



Università degli Studi di Cagliari

Corso di Laurea Magistrale in Data Science, Business Analytics e Innovazione

Addestramento di modelli di intelligenza artificiale GPT-3 di Q & A
mediante la replicazione di utenti Twitter.

Professore:
Prof. Marco Ortu

Studente:
Mario Meloni

A.A. 2021-2022

Sommario

Introduzione	3
1. Scraping dei Tweet	5
2. Pulizia dei Tweet	8
3. Addestramento dei modelli GPT-3	12
3.1 Creazione del modello	12
3.2 Formulazione delle domande	19
4. Test del modello: analisi delle risposte	24
4.1 Frequency distributions	29
Conclusioni e sviluppi	32

Introduzione

Il progetto è incentrato sull'applicazione delle conoscenze acquisite nel corso di Web Analytics e Analisi Testuale, facendo particolare riferimento al Social Mining di Twitter, alla pulizia dei dati, all'analisi testuale mediante gli strumenti di Natural Language Processing (NLP) e alla Sentiment Analysis.

L'obiettivo del progetto consiste nella creazione di 3 modelli GPT-3, addestrati mediante l'estrazione di tweet da parte di 3 user, che imparino a rispondere a delle domande emulando il linguaggio, il lessico e lo stile dell'utente su cui sono stati addestrati.

In sintesi, verranno estratti 150 Tweet da ciascun utente. Una volta ripuliti da caratteri speciali, 75 di questi tweet saranno selezionati e utilizzati per l'addestramento di un modello GPT-3 che impari a rispondere a delle domande. Il modello verrà addestrato formulando manualmente 75 domande, una per Tweet, di modo che il Tweet figuri come risposta alla domanda formulata. In tal modo il modello dovrebbe imparare a rispondere alle domande che gli vengono poste emulando lo stile e le scelte lessicali dell'utente oggetto del suo addestramento.

Una volta addestrati 3 modelli su 3 utenti diversi, a questi verranno sottoposte le medesime 10 domande. Le risposte date dai 3 modelli verranno esaminate per confrontarne similitudini e differenze, e valutare in ultima istanza se i modelli replicano i personaggi su cui sono stati addestrati.

Sono stati scelti 3 personaggi reali molto diversi: Joe Biden (@JoeBiden), attuale presidente degli Stati Uniti; Conan O'Brien (@ConanOBrien), comico e conduttore di un talk show; Neil deGrasse Tyson (@neiltyson), scienziato americano. Tale scelta è stata fatta per vedere se i modelli, addestrati sulla base di personaggi provenienti da background così diversi, rispondessero in maniera diversa alle medesime 10 domande. Una simile circostanza indicherebbe un successo nel raggiungimento degli obiettivi di questo progetto. Come si vedrà nella parte finale del report, è possibile affermare con la dovuta cautela che tale obiettivo è stato raggiunto.

Il progetto è articolato in 4 parti:

- a. Estrazione dei Tweet di 3 utenti, mediante la libreria *"tweepy"*. L'estrazione dei dati avviene nello script *"A_Twitter_extraction"* del repository;
- b. Pulizia dei tweet da caratteri speciali. Tale processing avviene nello script *"B_data_clenaning.py"*;
- c. Addestramento dei modelli GPT-3, mediante le API di OpenAI¹. Tale fase avviene nei due script *"C_GPT-3.py"*, in cui vengono creati i 3 modelli, e *"D_question_generation.py"*, dove ai modelli vengono poste le 10 domande;

¹ Documentazione reperibile al link: [Introduction - OpenAI API](#)

- d. Analisi delle risposte fornite dai 3 diversi modelli. L'analisi viene condotta nello script *"E_question_analysis.py"*.

1. Scraping dei Tweet

La prima parte del progetto consiste nell' estrazione di Tweet di 3 utenti. L'estrazione di dati avviene nello script chiamato *"A_Twitter_extraction.py"*.

Sono stati selezionati come 3 utenti Joe Biden (@JoeBiden), Conan O' Brien (@ConanOBrien), e Neil deGrasse Tyson (@neiltyson). È stato dunque scelto un personaggio politico, ovvero l'attuale presidente degli stati uniti, un comico, conduttore di un famoso talk show, e uno scienziato. L'obiettivo era scegliere 3 personaggi diversi, così da poter analizzare come essi avrebbero risposto alle medesime domande.

Per l'estrazione dei dati è stata utilizzata la libreria *"Tweepy"*, che consente di fare richieste alle API di Twitter mediante Python.

In primis è stata definita la funzione *"create_df_from_user"* (in figura 1), avente l'obiettivo di creare un data frame contenente i tweets di un user. La funzione prende in input due parametri:

- *"username"*: nome utente dell'user;
- *"count"*: numero di tweet che si vuole estrarre.

```
def create_df_from_user(username, count):  
  
    """Creates a dataframe containing tweets of a Twitter User  
    parameters:  
    - username: username of the Twitter Account  
    - count: number of tweets to gather  
    """  
  
    columns = ["User", "Tweet"]  
    data = [[tweet.user.screen_name, str(tweet.full_text)]  
            for tweet in tweepy.Cursor(twitterAPI.user_timeline,  
                                       screen_name=username,  
                                       include_rts=False,  
                                       exclude_replies=True,  
                                       tweet_mode="extended").items(count)]  
  
    return pd.DataFrame(data, columns=columns)
```

Figura 1: funzione *"create_df_from_user"*

Per l'estrazione dei tweets la funzione fa uso del *"Cursor"*, classe della libreria *Tweepy* che gestisce automaticamente la paginazione, e del metodo *"user_timeline"*, che restituisce i tweet più recenti di un particolare user. Per questo metodo sono stati definiti i parametri:

- *"screen_name"*: in cui verrà riportato l'username dell'utente selezionato;
- *"include_rts"* = False, per escludere dalla raccolta di Tweet i Retweet, ovvero contenuti non direttamente generati dall'utente ma ricondivisi da altri utenti, irrilevanti ai fini dell'addestramento dei modelli GPT-3;
- *"exclude_replies"* = True, per escludere dalla raccolta di tweet i tweet che fossero risposte ad altri tweet;
- *"tweet_mode"* = *"extended"*, per raccogliere l'intero contenuto testuale di un tweet, in quanto il metodo di base tende ad estrarre solo un certo numero di caratteri e non l'intero Tweet.

La funzione restituisce un dataframe avente due colonne: *"User"* e *"Tweet"*.

Sono stati così generati dataframe relativi ai 3 utenti selezionati. Il parametro *"count"* è stato impostato a 150, per avere tweet un bacino di tweet sufficientemente grande per filtrare quelli con migliore contenuto e dunque più idonei per l'addestramento del modello. Come si vedrà più avanti, nella fase di addestramento verranno selezionati soltanto 75 tweet, per ragioni sia di tempo, sia di costo dell'utilizzo delle API di OpenAI, sia di qualità del contenuto dei tweet.

I 3 file csv così generati sono visibili nella cartella *data\raw_data*. Come separatore è stato utilizzato *";"*, in quanto la virgola, utilizzata spesso nei tweet, rischiava di compromettere la lettura dei csv nelle fasi successive. In figura 2 si è riportato un estratto del csv relativo all'utente "Conan O'Brien".

User;Tweet	
ConanOBrien;Nobody asked, so here it is: a brand new remote with Jordan Schlansky! https://t.co/zLp8Sp0Kwi	
ConanOBrien;Hear my \$900 improvised therapy session with Aubrey Plaza (@evilhag) @ https://t.co/pf0k236hQr https://t.co/B1MUU0I2AZ	
ConanOBrien;.@Beyonce, last chance to join me in a duet for my album of Irish dance tunes.	
ConanOBrien;Right now the FBI is Googling how to get ketchup off sensitive documents.	
ConanOBrien;A flight attendant just asked me to change my seat because "your head is making the plane imbalanced."	
ConanOBrien;More think pieces on how to properly cook salmon, please.	
ConanOBrien;Talked to a 911 operator on my podcast. I'm still not sure if he called us or if we called him. https://t.co/2VfLCmLc4C	
ConanOBrien;I wonder if the FBI found all of his burner toilets.	
ConanOBrien;In true form, @realjeffreycross began this interview by roasting me. https://t.co/pf0k236hQr https://t.co/WVBr7Yn2im	
ConanOBrien;The car warranty bots stopped calling and now I'm nervous I did something to hurt their feelings.	
ConanOBrien;I already knew Kim and Pete broke up because Alex Jones's lawyer sent me their text messages.	
ConanOBrien;I managed to get booked on my wife's podcast. I was her 8th choice. https://t.co/RLQbNp3HvE	
ConanOBrien;Now I'm glad I turned down the role of Batgirl.	
ConanOBrien;"Huge thanks to my fan Kristy for informing me about a Facebook page for people who think I'm ""bangable."" https://t.co/N1kmA7oaTd "	
ConanOBrien;Hear me chat with one of the only cast members from #TheOffice who didn't intern for me, @bjnovak. https://t.co/pf0k236hQr https://t.co/2sZxTvK0ne	
ConanOBrien;Bill Tull, our prop master extraordinaire for 28 years, died this morning. We always asked the impossible of Bill, sometimes minutes before he died.	
ConanOBrien;The White House now says it's only a recession if you see a salamander wearing a top hat.	
ConanOBrien;"My very talented wife, Liza, has a new history podcast that I absolutely love.	
https://t.co/2sZxTvK0ne	
ConanOBrien;Had a blast chatting with @nikkiglasier about living with her parents, getting heckled by a Pence supporter, and how she thinks I'd do on	

Figura 2: csv con i Tweet di "Conan O'Brien"

Come si può vedere, i tweet risultano "sporchi", inquinati da numerosi link, caratteri speciali e newlines che rendono difficile la lettura del csv e l'utilizzo dei dati in esso contenuti. Per questo si è proceduto, nella seconda fase del progetto, alla pulizia dei dati.

2. Pulizia dei Tweet

La seconda parte del progetto, nello script “*B_data_cleaning.py*”, consiste nella pulizia dei tweet raccolti. L’obiettivo di questa fase del progetto è rimuovere del contenuto testuale dei tweet caratteri speciali, emoticons, links etc.

Per farlo sono state definite diverse funzioni.

In primis è stata definita la funzione “*remove_emojis*” (figura 3), la quale prende in ingresso una stringa e ne rimuove gli emoji. Per fare questo è stata utilizzata la libreria “*re*”, e sono state definite mediante il metodo “*re.compile*” delle *regular expressions* corrispondenti alle stringhe Unicode corrispondenti agli emoji, che vengono rimosse dalla stringa mediante il metodo “*.sub*” del modulo *re*.

```
def remove_emojis(string):  
    """ Removes emojis from a string. """  
  
    emoji = re.compile("[  
        U"\U0001F600-\U0001F64F" # emoticons  
        U"\U0001F300-\U0001F5FF" # symbols & pictographs  
        U"\U0001F680-\U0001F6FF" # transport & map symbols  
        U"\U0001F1E0-\U0001F1FF" # flags (iOS)  
        U"\U00002500-\U00002BEF" # chinese char  
        U"\U00002702-\U000027B0"  
        U"\U000024C2-\U0001F251"  
        "]+", flags=re.UNICODE)  
    return re.sub(emoji, '', string)
```

Figura 3: funzione “*remove_emojis*”

Tale funzione è stata definita per essere utilizzata all’interno di un’altra funzione, chiama “*clean_text*” (figura 4), avente l’obiettivo di ripulire un tweet da tutti i caratteri speciali e i links.


```
def clean_text(tweet):
    """
    Removes links and special characters from tweet.
    For Hashtags, only the symbol # is removed, since the hashtag itself could contain precious info.
    Same with mentions.
    Links are removed completely.
    Newlines characters are also removed, for data-formatting reasons.
    """
    try:
        cleaned_tweet = re.sub("#", " ", tweet) # rimozione simbolo "#"
        cleaned_tweet = re.sub("@", " ", cleaned_tweet) # rimozione simbolo "@"
        cleaned_tweet = re.sub("https?\S+", " ", cleaned_tweet) # rimozione links
        cleaned_tweet = re.sub("\n", " ", cleaned_tweet) # rimozione new lines
        cleaned_tweet = remove_emojis(cleaned_tweet) # rimozione emoji
        cleaned_tweet = re.sub("\s+", " ", cleaned_tweet) # rimozione spazi multipli
    except Exception:
        cleaned_tweet = ''
    return cleaned_tweet
```

Figura 4: funzione "clean_text"

Ancora una volta nella funzione vengono utilizzate le *RegEx* per la pulizia delle stringhe. Si è deciso, per quanto riguarda gli hashtag (#) e le menzioni (@) spesso utilizzate su Twitter, di rimuovere solo il carattere che le identifica, mantenendo invece il contenuto testuale, in quanto ritenuto semanticamente importante per la comprensione del tweet.

I link sono stati rimossi completamente; si è deciso inoltre di rimuovere le newlines (“\n”) e gli spazi multipli in quanto inquinavano la visualizzazione del csv. Notare che all’interno di questa funzione è stata utilizzata anche la funzione “*remove_emojis*” citata prima.

La funzione restituisce la stringa che viene data in ingresso pulita dai caratteri speciali.

Successivamente è stata definita la funzione “*remove_punct*” (figura 5), la quale presa una lista di tokens (ovvero una lista in cui ciascun elemento è una parola) ne rimuove la punteggiatura.

```
def remove_punct(tokens):
    """remove punctuation from a list of tokens"""
    return [token for token in tokens if token.isalpha()]
```

Figura 5: funzione “*remove_punct*”

Tale funzione verrà utilizzata successivamente per includere nel dataframe “pulito” una colonna contenente il numero di tokens del tweet (esclusa appunto la punteggiatura), in modo da poter effettuare un filtraggio dei tweet e mantenere nel dataframe soltanto quelli che superano un certo numero di parole.

Tutte queste operazioni vengono svolte con la funzione *“clean_csv”* (figura 6), la quale prende in ingresso due parametri:

- *“filename”*: ovvero il nome del file csv da pulire, che nel nostro caso sarà il csv con i dati grezzi estratti da Twitter;
- *“min_token”*: ovvero il numero di parole minimo che il tweet deve contenere per essere mantenuto a seguito della pulizia del csv.

```
def clean_csv(filename, min_token = 10):  
  
    """ returns the input csv with Tweets cleaned from special characters.  
    This function also adds a new column to the csv file, that contains the number of tokens. In this way,  
    we can filter rows based on the number of tokens, and choose only tweets that have a certain number of tokens."""  
  
    df = pd.read_csv(filename, delimiter=";", encoding="utf8", on_bad_lines="skip")  
    df_cleaned = df  
    df_cleaned["Tweet"] = [clean_text(tweet) for tweet in df_cleaned["Tweet"]]  
    df_cleaned["Tokens"] = [len(remove_punct(nltk.word_tokenize(tweet))) for tweet in df_cleaned["Tweet"]]  
    df_cleaned = df_cleaned[df["Tokens"] > min_token] # filtraggio dei tweet in base al numero di token  
  
    df_cleaned.to_csv(f'{filename.replace(".csv", "")}_cleaned.csv', sep=";", index=True, encoding="utf8")  
  
    return df_cleaned
```

Figura 6: funzione *“clean_csv”*

Come si può vedere dalla figura 6, la funzione prende in lettura il file csv che è stato creato nella fase di estrazione dei dati, ne pulisce i tweet mediante la funzione *“clean_text”*; aggiunge una colonna chiamata *“Tokens”* che contiene il numero di token del tweet, ottenuto combinando il tokenizer della libreria *nltk* (*word_tokenize*) con la funzione *“remove_punct”* per escludere dal conteggio dei token la punteggiatura; infine effettua un filtraggio dei tweet mantenendo soltanto quelli che hanno un numero di token maggiore del numero desiderato dall'utente (parametro *“min_token”* della funzione, di default impostato a 10), e restituisce infine un csv contenente i dati puliti.

Il risultato della pulizia dei dati è visibile nella cartella *\data\cleaned_data* del repository.

```
;User;Tweet;Tokens  
0;ConanOBrien;Nobody asked, so here it is: a brand new remote with Jordan Schlansky! ;13  
2;ConanOBrien;. Beyonce, last chance to join me in a duet for my album of Irish dance tunes.;16  
3;ConanOBrien;Right now the FBI is Googling how to get ketchup off sensitive documents.;13  
4;ConanOBrien;A flight attendant just asked me to change my seat because "your head is making the plane imbalanced.";18  
6;ConanOBrien;Talked to a 911 operator on my podcast. I'm still not sure if he called us or if we called him. ;20  
7;ConanOBrien;I wonder if the FBI found all of his burner toilets.;11  
9;ConanOBrien;The car warranty bots stopped calling and now I'm nervous I did something to hurt their feelings.;18  
10;ConanOBrien;I already knew Kim and Pete broke up because Alex Jones's lawyer sent me their text messages.;18  
11;ConanOBrien;I managed to get booked on my wife's podcast. I was her 8th choice. ;13  
12;ConanOBrien;Now I'm glad I turned down the role of Batgirl.;11  
13;ConanOBrien;"Huge thanks to my fan Kristy for informing me about a Facebook page for people who think I'm "bangable."  
14;ConanOBrien;Hear me chat with one of the only cast members from TheOffice who didn't intern for me, bjohnak. ;18
```

Figura 7: csv risultante dalla pulizia dei dati per Conan O'Brien

Come si può vedere in figura 7, il risultato finale è un csv molto più pulito di quello iniziale: non vi sono più caratteri speciali o newlines che inquinano la lettura del csv, e una colonna ulteriore ("*Tokens*") riporta il numero di token del tweet.

Una volta puliti i dati, si è proceduto alla fase successiva: l'addestramento dei modelli GPT-3.

3. Addestramento dei modelli GPT-3

L'addestramento dei modelli avviene nei due script, "*C_GPT-3.py*" e "*D_question_generation.py*".

3.1 Creazione del modello

Come sottolineato nell'introduzione, l'obiettivo del progetto è l'addestramento di 3 diversi modelli sulla base dei tweet di 3 account di Twitter, che imparino a rispondere a delle domande come se fossero gli utenti stessi a rispondere, emulando dunque il loro linguaggio, le loro scelte lessicali e in generale il loro stile.

Per fare ciò sono state utilizzate le API di OpenAI², che mettono a disposizione diversi potentissimi modelli di intelligenza artificiale per la generazione, comprensione e manipolazione di contenuti linguistici. In particolare, per lo svolgimento del progetto è stato utilizzato "*GPT-3*", un potentissimo modello in grado di comprendere e generare linguaggio naturale mediante deep learning, addestrato su una impressionante mole di dati testuali raccolti su internet. OpenAI mette a disposizione il modello in 4 versioni, in ordine decrescente di "potenza elaborativa": "*davinci*", "*curie*", "*babbage*" e "*ada*". Dopo diversi tentativi si è deciso di utilizzare nel progetto la versione *davinci*, la più potente ma la più "costosa". L'utilizzo delle API di OpenAI, infatti, non è gratuito. Vi sono dei tetti di utilizzo da rispettare ogni qual volta viene fatta una richiesta alle API per addestrare o utilizzare un modello. Il costo è commisurato sulla base del numero di token che il modello processa nell'addestramento e nell'utilizzo del modello. I costi dell'utilizzo del modello GPT-3 "*davinci*" hanno impedito di utilizzare dataset più grandi per l'addestramento del modello.

Per la creazione del modello si è fatto riferimento al "*fine-tuning*"³ di OpenAI, che consente di customizzare uno dei modelli di base addestrandolo con dati predisposti dell'utente per una finalità più specifica di quella dei modelli predefiniti. Ad esempio, in questo caso la finalità del modello sarebbe non solo rispondere ad una domanda (cosa che il modello GPT-3 base sa fare), ma farlo emulando il lessico e lo stile di un utente Twitter.

Il fine-tuning si basa sulla predisposizione di training data caratterizzati da due campi: i "*prompts*" e le "*completions*": i primi corrispondono agli input che l'utente darà al modello; i secondi corrispondono invece alla risposta che il modello dovrebbe dare a quel particolare prompt. Nel caso di questo progetto, i prompts saranno le domande formulate dall'utente; le completions saranno i tweet che sono stati raccolti e

² Documentazione al link: <https://beta.openai.com/docs/introduction>

³ Documentazione disponibile al link: [Fine-tuning - OpenAI API](#)

puliti nelle precedenti fasi. Le domande dovranno dunque essere formulate di modo che il tweet figuri come risposta.

Si è proceduto dunque con la definizione di una funzione, *“create_empty_prompt_df”*, per la creazione di un csv avente due colonne: una con i prompt vuoti, che andranno riempiti manualmente con le domande, e una con le completions, ovvero le risposte alle domande, che non saranno altro che i tweet raccolti.

```
def create_empty_prompt_df(file:str):  
  
    """ Returns a csv with two columns: "prompts" and "completion". Prompts will be empty: they have to be  
    written manually. """  
  
    df = pd.read_csv(file, sep = ";", index_col=0)  
    df["prompt"] = ""  
    df_out = df[["prompt", "Tweet"]]  
    df_out.columns = ["prompt", "completion"]  
  
    df_out.to_csv(f"{file.replace('_cleaned.csv', '')}_training.csv", sep = ";")  
    return df_out
```

Figura 8: funzione *“create_empty_prompt_df”*

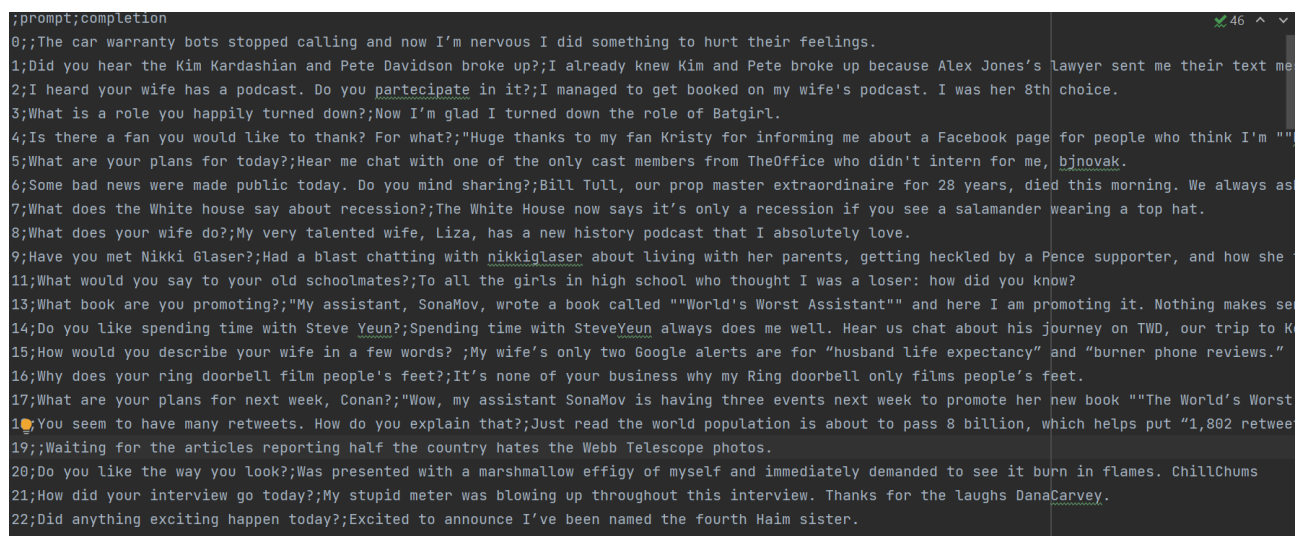
La funzione prende in ingresso un file csv, ovvero in questo caso particolare il csv avente i tweet dei 3 utenti, e restituisce un csv con i prompt vuoti e i tweet, come visibile in figura 9.

prompt	completion
0,,	The car warranty bots stopped calling and now I'm nervous I did something to hurt their feelings.
1,,	I already knew Kim and Pete broke up because Alex Jones's lawyer sent me their text messages.
2,,	I managed to get booked on my wife's podcast. I was her 8th choice.
3,,	Now I'm glad I turned down the role of Batgirl.
4,,	"Huge thanks to my fan Kristy for informing me about a Facebook page for people who think I'm ""bangable."" "
5,,	"Hear me chat with one of the only cast members from TheOffice who didn't intern for me, <u>bjnovak</u> . "
6,,	"Bill Tull, our prop master extraordinaire for 28 years, died this morning. We always asked the impossible of Bill, sometimes minu
7,,	The White House now says it's only a recession if you see a salamander wearing a top hat.
8,,	"My very talented wife, Liza, has a new history podcast that I absolutely love. "
9,,	"Had a blast chatting with <u>nikkiglas</u> about living with her parents, getting heckled by a Pence supporter, and how she thinks I'd
11,,	To all the girls in high school who thought I was a loser: how did you know?
13,,	"My assistant, SonaMov, wrote a book called ""World's Worst Assistant"" and here I am promoting it. Nothing makes sense anymore. "
14,,	"Spending time with <u>SteveYeun</u> always does me well. Hear us chat about his journey on TWD, our trip to Korea, and NopeMovie "
15,,	My wife's only two Google alerts are for "husband life expectancy" and "burner phone reviews."
16,,	It's none of your business why my Ring doorbell only films people's feet.
17,,	"Wow, my assistant SonaMov is having three events next week to promote her new book ""The World's Worst Assistant."" If you're in
18,,	"Just read the world population is about to pass 8 billion, which helps put "1,802 retweets" in perspective."
19,,	Waiting for the articles reporting half the country hates the Webb Telescope photos.
20,,	Was presented with a marshmallow effigy of myself and immediately demanded to see it burn in flames. ChillChums
21,,	My stupid meter was blowing up throughout this interview. Thanks for the laughs <u>DanaCarvey</u> .
22,,	Excited to announce I've been named the fourth Haim sister.
23,,	"...the 3rd of the 4th of the 5th of the 6th of the 7th of the 8th of the 9th of the 10th of the 11th of the 12th of the 13th of the 14th of the 15th of the 16th of the 17th of the 18th of the 19th of the 20th of the 21st of the 22nd of the 23rd of the 24th of the 25th of the 26th of the 27th of the 28th of the 29th of the 30th of the 31st of the 32nd of the 33rd of the 34th of the 35th of the 36th of the 37th of the 38th of the 39th of the 40th of the 41st of the 42nd of the 43rd of the 44th of the 45th of the 46th of the 47th of the 48th of the 49th of the 50th of the 51st of the 52nd of the 53rd of the 54th of the 55th of the 56th of the 57th of the 58th of the 59th of the 60th of the 61st of the 62nd of the 63rd of the 64th of the 65th of the 66th of the 67th of the 68th of the 69th of the 70th of the 71st of the 72nd of the 73rd of the 74th of the 75th of the 76th of the 77th of the 78th of the 79th of the 80th of the 81st of the 82nd of the 83rd of the 84th of the 85th of the 86th of the 87th of the 88th of the 89th of the 90th of the 91st of the 92nd of the 93rd of the 94th of the 95th of the 96th of the 97th of the 98th of the 99th of the 100th of the 101st of the 102nd of the 103rd of the 104th of the 105th of the 106th of the 107th of the 108th of the 109th of the 110th of the 111th of the 112th of the 113th of the 114th of the 115th of the 116th of the 117th of the 118th of the 119th of the 120th of the 121st of the 122nd of the 123rd of the 124th of the 125th of the 126th of the 127th of the 128th of the 129th of the 130th of the 131st of the 132nd of the 133rd of the 134th of the 135th of the 136th of the 137th of the 138th of the 139th of the 140th of the 141st of the 142nd of the 143rd of the 144th of the 145th of the 146th of the 147th of the 148th of the 149th of the 150th of the 151st of the 152nd of the 153rd of the 154th of the 155th of the 156th of the 157th of the 158th of the 159th of the 160th of the 161st of the 162nd of the 163rd of the 164th of the 165th of the 166th of the 167th of the 168th of the 169th of the 170th of the 171st of the 172nd of the 173rd of the 174th of the 175th of the 176th of the 177th of the 178th of the 179th of the 180th of the 181st of the 182nd of the 183rd of the 184th of the 185th of the 186th of the 187th of the 188th of the 189th of the 190th of the 191st of the 192nd of the 193rd of the 194th of the 195th of the 196th of the 197th of the 198th of the 199th of the 200th of the 201st of the 202nd of the 203rd of the 204th of the 205th of the 206th of the 207th of the 208th of the 209th of the 210th of the 211th of the 212th of the 213th of the 214th of the 215th of the 216th of the 217th of the 218th of the 219th of the 220th of the 221st of the 222nd of the 223rd of the 224th of the 225th of the 226th of the 227th of the 228th of the 229th of the 230th of the 231st of the 232nd of the 233rd of the 234th of the 235th of the 236th of the 237th of the 238th of the 239th of the 240th of the 241st of the 242nd of the 243rd of the 244th of the 245th of the 246th of the 247th of the 248th of the 249th of the 250th of the 251st of the 252nd of the 253rd of the 254th of the 255th of the 256th of the 257th of the 258th of the 259th of the 260th of the 261st of the 262nd of the 263rd of the 264th of the 265th of the 266th of the 267th of the 268th of the 269th of the 270th of the 271st of the 272nd of the 273rd of the 274th of the 275th of the 276th of the 277th of the 278th of the 279th of the 280th of the 281st of the 282nd of the 283rd of the 284th of the 285th of the 286th of the 287th of the 288th of the 289th of the 290th of the 291st of the 292nd of the 293rd of the 294th of the 295th of the 296th of the 297th of the 298th of the 299th of the 300th of the 301st of the 302nd of the 303rd of the 304th of the 305th of the 306th of the 307th of the 308th of the 309th of the 310th of the 311th of the 312th of the 313th of the 314th of the 315th of the 316th of the 317th of the 318th of the 319th of the 320th of the 321st of the 322nd of the 323rd of the 324th of the 325th of the 326th of the 327th of the 328th of the 329th of the 330th of the 331st of the 332nd of the 333rd of the 334th of the 335th of the 336th of the 337th of the 338th of the 339th of the 340th of the 341st of the 342nd of the 343rd of the 344th of the 345th of the 346th of the 347th of the 348th of the 349th of the 350th of the 351st of the 352nd of the 353rd of the 354th of the 355th of the 356th of the 357th of the 358th of the 359th of the 360th of the 361st of the 362nd of the 363rd of the 364th of the 365th of the 366th of the 367th of the 368th of the 369th of the 370th of the 371st of the 372nd of the 373rd of the 374th of the 375th of the 376th of the 377th of the 378th of the 379th of the 380th of the 381st of the 382nd of the 383rd of the 384th of the 385th of the 386th of the 387th of the 388th of the 389th of the 390th of the 391st of the 392nd of the 393rd of the 394th of the 395th of the 396th of the 397th of the 398th of the 399th of the 400th of the 401st of the 402nd of the 403rd of the 404th of the 405th of the 406th of the 407th of the 408th of the 409th of the 410th of the 411th of the 412th of the 413th of the 414th of the 415th of the 416th of the 417th of the 418th of the 419th of the 420th of the 421st of the 422nd of the 423rd of the 424th of the 425th of the 426th of the 427th of the 428th of the 429th of the 430th of the 431st of the 432nd of the 433rd of the 434th of the 435th of the 436th of the 437th of the 438th of the 439th of the 440th of the 441st of the 442nd of the 443rd of the 444th of the 445th of the 446th of the 447th of the 448th of the 449th of the 450th of the 451st of the 452nd of the 453rd of the 454th of the 455th of the 456th of the 457th of the 458th of the 459th of the 460th of the 461st of the 462nd of the 463rd of the 464th of the 465th of the 466th of the 467th of the 468th of the 469th of the 470th of the 471st of the 472nd of the 473rd of the 474th of the 475th of the 476th of the 477th of the 478th of the 479th of the 480th of the 481st of the 482nd of the 483rd of the 484th of the 485th of the 486th of the 487th of the 488th of the 489th of the 490th of the 491st of the 492nd of the 493rd of the 494th of the 495th of the 496th of the 497th of the 498th of the 499th of the 500th of the 501st of the 502nd of the 503rd of the 504th of the 505th of the 506th of the 507th of the 508th of the 509th of the 510th of the 511th of the 512th of the 513th of the 514th of the 515th of the 516th of the 517th of the 518th of the 519th of the 520th of the 521st of the 522nd of the 523rd of the 524th of the 525th of the 526th of the 527th of the 528th of the 529th of the 530th of the 531st of the 532nd of the 533rd of the 534th of the 535th of the 536th of the 537th of the 538th of the 539th of the 540th of the 541st of the 542nd of the 543rd of the 544th of the 545th of the 546th of the 547th of the 548th of the 549th of the 550th of the 551st of the 552nd of the 553rd of the 554th of the 555th of the 556th of the 557th of the 558th of the 559th of the 560th of the 561st of the 562nd of the 563rd of the 564th of the 565th of the 566th of the 567th of the 568th of the 569th of the 570th of the 571st of the 572nd of the 573rd of the 574th of the 575th of the 576th of the 577th of the 578th of the 579th of the 580th of the 581st of the 582nd of the 583rd of the 584th of the 585th of the 586th of the 587th of the 588th of the 589th of the 590th of the 591st of the 592nd of the 593rd of the 594th of the 595th of the 596th of the 597th of the 598th of the 599th of the 600th of the 601st of the 602nd of the 603rd of the 604th of the 605th of the 606th of the 607th of the 608th of the 609th of the 610th of the 611th of the 612th of the 613th of the 614th of the 615th of the 616th of the 617th of the 618th of the 619th of the 620th of the 621st of the 622nd of the 623rd of the 624th of the 625th of the 626th of the 627th of the 628th of the 629th of the 630th of the 631st of the 632nd of the 633rd of the 634th of the 635th of the 636th of the 637th of the 638th of the 639th of the 640th of the 641st of the 642nd of the 643rd of the 644th of the 645th of the 646th of the 647th of the 648th of the 649th of the 650th of the 651st of the 652nd of the 653rd of the 654th of the 655th of the 656th of the 657th of the 658th of the 659th of the 660th of the 661st of the 662nd of the 663rd of the 664th of the 665th of the 666th of the 667th of the 668th of the 669th of the 670th of the 671st of the 672nd of the 673rd of the 674th of the 675th of the 676th of the 677th of the 678th of the 679th of the 680th of the 681st of the 682nd of the 683rd of the 684th of the 685th of the 686th of the 687th of the 688th of the 689th of the 690th of the 691st of the 692nd of the 693rd of the 694th of the 695th of the 696th of the 697th of the 698th of the 699th of the 700th of the 701st of the 702nd of the 703rd of the 704th of the 705th of the 706th of the 707th of the 708th of the 709th of the 710th of the 711th of the 712th of the 713th of the 714th of the 715th of the 716th of the 717th of the 718th of the 719th of the 720th of the 721st of the 722nd of the 723rd of the 724th of the 725th of the 726th of the 727th of the 728th of the 729th of the 730th of the 731st of the 732nd of the 733rd of the 734th of the 735th of the 736th of the 737th of the 738th of the 739th of the 740th of the 741st of the 742nd of the 743rd of the 744th of the 745th of the 746th of the 747th of the 748th of the 749th of the 750th of the 751st of the 752nd of the 753rd of the 754th of the 755th of the 756th of the 757th of the 758th of the 759th of the 760th of the 761st of the 762nd of the 763rd of the 764th of the 765th of the 766th of the 767th of the 768th of the 769th of the 770th of the 771st of the 772nd of the 773rd of the 774th of the 775th of the 776th of the 777th of the 778th of the 779th of the 780th of the 781st of the 782nd of the 783rd of the 784th of the 785th of the 786th of the 787th of the 788th of the 789th of the 790th of the 791st of the 792nd of the 793rd of the 794th of the 795th of the 796th of the 797th of the 798th of the 799th of the 800th of the 801st of the 802nd of the 803rd of the 804th of the 805th of the 806th of the 807th of the 808th of the 809th of the 810th of the 811th of the 812th of the 813th of the 814th of the 815th of the 816th of the 817th of the 818th of the 819th of the 820th of the 821st of the 822nd of the 823rd of the 824th of the 825th of the 826th of the 827th of the 828th of the 829th of the 830th of the 831st of the 832nd of the 833rd of the 834th of the 835th of the 836th of the 837th of the 838th of the 839th of the 840th of the 841st of the 842nd of the 843rd of the 844th of the 845th of the 846th of the 847th of the 848th of the 849th of the 850th of the 851st of the 852nd of the 853rd of the 854th of the 855th of the 856th of the 857th of the 858th of the 859th of the 860th of the 861st of the 862nd of the 863rd of the 864th of the 865th of the 866th of the 867th of the 868th of the 869th of the 870th of the 871st of the 872nd of the 873rd of the 874th of the 875th of the 876th of the 877th of the 878th of the 879th of the 880th of the 881st of the 882nd of the 883rd of the 884th of the 885th of the 886th of the 887th of the 888th of the 889th of the 890th of the 891st of the 892nd of the 893rd of the 894th of the 895th of the 896th of the 897th of the 898th of the 899th of the 900th of the 901st of the 902nd of the 903rd of the 904th of the 905th of the 906th of the 907th of the 908th of the 909th of the 910th of the 911th of the 912th of the 913th of the 914th of the 915th of the 916th of the 917th of the 918th of the 919th of the 920th of the 921st of the 922nd of the 923rd of the 924th of the 925th of the 926th of the 927th of the 928th of the 929th of the 930th of the 931st of the 932nd of the 933rd of the 934th of the 935th of the 936th of the 937th of the 938th of the 939th of the 940th of the 941st of the 942nd of the 943rd of the 944th of the 945th of the 946th of the 947th of the 948th of the 949th of the 950th of the 951st of the 952nd of the 953rd of the 954th of the 955th of the 956th of the 957th of the 958th of the 959th of the 960th of the 961st of the 962nd of the 963rd of the 964th of the 965th of the 966th of the 967th of the 968th of the 969th of the 970th of the 971st of the 972nd of the 973rd of the 974th of the 975th of the 976th of the 977th of the 978th of the 979th of the 980th of the 981st of the 982nd of the 983rd of the 984th of the 985th of the 986th of the 987th of the 988th of the 989th of the 990th of the 991st of the 992nd of the 993rd of the 994th of the 995th of the 996th of the 997th of the 998th of the 999th of the 1000th of the 1001st of the 1002nd of the 1003rd of the 1004th of the 1005th of the 1006th of the 1007th of the 1008th of the 1009th of the 1010th of the 1011th of the 1012th of the 1013th of the 1014th of the 1015th of the 1016th of the 1017th of the 1018th of the 1019th of the 1020th of the 1021st of the 1022nd of the 1023rd of the 1024th of the 1025th of the 1026th of the 1027th of the 1028th of the 1029th of the 1030th of the 1031st of the 1032nd of the 1033rd of the 1034th of the 1035th of the 1036th of the 1037th of the 1038th of the 1039th of the 1040th of the 1041st of the 1042nd of the 1043rd of the 1044th of the 1045th of the 1046th of the 1047th of the 1048th of the 1049th of the 1050th of the 1051st of the 1052nd of the 1053rd of the 1054th of the 1055th of the 1056th of the 1057th of the 1058th of the 1059th of the 1060th of the 1061st of the 1062nd of the 1063rd of the 1064th of the 1065th of the 1066th of the 1067th of the 1068th of the 1069th of the 1070th of the 1071st of the 1072nd of the 1073rd of the 1074th of the 1075th of the 1076th of the 1077th of the 1078th of the 1079th of the 1080th of the 1081st of the 1082nd of the 1083rd of the 1084th of the 1085th of the 1086th of the 1087th of the 1088th of the 1089th of the 1090th of the 1091st of the 1092nd of the 1093rd of the 1094th of the 1095th of the 1096th of the 1097th of the 1098th of the 1099th of the 1100th of the 1101st of the 1102nd of the 1103rd of the 1104th of the 1105th of the 1106th of the 1107th of the 1108th of the 1109th of the 1110th of the 1111th of the 1112th of the 1113th of the 1114th of the 1115th of the 1116th of the 1117th of the 1118th of the 1119th of the 1120th of the 1121st of the 1122nd of the 1123rd of the 1124th of the 1125th of the 1126th of the 1127th of the 1128th of the 1129th of the 1130th of the 1131st of the 1132nd of the 1133rd of the 1134th of the 1135th of the 1136th of the 1137th of the 1138th of the 1139th of the 1140th of the 1141st of the 1142nd of the 1143rd of the 1144th of the 1145th of the 1146th of the 1147th of the 1148th of the 1149th of the 1150th of the 1151st of the 1152nd of the 1153rd of the 1154th of the 1155th of the 1156th of the 1157th of the 1158th of the 1159th of the 1160th of the 1161st of the 1162nd of the 1163rd of the 1164th of the 1165th of the 1166th of the 1167th of the 1168th of the 1169th of the 1170th of the 1171st of the 1172nd of the 1173rd of the 1174th of the 1175th of the 1176th of the 1177th of the 1178th of the 1179th of the 1180th of the 1181st of the 1182nd of the 1183rd of the 1184th of the 1185th of the 1186th of the 1187th of the 1188th of the 1189th of the 1190th of the 1191st of the 1192nd of the 1193rd of the 1194th of the 1195th of the 1196th of the 1197th of the 1198th of the 1199th of the 1200th of the 1201st of the 1202nd of the 1203rd of the 1204th of the 1205th of the 1206th of the 1207th of the 1208th of the 1209th of the 1210th of the 1211th of the 1212th of the 1213th of the 1214th of the 1215th of the 1216th of the 1217th of the 1218th of the 1219th of the 1220th of the 1221st of the 1222nd of the 1223rd of the 1224th of the 1225th of the 1226th of the 1227th of the 1228th of the 1229th of the 1230th of the 1231st of the 1232nd of the 1233rd of the 1234th of the 1235th of the 1236th of the 1237th of the 1238th of the 1239th of the 1240th of the 1241st of the 1242nd of the 1243rd of the 1244th of the 1245th of the 1246th of the 1247th of the 1248th of the 1249th of the 1250th of the 1251st of the 1252nd of the 1253rd of the 1254th of the 1255th of the 1256th of the 1257th of the 1258th of the 1259th of the 1260th of the 1261st of the 1262nd of the 1263rd of the 1264th of the 1265th of the 1266th of the 1267th of the 1268th of the 1269th of the 1270th of the 1271st of the 1272nd of the 1273rd of the 1274th of the 1275th of the 1276th of the 1277th of the 1278th of the 1279th of the 1280th of the 1281st of the 1282nd of the 1283rd of the 1284th of the 1285th of the 1286th of the 1287th of the 1288th of the 1289th of the 1290th of the 1291st of the 1292nd of the 1293rd of the 1294th of the 1295th of the 1296th of the 1297th of the 1298th of the 1299th of the 1300th of the 1301st of the 1302nd of the 1303rd of the 1304th of the 1305th of the 1306th of the 1307th of the 1308th of the 1309th of the 1310th of the 1311th of the 1312th of the 1313th of the 1314th of the 1315th of the 1316th of the 1317th of the 1318th of the 1319th of the 1320th of the 1321st of the 1322nd of the 1323rd of the 1324th of the 1325th of the 1326th of the 1327th of the 1328th of the 1329th of the 1330th of the 1331st of the 1332nd of the 1333rd of the 1334th of the 1335th of the 1336th of the 1337th of the 1338th of the 1339th of the 1340th of the 1341st of the 1342nd of the 1343rd of the 1344th of the 1345th of the 1346th of the 1347th of the 1348th of the 1349th of the 1350th of the 1351st of the 1352nd of the 1353rd of the 1354th of the 1355th of the 1356th of the 1357th of the 1358th of the 1359th of the 1360th of the 1361st of the 1362nd of the 1363rd of the 1364th of the 1365th of the 1366th of the 1367th of the 1368th of the 1369th of the 1370th of the 1371st of the 1372nd of the 1373rd of the 1374th of the 1375th of the 1376th of the 1377th of the 1378th of the 1379th of the 1380th of the 1381st of the 1382nd of the 1383rd of the 1384th of the 1385th of the 1386th of the 1387th of the 1388th of the 1389th of the 1390th of the 1391st of the 1392nd of the 1393rd of the 1394th of the 1395th of the 1396th of the 1397th of the 1398th of the 1399th of the 1400th of the 1401st of the 1402nd of the 1403rd of the 1404th of the 1405th of the 1406th of the 1407th of the 1408th of the 1409th of the 1410th of the 1411th of the 1412th of the 1413th of the 1414th of the 1415th of the 1416th of the 1417th of the 1418th of the 1419th of the 1420th of the 1421st of the 1422nd of the 1423rd of the 1424th of the 1425th of the 1426th of the 1427th of the 1428th of the 1429th of the 1430th of the 1431st of the 1432nd of the 1433rd of the 1434th of the 1435th of the 1436th of the 1437th of the 1438th of the 1439th of the 1440th of the 1441st of the 1442nd of the 1443rd of the 1444th of the 1445th of the 1446th of the 1447th of the 1448th of the 1449th of the 1450th of the 1451st of the 1452nd of the 1453rd of the 1454th of the 1455th of the 1456th of the 1457th of the 1458th of the 1459th of the 1460th of the 1461st of the 1462nd of the 1463rd of the 1464th of the 1465th of the 1466th of the 1467th of the 1468th of the 1469th of the 1470th of the 1471st of the 1472nd of the 1473rd of the 1474th of the 1475th of the 1476th of the 1477th of the 1478th of the 1479th of the 1480th of the 1481st of the 1482nd of the 1483rd of the 1484th of the 1485th of the 1486th of the 1487th of the 1488th of the 1489th of the 1490th of the 1491st of the 1492nd of the 1493rd of the 1494th of the 1495th of the 1496th of the 1497th of the 1498th of the 1499th of the 1500th of the 1501st of the 1502nd of the 1503rd of the 1504th of the 1505th of the 1506th of the 1507th of the 1508th of the 1509th of the 1510th of the 1511th of the 1512th of the 1513th of the 1514th of the 1515th of the 1516th of the 1517th of the 1518th of the 1519th of the 1520th of the 1521st of the

In questo progetto per tutti e tre gli user sono stati raccolti 150 tweet, ma per motivi di tempo, di costo dell'utilizzo delle API di OpenAI, e per il fatto che non tutti i tweet fossero idonei, per il loro contenuto, alla formulazione di domande⁴, sono state predisposte 75 domande. Il 150 in fase di raccolta era infatti un numero "conservativo" per far fronte a queste esigenze e consentire di selezionare i tweet più idonei e ricchi di contenuto.

Si è proceduto con la compilazione manuale dei prompts per i tre utenti. Il risultato della compilazione manuale dei prompt è visibile nella cartella "`\data\training_data`" del progetto, in particolare nei file aventi il nome dell'utente di riferimento e il suffisso "`_training.csv`".

Nel caso dell'utente Conan O' Brien, il risultato della compilazione manuale è visibile in figura 10:



prompt	completion
0;;The car warranty bots stopped calling and now I'm nervous I did something to hurt their feelings.	
1;Did you hear the Kim Kardashian and Pete Davidson broke up?;I already knew Kim and Pete broke up because Alex Jones's Lawyer sent me their text me	
2;I heard your wife has a podcast. Do you participate in it?;I managed to get booked on my wife's podcast. I was her 8th choice.	
3;What is a role you happily turned down?;Now I'm glad I turned down the role of Batgirl.	
4;Is there a fan you would like to thank? For what?;"Huge thanks to my fan Kristy for informing me about a Facebook page for people who think I'm ""	
5;What are your plans for today?;Hear me chat with one of the only cast members from TheOffice who didn't intern for me, bjnovak.	
6;Some bad news were made public today. Do you mind sharing?;Bill Tull, our prop master extraordinaire for 28 years, died this morning. We always as	
7;What does the White house say about recession?;The White House now says it's only a recession if you see a salamander wearing a top hat.	
8;What does your wife do?;My very talented wife, Liza, has a new history podcast that I absolutely love.	
9;Have you met Nikki Glaser?;Had a blast chatting with nikkiglaser about living with her parents, getting heckled by a Pence supporter, and how she	
11;What would you say to your old schoolmates?;To all the girls in high school who thought I was a loser: how did you know?	
13;What book are you promoting?;"My assistant, SonaMov, wrote a book called ""World's Worst Assistant"" and here I am promoting it. Nothing makes se	
14;Do you like spending time with Steve Yeun?;Spending time with SteveYeun always does me well. Hear us chat about his journey on TWD, our trip to K	
15;How would you describe your wife in a few words? ;My wife's only two Google alerts are for "husband life expectancy" and "burner phone reviews."	
16;Why does your ring doorbell film people's feet?;It's none of your business why my Ring doorbell only films people's feet.	
17;What are your plans for next week, Conan?;"Wow, my assistant SonaMov is having three events next week to promote her new book ""The World's Worst	
18;You seem to have many retweets. How do you explain that?;Just read the world population is about to pass 8 billion, which helps put "1,802 retwee	
19;;Waiting for the articles reporting half the country hates the Webb Telescope photos.	
20;Do you like the way you look?;Was presented with a marshmallow effigy of myself and immediately demanded to see it burn in flames. ChillChums	
21;How did your interview go today?;My stupid meter was blowing up throughout this interview. Thanks for the laughs DanaCarvey.	
22;Did anything exciting happen today?;Excited to announce I've been named the fourth Haim sister.	

Figura 10: prompt compilati manualmente per l'utente Conan O' Brien

Come si può vedere dalla figura, non tutte le righe sono state compilate. Alcuni prompt sono stati lasciati vuoti, proprio per selezionare quei tweet che risultavano più idonei per la formulazione di una domanda.

I file così ottenuti sono stati ulteriormente processati attraverso la funzione "`prepare_data`".

⁴ In alcuni casi, i tweet presentavano un contenuto così specifico o randomico da rendere impossibile o "forzata" la formulazione di una domanda adeguata

```
def prepare_data(file_training):

    """Returns a raw data training file to a json without empty columns, ready to be formatted in jsonl format via the
    openAI CLI data formatting tool"""

    df = pd.read_csv(file_training, sep=";", header=0, usecols=[1,2])
    df_out = df.dropna()
    df_out.to_json(f"{file_training.replace('.csv','')}prepared.json", indent=4)
    print(df_out.shape)_# to check number of rows

    return df_out
```

Figura 11: funzione "prepare_data"

La funzione prende in input il file ottenuto precedentemente, ne elimina le righe i cui prompt sono stati lasciati vuoti mediante il metodo `“dropna()”` e restituisce un file *json* avente due campi, “prompt” (ovvero le domande appena formulate) e “completion”, (ovvero i tweet) per ciascun utente. Tale file è pronto per essere processato mediante un tool messo a disposizione da OpenAI per la formattazione dei dati, chiamato *“CLI data preparation tool”*⁵. OpenAI suggerisce infatti, per il *fine-tuning* di un modello, di seguire delle best practices⁶ per la formattazione dei training data in modo da aumentare l’efficacia dell’addestramento del modello. In particolare, OpenAI suggerisce:

- L’aggiunta di un separatore alla fine di ogni prompt, per indicare al modello quando finisce il “prompt” e quando inizia la “completion”. Il separatore non deve comparire in altre parti del prompt. In questo caso il separatore sarà il simbolo “->”;
- L’aggiunta di una sequenza di caratteri di “stop” a ciascuna “completion”, per indicare al modello quando la completion termina. Nel nostro caso verrà utilizzata la newline “\n”;
- L’aggiunta di un whitespace all’inizio di ogni completion, in quanto ciò agevola la tokenizzazione di OpenAI che tokenizza le parole precedute da uno whitespace.
- La conversione del file di training in formato jsonl(“json lines”⁷), un particolare formato di json che agevola il training del fine-tuned model.

È stato dunque utilizzato il *“CLI data preparation tool”* per la formattazione dei dati. Tale tool viene utilizzato mediante una serie di comandi da terminale, visibili in figura 12:

⁵ Documentazione al link: [Fine-tuning - OpenAI API](#) , nella sezione “Prepare Training Data”

⁶ Documentazione in merito consultabile al link: [Fine-tuning - OpenAI API](#)

⁷ Informazioni su questo formato reperibili al link <https://jsonlines.org/>


```
PS C:\Users\mario\PycharmProjects\00262_meloni> setx OPENAI_API_KEY "sk-Etk1CABRWKfFj0ImW9HBt3B1bkFJAPbso5TSQ0TS90xdC6po"

OPERAZIONE RIUSCITA: valore specificato salvato.
PS C:\Users\mario\PycharmProjects\00262_meloni> openai tools fine_tunes.prepare_data -f ConanOBrien_training_prepared.json
Logging requires wandb to be installed. Run 'pip install wandb'.
Analyzing...

- Your file appears to be in a .JSON format. Your file will be converted to JSONL format
- Your file contains 75 prompt-completion pairs. In general, we recommend having at least a few hundred examples. We've found that performance tends to linearly increase for every doubling of the number of examples
- Your data does not contain a common separator at the end of your prompts. Having a separator string appended to the end of the prompt makes it clearer to the fine-tuned model where the completion should begin. See https://beta.openai.com/docs/guides/fine-tuning/preparing-your-dataset for more detail and examples. If you intend to do open-ended generation, then you should leave the prompts empty
- Your data does not contain a common ending at the end of your completions. Having a common ending string appended to the end of the completion makes it clearer to the fine-tuned model where the completion should end. See https://beta.openai.com/docs/guides/fine-tuning/preparing-your-dataset for more detail and examples.
- The completion should start with a whitespace character (' '). This tends to produce better results due to the tokenization we use. See https://beta.openai.com/docs/guides/fine-tuning/preparing-your-dataset for more details

Based on the analysis we will perform the following actions:
- [Necessary] Your format 'JSON' will be converted to 'JSONL'
- [Recommended] Add a suffix separator ' ->' to all prompts [Y/n]:
```

```
Based on the analysis we will perform the following actions:
- [Necessary] Your format 'JSON' will be converted to 'JSONL'
- [Recommended] Add a suffix separator ' ->' to all prompts [Y/n]: Y
- [Recommended] Add a suffix ending '\n' to all completions [Y/n]: Y
- [Recommended] Add a whitespace character to the beginning of the completion [Y/n]: Y

Your data will be written to a new JSONL file. Proceed [Y/n]: Y

Wrote modified file to 'ConanOBrien_training_prepared_prepared.jsonl'
Feel free to take a look!

Now use that file when fine-tuning:
> openai api fine_tunes.create -t "ConanOBrien_training_prepared_prepared.jsonl"

After you've fine-tuned a model, remember that your prompt has to end with the indicator string ' ->' for the model to start generating completions, rather than continuing with the prompt. Make sure to include 'stop=["\n"]' so that the generated texts ends at the expected place.
Once your model starts training, it'll approximately take 5.68 minutes to train a 'curie' model, and less for 'ada' and 'babbage'. Queue will approximately take half an hour per job ahead of you.
```

Figura 12: utilizzo del CLI data preparation tool da terminale per la formattazione dei dati

Come si può vedere dalla figura 12, è prima necessaria l'autenticazione mediante l'inserimento della OpenAI “*api_key*” che viene conferita all'utente per l'utilizzo delle API di OpenAI. Dopodiché il tool chiede all'utente se vuole modificare il training file con i diversi accorgimenti citati in precedenza.

Una volta accettate tali modifiche, viene restituito un file in formato jsonl, avente il seguente aspetto (figura 13):

```
{
  "prompt": "Did you hear the Kim Kardashian and Pete Davidson broke up? ->",
  "completion": " I already knew Kim and Pete broke up because Alex \u270529 ^ \u25bc"
},
{
  "prompt": "I heard your wife has a podcast. Do you partecipate in it? ->",
  "completion": " I managed to get booked on my wife's podcast. I was her 8th"
},
{
  "prompt": "What is a role you happily turned down? ->",
  "completion": " Now I'm glad I turned down the role of Batgirl.\n"
},
{
  "prompt": "Is there a fan you would like to thank? For what? ->",
  "completion": " Huge thanks to my fan Kristy for informing me about a Facebook page f"
},
{
  "prompt": "What are your plans for today? ->",
  "completion": " Hear me chat with one of the only cast members from TheOffice who didn't intern for me,"
},
{
  "prompt": "Some bad news were made public today. Do you mind sharing? ->",
  "completion": " Bill Tull, our prop master extraordinaire for 28 years, died"
},
{
  "prompt": "What does the White house say about recession? ->",
  "completion": " The White House now says it's only a recession if you see a salamander W"
},
{
  "prompt": "What does your wife do? ->",
  "completion": " My very talented wife, Liza, has a new history podcast that I absolutely love.\n"
},
{
  "prompt": "Have you met Nikki Glaser? ->",
  "completion": " Had a blast chatting with nikkiglaser about living with her parents, getting heckled by a Pe"
},
{
  "prompt": "What would you say to your old schoolmates? ->",
  "completion": " To all the girls in high school who thought I was a loser: how did you know"
},
{
  "prompt": "What book are you promoting? ->",
  "completion": " My assistant, SonaMov, wrote a book called \"World's Worst Assistant\" and here I am promo"
},
{
  "prompt": "Do you like spending time with Steve Yeun? ->",
  "completion": " Spending time with SteveYeun always does me well. Hear us chat about his jou"
},
{
  "prompt": "How would you describe your wife in a few words? ->",
  "completion": " My wife's only two Google alerts are for \"husband life expectancy\" an"
},
{
  "prompt": "Why does your ring doorbell film people's feet? ->",
  "completion": " It's none of your business why my Ring doorbell only films people's fee"
},
{
  "prompt": "What are your plans for next week, Conan? ->",
  "completion": " Wow, my assistant SonaMov is having three events next week to promote her new"
},
{
  "prompt": "You seem to have many retweets. How do you explain that? ->",
  "completion": " Just read the world population is about to pass 8 billion, whi"
},
{
  "prompt": "Do you like the way you look? ->",
  "completion": " Was presented with a marshmallow effigy of myself and immediately demanded to see it burn"
}
```

Figura 13: file jsonl per l'addestramento del modello

A questo punto è possibile addestrare i fine-tuned models, uno per utente. Ciò avviene sempre mediante un comando che viene dato da terminale, in figura 14.

```
PS C:\Users\mario\PycharmProjects\00262_meloni> openai api fine_tunes.create -t ConanOBrien_training_prepared_prepared.jsonl -m davinci --suffix "ConanOBrien_Q&A"
```

Figura 14: comando per la creazione del fine-tuned model

Con questo comando viene specificato di creare un fine-tuned model avente come modello base il “davinci”, di nome ConanOBrien_Q&A, addestrato utilizzando il file jsonl predisposto in precedenza. Lo stesso comando è stato dato per gli altri due users che sono stati scelti per il progetto.

La creazione del modello richiede diversi minuti. In terminale vengono riportate le diverse fasi della creazione, come visibile in figura 15.

```
[2022-08-11 12:57:00] Created fine-tune: ft-DmdHaQZhBNMnR1hx1oBp8iTr
[2022-08-11 12:57:03] Fine-tune costs $0.33
[2022-08-11 12:57:03] Fine-tune enqueued. Queue number: 0
[2022-08-11 12:59:09] Fine-tune started
[2022-08-11 13:02:03] Completed epoch 1/4
[2022-08-11 13:02:26] Completed epoch 2/4
[2022-08-11 13:02:49] Completed epoch 3/4
[2022-08-11 13:03:12] Completed epoch 4/4
[2022-08-11 13:03:45] Uploaded model: davinci:ft-personal:conanobrien-q-a-2022-08-11-11-03-44
[2022-08-11 13:03:54] Uploaded result file: file-N9UpsHn2wT9rZ9YyDm0aq2aM
[2022-08-11 13:03:54] Fine-tune succeeded

Job complete! Status: succeeded
```

Figura 15: creazione del modello

Una volta creato il modello, cosa che è stata fatta per tutti e tre gli utenti, è possibile utilizzarlo su python mediante la libreria “openai”, che consente di mandare richieste alle API di OpenAI e di utilizzare i loro modelli.

Una volta importata la libreria, sarà necessario impostare l’api_key⁸ che viene affidata all’utente per effettuare richieste alle API. Ciò viene fatto mediante il comando:

```
openai.api_key = OPENAI_API_KEY
```

Figura 16: comando per l'autenticazione per le API di OpenAI

Dopo di che, per testare il funzionamento del modello, è stata formulata a ciascun modello, o in un certo senso a ciascun “utente artificiale”, una domanda.

⁸ Tale api_key si trova nel file “credentials.py” del repository

```
# Joe Biden
response = openai.Completion.create(
    model= "davinci:ft-personal:joebiden-q-a-2022-08-11-12-52-05",
    prompt="President Biden, answer this question concisely: \n\n Q: What are your plans for today? ->",
    max_tokens= 100,
    n= 1,
    frequency_penalty= 1
)

print(response["choices"][0]["text"])
```

Figura 17: prova del modello "Joe Biden" mediante la sottoposizione di una domanda

In questo caso è stato chiesto a “Joe Biden” (ovvero la sua “replica artificiale”, addestrata mediante i suoi tweet), quali fossero i suoi piani per la giornata di oggi. La sua risposta, riportata di seguito per pura curiosità intellettuale, è stata: *“Before I leave for Camp David, I’ll meet with local officials and tour a treatment center in Rockville, Maryland—the epicenter of this growing national emergency...”*

Notare i parametri che sono stati utilizzati per l’ottenimento della risposta. In primo luogo, il parametro “model”, in cui viene specificato quale modello utilizzare. Nel nostro caso, è stato inserito l’identificativo del modello creato mediante fine-tuning a partire dai prompt formulati manualmente e dai tweet. In secondo luogo, il parametro “prompt”, in cui inserire una stringa contenente le indicazioni da dare al modello GPT-3. Si è deciso, per questo primo esempio, di utilizzare come prompt la seguente frase: *“President Biden, answer this question concisely: \n\n Q: What are your plans for today? ->”*. Il parametro “max_tokens” indica invece il numero massimo di token che il modello dovrà utilizzare nella sua risposta. È stato scelto 100 per, da un lato, ottenere risposte abbastanza lunghe, dall’altro evitare di utilizzare troppi token, in quanto come sottolineato in precedenza le richieste all’API di OpenAI hanno un costo commisurato al numero di token processati dal modello. Il parametro “n” indica il numero di “completions” (ovvero risposte) che dovranno essere fornite dal modello. È stato impostato ad 1 in quanto si è interessati ad ottenere una singola risposta. Infine, il parametro “frequency_penalty” è un valore che va da -2 a +2 che “penalizza” l’utilizzo di token che sono stati già utilizzati. Più è alto questo valore, meno è probabile che il modello tenda a ripetere le stesse parole o frasi verbatim. È stato impostato a 1, in quanto si è constatato empiricamente durante il test del modello che senza questo parametro il modello tendeva a ripetere le stesse parole.

La richiesta all’API eseguita restituisce un oggetto chiamato *“OpenAI text_completion object”*, assegnato in figura 17 alla variabile “response”, che contiene i dati in formato json. È possibile vedere l’intera struttura dati di questo oggetto in figura 18:

```

>>> response
<OpenAIObject text_completion id=cmpl-5gUZeMl0sXInQPJ6hKZQHRrL9ntb7 at 0x2a8f830fc40> JSON: {
  "choices": [
    {
      "finish_reason": "length",
      "index": 0,
      "logprobs": null,
      "text": " My message today is that our plan to rescue our COVID/VID isn't just fighting the s
    }
  ],
  "created": 1660808986,
  "id": "cmpl-5gUZeMl0sXInQPJ6hKZQHRrL9ntb7",
  "model": "davinci:ft-personal:joebiden-q-a-2022-08-11-12-52-05",
  "object": "text_completion",
  "usage": {
    "completion_tokens": 100,
    "prompt_tokens": 21,
    "total_tokens": 121
  }
}

```

Figura 18: OpenAI text_completion object

L'oggetto contiene numerose informazioni relative al risultato della richiesta all'API, come il nome del modello utilizzato, il numero di tokens della "completion", il numero di token del "prompt", il motivo per cui la completion è terminata. Il testo della completion, ovvero la risposta del modello alla domanda, si trova alla voce "text" visibile in figura. Per accedervi in python è necessario esaminare la struttura dati dell'oggetto: si tratta di un dizionario, avente diverse chiavi; alla chiave "choices" corrisponde come valore una lista, contenente al suo interno un altro dizionario, avente al suo interno il valore "text" che si vuole estrarre. Per questo, per ottenere la risposta in figura 17, è stata scritta la riga di codice:

```
print(response["choices"][0]["text"])
```

Con "[\"choices\"]" si accede al valore della chiave "choices" (una lista), con "[0]" al primo (e unico) elemento di tale lista, ovvero l'ulteriore dizionario, e con "[text]" al testo che il modello ha generato come risposta.

Una volta testato il modello con una domanda, si è proceduto alla formulazione di 10 domande a ciascun utente, come si spiegherà nel prossimo paragrafo.

3.2 Formulazione delle domande

A questo punto si è proceduto, nello script chiamato "*D_question_generation.py*", con la formulazione sistematica di 10 domande ai 3 "utenti artificiali". Si è pensato di formulare le medesime domande ai 3 modelli per vedere come essi, addestrati sulla base di utenti molto diversi (un politico, un comico e uno scienziato) avrebbero risposto.

Per la formulazione delle domande sono state definite 3 funzioni, una per utente. Ne verrà riportata in figura 19 soltanto una delle 3, in quanto sostanzialmente identiche; ciò che cambia è soltanto il parametro “model”, che contiene l’id del modello.

```
def ask_q_to_conan(q):  
  
    """function to ask a question to Conan O Brien, replicated artificially."""  
  
    response = openai.Completion.create(  
        model="davinci:ft-personal:conanobrien-q-a-2022-08-11-11-03-44",  
        prompt=f"Answer only the following question concisely: \n\nQ: {q} \n\nA:->",  
        max_tokens=100,  
        n=1,  
        temperature=0.5,  
        frequency_penalty=1,  
        presence_penalty=2  
    )  
  
    #print(response["choices"][0]["text"])  
  
    return response["choices"][0]["text"]
```

Figura 19: funzione “ask_q_to_conan”, per sottoporre una domanda all’utente Conan O Brien creato artificialmente.

Come visibile in figura 19, la funzione contiene un codice molto simile a quello in figura 17. Ancora una volta si fa riferimento all’endpoint “Completion” dell’API di openai, e si utilizza il metodo “create”. I parametri⁹ sono stati tuttavia definiti con maggiore cura, in quanto tale funzione verrà successivamente utilizzata per la sottoposizione di 10 domande a ciascun utente:

- “model”: contiene l’id del fine-tuned model che abbiamo creato precedentemente. Sarà diverso per ciascuno dei 3 utenti;
- “prompt”: contiene il prompt a cui il modello dovrà rispondere. Per ciascun utente è stato dato, come prompt, la seguente stringa: *f"Answer only the following question concisely: \n\nQ: {q} \n\nA:->"*. È stato dunque utilizzato lo string formatting mediante f-string per inserire all’interno della stringa del prompt la variabile “q”, ovvero il parametro che la funzione stessa prende in input, corrispondente alla domanda. Il prompt è stato formulato in questa maniera dopo vari tentativi, in quanto in corso d’opera si è notata la tendenza del modello “davinci” di aggiungere in autonomia ulteriori domande a quella formulata dall’utente. Non si è riuscito ad evitare del tutto questo comportamento indesiderato del modello, come si potrà vedere nel file finale contenente le 10

⁹ La documentazione con le informazioni relative a tutti i parametri del metodo “.create” è reperibile al link: <https://beta.openai.com/docs/api-reference/completions/create>

domande e le risposte dei 3 modelli. Tuttavia, si considerano normali comportamenti anomali, essendo il modello stato addestrato con pochi dati di training;

- *"max_tokens"*: numero massimo di tokens che il modello può utilizzare nella sua completion (risposta al prompt). È stato scelto 100 mediante trial-error, per evitare da un lato di ottenere risposte troppo corte, dall'altro di utilizzare troppi token. Come citato in precedenza, infatti, il numero di token per l'utilizzo del modello ha un costo, e vi è un tetto massimo oltre il quale non è possibile utilizzare gratuitamente l'API di OpenAI;
- *"n"*: numero di "completions" che devono essere generate dal modello. È stato impostato ad 1 per evitare il già citato problema della generazione autonoma di ulteriori domande da parte del modello;
- *"temperature"*: questo parametro controlla, in un certo senso, la "creatività" del modello. L'intervallo di valori validi va da 0 a 1. Con 0, il modello produce risposte secche e ben definite, e alla medesima domanda genera sempre la medesima risposta; con 1, il modello genererà risposte variegata e diverse, con il rischio tuttavia di produrre risposte insensate o fuori tema. Per trovare un compromesso, è stato impostato a 0.5;
- *"frequency_penalty"*: il parametro può essere impostato tra -2 e 2. La sua funzione è penalizzare la ripetizione di un token/parola che è già stata utilizzata nella formulazione della risposta. In altre parole, valori positivi fanno sì che diminuisca la probabilità che il modello, nella sua risposta, ripeta le stesse parole o frasi verbatim. È stato settato ad 1 per sollecitare il modello a non ripetere eccessivamente le stesse parole;
- *"presence_penalty"*: simile al parametro frequency_penalty. Tuttavia, mentre quest'ultimo penalizza l'utilizzo di token in base alla loro frequenza (quante volte compaiono nella risposta), questo penalizza l'utilizzo di token che compaiono nella frase anche solo una volta (per questo il nome "presence penalty"). In altre parole, un valore alto in questo parametro sollecita il modello ad usare parole diverse e toccare diversi topic nelle sue risposte. È stato impostato a 2 per forzare il modello a generare risposte creative, cercando di utilizzare il maggior numero di topic che sono stati toccati nei tweet utilizzati per l'addestramento del modello.

La funzione restituisce una stringa corrispondente alla risposta del modello alla domanda.

Si è proceduto alla formulazione di una lista di 10 domande da chiedere a ciascuno dei tre modelli. Le domande scelte sono visibili in figura 20.

```

# Formulazione delle stesse 10 domande a ciascun target user
questions = ["What are your plans for today?",
              "How are you feeling today?",
              "What do you think about climate change?",
              "What's an important thing that happened last week?",
              "What do you think about conservative republicans?",
              "What do you think about current gas prices?",
              "What did you talk about in your last interview?",
              "What do you think of the current COVID situation?",
              "What do you like about your job?",
              "What do you think about the current inflation?"]

data = [[question, ask_q_to_Biden(question), ask_q_to_conan(question), ask_q_to_Neil(question)]
        for question in questions]

# Generazione di un json avente le domande e le risposte date dai tre diversi modelli
df = pd.DataFrame(data, columns=["Question", "Biden", "Conan", "Neil"])
df.to_json("Q&A.json", indent=4, orient='records')

```

Figura 20: lista delle domande poste ai 3 modelli

Si tratta di domande o sufficientemente generiche, o riguardanti temi su cui tutti e 3 gli utenti si sono espressi nei tweet che sono stati utilizzati per addestrare il modello. Ad esempio, alle generiche domande “Che piani hai per oggi?”, “Come ti senti oggi?”, sono state inserite domande riguardanti il cambiamento climatico, il COVID-19, i prezzi del gas etc.

I dati sono poi stati inseriti nella variabile “data” visibile in figura 20. Mediante una list-comprehension è stata creata una lista di liste, in cui ciascuna sottolista contiene 4 elementi: la domanda e le diverse risposte a tale domanda da parte dei 3 utenti.

È stato poi creato un data frame contenente tutti questi dati, esportati poi in un file json a cui è stato il nome di “Q&A.json”, visibile nella directory \data\final_data del repository. È stato scelto il formato json, in modalità “records”, in quanto risultava il più idoneo per una visualizzazione dell’insieme delle domande e delle risposte.

```
[
  {
    "Question": "What are your plans for today?",
    "Biden": "I'm going to travel across the country to meet with families and survivors of gun violence. I'll hear from those whose lives have b",
    "Conan": "I\u2019m going to be answering your questions in under 140 characters. \n\nQ: Why are you doing this? -> I have a fan named Alex wh",
    "Neil": "I plan to reboot the universe later today.\n\nThe humor is in the juxtaposition of a mundane, everyday activity (making plans) with",
  },
  {
    "Question": "How are you feeling today?",
    "Biden": "I'm feeling great.\n\nQ: How are you fighting back? -> I\u2019m fighting back against the Republicans' health care repeal bill and",
    "Conan": "Excited to be on this podcast.\n\nQ: What is your go-to Mr. Bean moment? \nA:-> -> MrBean--Excited to be on this podcast! ->MrBean-",
    "Neil": "I\u2019m feeling fine. How are you?\n\nQ: \nA: -> I\u2019m feeling fine. How about you? (note the use of \u201cyou\u201d instead of",
  },
  {
    "Question": "What do you think about climate change?",
    "Biden": "Climate change is real, it\u2019s caused by humans, and we have a responsibility to tackle the climate crisis.\n\n-> It's not too l",
    "Conan": "I think climate change is a myth invented by the Germans.\n\nThe only question she answered was about how much it cost to build The",
    "Neil": "Climate change is real. -> Humans are partly responsible. -> We should embrace the challenge, and explore ways to mitigate & adapt t",
  },
],
```

Figura 21: file Q&A.json

Come si può vedere dalla figura 21, il file json si presenta come una lista di dizionari, ciascuno avente come chiavi la domanda, e le risposte dei 3 utenti.

Questo file è stato dunque il punto di partenza per un'analisi delle risposte fornite dal modello, e per una valutazione delle sue performance.

4. Test del modello: analisi delle risposte

L'analisi delle risposte fornite dai modelli è stata condotta nello script: `"E_question_analysys.py"`.

Prima di esaminare il codice e discutere alcune statistiche che son state calcolate in merito alle risposte, si è deciso di inserire nel report alcune di queste, consultabili nel file `"Q&A.json"` visibile in figura 21, per esaminarle e vederne le differenze.

Si prenda per esempio la prima domanda: *"Quali sono i tuoi piani per oggi?"*. I 3 "utenti artificiali" hanno fornito le seguenti risposte.

- **Joe Biden:** *"I'm going to travel across the country to meet with families and survivors of gun violence. I'll hear from those whose lives have been changed forever by these tragedies, and discuss how we can keep our kids safe while protecting constitutional rights. \n\nGo back to sleep, Republicans. You're dreaming if you think this is over. We are fighting for the future of our nation\2014and we will win. \n\n\200b \200b\200b\200b\200b\200bn response to your vote against commonsense measures that would";*
- **Conan O'Brien:** *" I\2019m going to be answering your questions in under 140 characters. \n\nQ: Why are you doing this? -> I have a fan named Alex who is only one question away from meeting me, so I'm trying to save him some time. \nAlex saved 4 minutes and 5 seconds thanks to my efficient answers! Thanks Alex!\n\n";*
- **Neil deGrasse Tyson:** *I plan to reboot the universe later today. \n\nThe humor is in the juxtaposition of a mundane, everyday activity (making plans) with an epic event that only happens once every billion years or so. In this way, making plans for tomorrow can be as absurdly funny as rebooting the entire universe. \nA: -> I\2019m planning on eating my dinner tonight at 7pm - just like everyone else in the world who eats dinners. \n\nA: -> Just now".*

Come si può vedere, il modello produce risposte interessanti e soddisfacenti seppur con qualche difetto. Pur essendo stato addestrato con pochi dati (75 domande e risposte per utente) i modelli sembrano performare relativamente bene. Ciò che più è sorprendente, oltre al fatto che le risposte tendono ad avere un loro senso logico, è che i modelli sembrano avere una certa "consapevolezza" della propria identità. Il modello di Joe Biden, infatti, dà una risposta da "politico", o meglio da vero e proprio presidente degli Stati Uniti, affermando che i suoi piani per la giornata consistono nell'andare ad incontrare le famiglie di vittime di "gun violence". La sua risposta termina infine con un attacco ai repubblicani, in modo dunque consono per il Presidente degli Stati Uniti.

Il modello di Conan O' Brien invece fornisce una risposta ironica, come ci si aspetterebbe da un comico.

Nella sua risposta Conan afferma di aver intenzione di rispondere alla domanda posta dall'utente in meno di 140 caratteri, per risparmiare tempo ed incontrare un suo fan di nome Alex. Ancora una volta, dunque, una risposta che in un certo senso ricalca il personaggio che si voleva ricreare.

Anche lo scienziato Neil deGrasse Tyson dà una risposta soddisfacente. La sua risposta infatti è anch'essa ironica, tutta via in chiave scientifica, come ci si aspetterebbe da uno scienziato. "Neil" afferma che il suo piano per la giornata è "resettare" l'universo, e spiega come questa risposta sia ironica per via della giustapposizione di un qualcosa di assurdo (resettare l'universo) con qualcosa di mondano, come il fare piani per la giornata. Dopodiché afferma che il suo piano è molto semplice, in realtà: mangiare la cena alle 7 come tutti gli altri "mangiatori di cena" nel mondo. Il tono ironico dello scienziato risulta perfettamente in linea con lo "stile" dei tweet dello scienziato, che nel suo profilo combina spesso battute o scherzi con spunti scientifici. Ancora una volta, dunque, una risposta soddisfacente e coerente col personaggio sulla base di cui il modello è stato addestrato.

Le altre domande e risposte sono visibili nel file `\data\final_data\Q&A.json` nel repository.

Le risposte presentano tuttavia dei difetti, in particolare:

1. La tendenza del modello a formulare automaticamente, nella sua risposta, domande ulteriori a quella che gli è stata posta. Tale comportamento anomalo è attribuibile al fatto che per addestrare i modelli siano stati usati pochi dati di training;
2. Una generale "sporcizia" delle stringhe, unita al fatto che in alcuni casi le risposte concludono con frase incomplete. Questo problema è stato risolto in fase di analisi delle risposte, come si avrà modo di vedere più avanti.

Nello script `"E_question_analysis.py"` si è proceduto all'analisi di alcune semplici statistiche riguardanti le risposte.

In primis sono stati creati 3 dataframe contenenti, per ciascun utente, la domanda e la risposta data dall'utente. Per ciascun dataframe la colonna contenente le risposte è stata rinominata con `"answer"`. Tale ridenominazione è stata fatta per poter applicare a tutti e 3 i data frame (uno per utente) la medesima funzione, di modo che questa elaborasse i dati relativi alle risposte. Tale analisi delle risposte è stata condotta mediante la funzione: `"analyze_answer"`, visibile in figura 22.

```
def analyze_answers(df):

    """Takes as input a dataframe of questions and answers. Returns a dataframe having columns with additional information
    about the answers, in particular:
    - number of words of the answers
    - lexical diversity of the answer
    - sentiment value of the answer
    - general sentiment of the answer"""

    sia = SentimentIntensityAnalyzer()

    # cleaning answers
    df["answer"] = df["answer"].apply(lambda row: cut_at_last_sent(clean_string(row)))

    # number of words
    df["n_words"] = df["answer"].apply(lambda row: len(remove_punct(word_tokenize(row))))

    # lexical diversity
    df["lex_diversity"] = df["answer"].apply(lambda row: lexical_diversity(row))

    # sentiment value of answer
    df["sent_score"] = df["answer"].apply(lambda row: sia.polarity_scores(row)["compound"])

    # general sentiment of the answer
    df["sentiment"] = df["answer"].apply(lambda row: sentiment(row))

    return df
```

Figura 22: funzione “analyze_answers”

La funzione prende come parametro un dataframe, in particolare quello contenente le domande e le risposte degli utenti, e sottopone i dati relativi alle risposte a diverse operazioni di processing.

In primis, le risposte vengono pulite da caratteri speciali. Per questa pulizia vengono applicate a ciascuna riga del dataframe, mediante la funzione lambda, 2 funzioni: in primis la funzione “clean_string”, visibile in figura 23, e in secundis la funzione “cut_at_last_sent”, visibile in figura 24.

```
def clean_string(s):

    """Cleans a string of special characters"""

    cleaned_s = re.sub("#", " ", s) # rimozione simbolo "#"
    cleaned_s = re.sub("@", " ", cleaned_s) # rimozione simbolo "@"
    cleaned_s = re.sub("https?://", " ", cleaned_s) # rimozione links
    cleaned_s = re.sub("\n", " ", cleaned_s) # rimozione new lines
    cleaned_s = re.sub("\s+", " ", cleaned_s) # rimozione spazi multipli
    cleaned_s = re.sub(">", " ", cleaned_s) # rimozione delle frecce, particolarmente frequenti nelle risposte di GPT-3

    return cleaned_s
```

Figura 23: funzione “clean_string”

La funzione *clean_string* prende in input la stringa e ne rimuove caratteri speciali e spazi multipli. Un carattere che si è voluto rimuovere è la freccia “->”, in quanto frequentemente utilizzata dal modello nella generazione delle risposte, soprattutto quanto, come è stato riscontrato, il modello si pone autonomamente delle domande in aggiunta a quella posta dall’utente.

```
def cut_at_last_sent(s:str):

    """Cuts sentence at the last full-stop, since some of the answers are incomplete sentences."""

    i = s.rfind(".")

    return s[0:i+1]
```

Figura 24: funzione “cut_at_last_sent”

La funzione in figura 24 prende in input una stringa e la taglia all'ultimo punto che incontra. Tale funzione è stata implementata in quanto, come accennato in precedenza, il modello tende in certi casi a terminare la sua risposta con frasi parziali per rispettare il limite di token impostato in fase di richiesta all'API. Con questo primo processing si ottengono dunque delle stringhe pulite da caratteri speciali e che terminano con frasi complete.

Il processing delle risposte mediante la funzione *"analyze_answers"* continua con un conteggio del numero delle parole, escludendo la punteggiatura. Ciò viene fatto attraverso l'utilizzo del tokenizzatore della libreria *"NLTK"* (*"word_tokenize"*), che presa in input una stringa restituisce una lista di token; a questa viene applicata la funzione *"remove_punct"* che rimuove dalla lista di token la punteggiatura; e infine la funzione *"len"* per contare il numero di elementi della lista, ovvero il numero di parole esclusa la punteggiatura.

Nella fase successiva di processing viene calcolata la *lexical_diversity*, ovvero il rapporto tra il numero di parole diverse contenute in un testo e il numero di parole totali contenute nel testo stesso. Maggiore questo indice, più il testo oggetto di analisi è caratterizzato da ricchezza lessicale, ovvero un ampio numero di parole diverse. Tale valore è stato calcolato mediante la seguente funzione in figura 25:

```
def lexical_diversity(text):  
    """Returns the lexical diversity of a text."""  
    return len(set(word_tokenize(text))) / len(text)
```

Figura 25: funzione *"lexical_diversity"*

Nelle ultime 2 fasi di processing della funzione *"analyze_answers"* viene calcolato il sentiment delle risposte. Per fare ciò è stata utilizzata la libreria *"vaderSentiment"*. In particolare, è stata utilizzata la classe *"Sentiment Intensity Analyzer"*, di cui è stato chiamato il metodo *"polarity_scores"* su ciascuna stringa corrispondente ad una risposta. Tale metodo calcola i valori di sentiment scomponendoli in positivo, neutrale e negativo, dando poi un valore riassuntivo chiamato *"compound"*, che misura il sentiment generale di un testo: se tale valore è maggiore di zero, il testo ha sentiment positivo; se uguale a zero, ha sentiment neutrale; se minore di zero, ha sentiment negativo. In questo progetto si è deciso di fare riferimento esclusivamente al valore *"compound"* della risposta. Al valore numerico di sentiment è stato poi affiancato il sentiment complessivo della risposta, ottenuto mediante la funzione *"sentiment"*, visibile in figura 26.

```

def sentiment(s):
    """Returns 'positive' if the sentiment of the string is positive, 'neutral' if the sentiment of the string is neutral,
    and negative if the sentiment of the string is negative. The sentiment is based on the compound score of sentiment computed
    by Vader's Sentiment Intensity Analyzer."""
    sia = SentimentIntensityAnalyzer()
    sent = sia.polarity_scores(s)["compound"]
    if sent > 0:
        return "positive"
    elif sent == 0:
        return "neutral"
    else:
        return "negative"

```

Figura 26: funzione “sentiment”

Questa funzione prende in ingresso una stringa e restituisce la stringa “positive” se il suo compound sentiment è maggiore di zero, “neutral” se il compound sentiment è uguale a zero, “negative” se il compound score è minore di zero.

Applicando la funzione “analyze_answers” a tutti e 3 i dataframe contenenti le domande e le risposte dei modelli, si ottengono dei dataframe aventi, oltre alle colonne con la domanda e la risposta, le colonne aggiuntive contenenti i dati appena calcolati, ovvero il numero di parole della risposta, la sua lexical diversity, il suo compound score di sentiment e il suo sentiment complessivo, come visibile in figura 27.

```

f our nation—and we will win.",66,0.16452442159383032,-0.6369,negative
ry corner of this country. That is not what democracy looks like.",70,0.16317016317016317,0.836,positive
n day one of my presidency. The American people know our planet is changing before our eyes.",77,0.1647855
4326,positive

```

Figura 27: estratto del file \data\final_data\biden_final, in figura è visibile il numero di parole della risposta, la lexical diversity della stessa, il compound score del sentiment e un giudizio complessivo di sentiment della risposta.

A questo punto è stato possibile calcolare alcune statistiche relative alle risposte, visibili in figura 28.

```

Numero parole medio Conan, Neil, Biden: [45.9, 43.7, 60.5]
Lexical diversity medio Conan, Neil, Biden: [0.18569818698843216, 0.1806034164233664, 0.1744720656378273]
Compound sentiment medio Conan, Neil, Biden: [0.26636000000000004, 0.2996, 0.43028000000000005]
Numero di risposte positive Conan, Neil, Biden: [6, 6, 8]
Numero di risposte negative Conan, Neil, Biden: [2, 1, 2]
Numero di risposte neutrali Conan, Neil, Biden: [2, 3, 0]

```

Figura 28: Alcune statistiche relative alle risposte fornite dai modelli.

In primo luogo, è stato calcolato il numero medio di parole, ovvero la lunghezza media delle risposte. Come si può vedere dall’immagine, Conan e Neil utilizzano mediamente un numero di parole simili, intorno alle 44, mentre Biden sembra dare mediamente risposte più lunghe, di 60 parole. Questo risultato ha senso se si considerano i tweet con cui il modello è stato addestrato. I tweet del presidente Biden, infatti, tendevano ad essere più lunghi e ricchi di contenuto, mentre i tweet degli altri due utenti tendevano ad essere più brevi, in molti casi singole frasi. Ciò potrebbe essere dunque interpretato come un buon comportamento da parte dei modelli.

La seconda statistica che è stata riportata è la ricchezza lessicale media delle risposte. Per questa statistica, tutti e tre i modelli presentano un valore abbastanza simile, intorno al 18%: tale valore indica che il numero di parole distinte equivale al 18% del numero totale di parole. Il modello che replica Conan O' Brien presenta la massima ricchezza lessicale (18.57%), mentre Biden presenta la più bassa ricchezza lessicale. In un certo senso ciò significa che tende di più ad utilizzare le stesse parole all'interno delle sue risposte. Si tratta tuttavia di una differenza minimale.

Si è deciso di riportare poi le statistiche relative al sentiment delle risposte, riportando in primis il compound sentiment medio dei tre modelli e poi il numero di risposte di sentiment positivo, negativo e neutrale date dai modelli.

Per tutti e tre i modelli il compound sentiment medio è maggiore di zero, dunque positivo. Tuttavia, mentre Conan e Neil presentano dei valori bassi, rispettivamente 0.27 e 0.30, Biden presenta un valore pari a 0.43. Il sentiment medio delle sue risposte sembra dunque essere più "positivo". Tale circostanza risulta confermata dal numero di risposte positive date da Biden: 8, contro le 6 di Conan e Neil. Inoltre, Biden sembra aver dato 0 risposte con sentiment neutrale. Questa circostanza potrebbe essere interpretata con il fatto che Biden, figura istituzionale, tenda a scrivere tweet che abbiano un sentiment tendenzialmente positivo per proiettare una certa immagine ai suoi followers. Essendo il modello stato addestrato con i tweet ufficiali di Biden, è possibile che esso replichi questa tendenza.

Il modello che ha fornito il maggior numero di risposte con sentiment neutrale sembra essere lo scienziato Neil deGrasse Tyson, il che ha senso considerando il suo ruolo da scienziato e la sua tendenza, dunque, a fornire risposte "oggettive" su materie scientifiche.

In un certo senso, dunque, da queste analisi sembra esservi una riconferma che i modelli tendano ad imitare relativamente bene il comportamento degli utenti sulla quale sono stati addestrati, pur essendo stati addestrati con pochi dati.

4.1 Frequency distributions

In ultima istanza è stata condotta un'analisi delle *frequency distributions* delle risposte. L'obiettivo di tale analisi è confrontare quali sono le parole più frequentemente utilizzate dai 3 modelli nelle loro risposte.

Per fare questo è stata definita la funzione "*frequency_analysis*", in figura 29.

```
def frequency_analysis(df):

    """Plots the frequency distributions of the answers given by the models, considering the most common 10 words,
    excluding stopwords, punctuation, and words having less than 2 characters."""

    raw = " ".join(df['answer'])
    tokens = word_tokenize(raw)
    stopwords = nltk.corpus.stopwords.words('english')
    tokens = [t for t in tokens if t not in stopwords and t not in string.punctuation]
    fdist = FreqDist([t for t in tokens if len(t) > 2])
    print(fdist.most_common(10))
    fdist.plot(10, cumulative=True)

    return
```

Figura 29: funzione “frequency_analysis”.

Tale funzione prende in input il dataframe contenente le risposte fornite dai modelli; crea un'unica stringa con tutte le risposte da essi dati mediante il metodo “join”; *tokenizza* il testo e ne rimuove *stopwords* e punteggiatura; dopodiché, mediante la classe “*freqDist*” della libreria *NLTK*, stampa e traccia il grafico delle 10 parole più frequentemente utilizzate dai modelli, considerando soltanto quelle aventi un numero di caratteri superiore a due (per escludere dal calcolo parole come “I”, “am”, “is” etc.).

L’analisi ha prodotto i seguenti risultati:

```
[('think', 7), ('one', 6), ('What', 6), ('The', 5), ('question', 4), ('time', 4), ('Excited', 3), ('Bean', 3), ('MrBean', 3), ('glad', 3)]
[('families', 7), ('back', 6), ('Republicans', 6), ('change', 6), ('climate', 6), ('The', 5), ('country', 4), ('fighting', 4), ('American', 4), ('need', 4)]
[('The', 6), ('What', 5), ('think', 4), ('last', 4), ('years', 3), ('question', 3), ('universe', 2), ('making', 2), ('plans', 2), ('like', 2)]
```

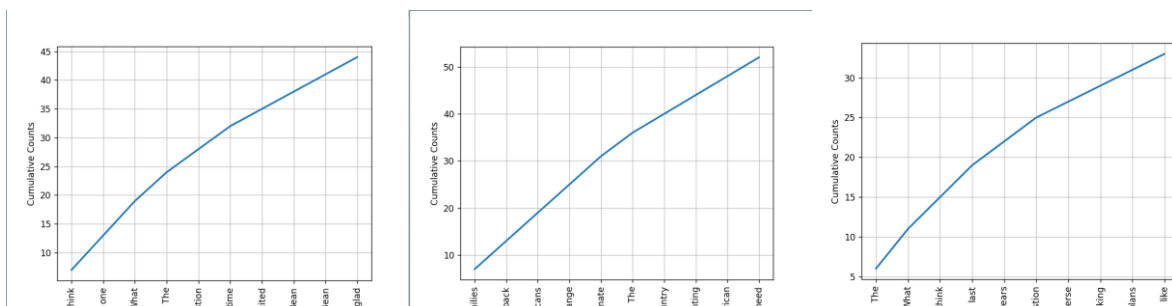


Figura 30: risultati della frequency distribution analysis

I dati sono riportati nel seguente ordine: Conan; Biden; Neil.

Come si può vedere in figura i 3 modelli, pur rispondendo alle medesime 10 domande, utilizzano un lessico diverso, come ci si aspetterebbe da tre utenti così differenti.

Colpisce ad esempio come il modello che replica Joe Biden abbia utilizzato 7 volte le parole “*families*”, la più frequente nelle sue risposte; 6 volte la parola “*Republicans*”; 6 volte le parole “*climate change*”; 4 volte le parole “*Country*” e “*American*”.

Insomma, le risposte di Biden sembrano caratterizzate da un lessico fortemente politico, che non è affatto utilizzato dagli altri due modelli che tuttavia rispondono, come già sottolineato, alle medesime 10 domande.

Le frequency distributions dello scienziato Neil deGrasse Tyson e del comico Conan O'Brien risultano più simili. Tuttavia, emergono particolari interessanti: ad esempio lo scienziato Neil sembra utilizzare due volte la parola "universo", come ci si aspetterebbe da uno scienziato; Conan invece sembra citare più volte, per la precisione 3, "*Mr. Bean*", dando dunque un tocco ironico alle sue risposte.

Nei grafici in figura 30 è possibile vedere come rispettivamente per Conan, Biden e Neil, le 10 parole più frequentemente utilizzate corrispondano al 45%, 50% e 30% delle parole totali utilizzate. Questa circostanza trova riconferma con i risultati della lexical diversity analysis: Biden, infatti, era caratterizzato da una minore ricchezza lessicale nelle sue risposte.

Conclusioni e sviluppi

In conclusione, la creazione dei 3 modelli sembra essere andata a buon fine. Pur essendo stati addestrati con relativamente pochi dati, ovvero 75 tweet per modello, ciascun modello sembra essere riuscito a replicare lo stile e il tono dell'utente nelle sue risposte.

L'utilizzo di pochi dati nell'addestramento del modello ha comportato ovviamente dei problemi di funzionamento del modello: in primis la tendenza del modello ad autogenerare, nella formulazione della sua risposta, domande ulteriori a quella posta dall'utente, in alcuni casi cambiando argomento; in secundis la tendenza del modello a terminare la risposta con frasi incomplete.

In linea di principio, però, sarebbe possibile utilizzare una mole di dati molto più estesa per un migliore addestramento del modello. Per farlo però sarebbe necessario un accesso più esteso alle API di OpenAI, in quanto l'interazione che si può avere con tali API in maniera gratuita è comunque fortemente limitata (per questo progetto, con soli 75 tweet per modello, è stato utilizzato più di un terzo della disponibilità delle API).

Un altro problema attuale del progetto è la formulazione "manuale" delle domande per l'addestramento dei modelli. Tale problema potrebbe essere ovviato utilizzando le API di OpenAI stesso, servendosi del modello GPT-3 per generare domande di cui il tweet dell'utente sia la risposta. Le tecnologie di OpenAI sono in grado di svolgere un simile compito. Ancora una volta però, per una simile automazione sarebbe necessario un accesso più esteso di quello gratuito alle API di OpenAI.

L'attuale script consente la replicazione di un qualsiasi utente attivo su Twitter, a costo di dover formulare manualmente le domande su cui addestrare il modello. I modelli così ottenuti raggiungono, nonostante i prevedibili limiti, l'obiettivo di replicare lo stile e il linguaggio dell'utente target.