contributo all'analisi del pensiero comune in un epoca importante come quella che si sta vivendo al momento della scrittura e possa essere una base per futuri progetti.