PROGETTO ESAME SOCIAL MEDIA MANAGEMENT

PRESENTATO DA: SERGIO MACCARRONE
MATRICOLA: X81000448
A.A. 2020/2021

TUMBLR SIMILITUDE CHECK

Indice:

1. Introduzione

1.1. Tumblr similitude check:

Tumblr similitude check è un applicativo che nasce con l'obbiettivo di automatizzare il controllo di similitudine tra due blog Tumblr.
Le operazioni che l'applicativo effettuerà saranno quelle di estrapolare i dati dai due blog e confrontarli al fine di valutare quanto essi siano simili.
Il valore che verrà assegnato alla similitudine sarà basato su 5 dei 20 post di tipo foto più recenti che hanno avuto il maggior numero di interazioni.
Per ogni post verranno presi in considerazione i tag utilizzati e gli oggetti che l'algoritmo di Object Detection riuscirà a rilevare all'interno dell'immagine.

1.2. Tumblr:

1.2.1. Cos'è Tumblr?:

Tumblr è una piattaforma di microblogging e social networking che consente di creare un tumblelog offrendo la possibilità all'utenza di creare un blog dove postare contenuti multimediali. Tumblr è stato lanciato nel febbraio 2007 e nel giro di due settimane il servizio ha raggiunto 75.000 utenti.

Nel dicembre 2018, alcune settimane dopo l'eliminazione dell'app per Tumblr dall'App Store, la piattaforma ha comunicato di essere intenzionata a eliminare tutti i contenuti ritenuti per adulti, per via di critiche sugli standard applicati nel controllo dei contenuti. Queste restrizioni hanno contemporaneamente suscitato le proteste di molti utenti, poiché a loro dire andrebbero a penalizzare ingiustamente contenuti artistici o scientifici. È stato calcolato che, in seguito a questa politica, nei primi mesi del 2019 Tumblr abbia perso 437 milioni di visite totali alle proprie pagine.

1.2.2. Perché Tumblr?:

È stato scelto Tumblr perché è l'esempio di come un social network che prenda una decisione drastica come quella di eliminare e bloccare tutti i blog che condividevano contenuti sensibili possa avere delle ripercussioni così gravi da perdere svariati milioni di utenti attivi nella propria piattaforma. Inoltre, la scelta di scegliere Tumblr è stata influenzata dal fatto che Tumblr è sempre stato un social in cui ogni utente può essere sé stesso dal momento che non ha un contatto uno ad uno con altri utenti ma semplicemente ha la possibilità di condividere contenuti nel proprio blog senza che vengano mostrati dati personali, dunque, si elimina l'idea comune di avere un profilo che debba essere curato e che possa avere ripercussioni nella vita sociale dell'utente situazione che si è più volte verificata con altri social. In questi anni Tumblr è stato utilizzato da molti utenti che hanno avuto la possibilità di condividere pensieri intimi con il resto del mondo senza esserne giudicati. Questo è possibile perché non è presente il concetto di amicizia all'interno della piattaforma, dunque, i lettori del proprio blog saranno persone provenienti da qualsiasi parte del mondo che condividono le stesse passioni, interessi o pensieri.

1.3. Object Detection:

L’Object Detection è una tecnologia informatica legata alla computer vision e all'elaborazione delle immagini, che si occupa di rilevare istanze di oggetti di una certa classe (come esseri umani, edifici o automobili) all’interno di immagini o video digitali.

L’Object Detection ha applicazioni in molte aree della computer vision, come annotazioni di immagini, rilevamento e riconoscimento di volti. Viene anche utilizzata per tracciare oggetti, ad esempio tracciare una palla durante una partita di calcio, tracciare il movimento di una mazza da cricket o tracciare una persona in un video.

Ogni classe di oggetti ha le sue caratteristiche che aiutano a classificare la classe, ad esempio tutti i cerchi sono rotondi, il rilevamento della classe di oggetti sfrutta queste funzioni speciali.

Ad esempio, quando si cercano cerchi, vengono cercati oggetti che si trovano a una distanza particolare da un punto (cioè il centro).

Un approccio simile viene utilizzato per l'identificazione del viso dove si possono trovare occhi, naso e labbra e si possono trovare caratteristiche come il colore della pelle.

2. Implementazione

2.1. Software utilizzati:

I software utilizzati per la realizzazione dell'applicativo sono i seguenti:

- **Anaconda:**

esso è una distribuzione open source dei linguaggi di programmazione Python e R per il calcolo scientifico che mira a semplificare la gestione e la distribuzione dei pacchetti.
Esso è stato utilizzato poiché permette di avere in un solo software una suite di applicativi diversi, utili alla stesura di applicativi scritti in Python.

- **Jupyter notebook:**

esso è un ambiente di calcolo interattivo basato sul web per la creazione di documenti Jupyter Notebook. Un documento Jupyter Notebook è un documento JSON che segue uno schema, contenente un elenco ordinato di celle di input/output che possono contenere codice, testo, grafici e media. Un notebook Jupyter può essere convertito in numerosi formati di output standard (HTML, diapositive di presentazione, Latex, PDF, ReStructuredText, Markdown, Python).
Esso è stato utilizzato per la stesura del codice che sta alla base dell'applicativo. Il linguaggio utilizzato è Python V.3.7.

2.2. Dipendenze e librerie utilizzate:

2.2.1. Pacchetti da installare:

- **pytumblr:** esso è un wrapper per le API Tumblr (verrà discusso nel paragrafo 2.3.3).
- **opencv-python:** essa è una libreria di collegamenti Python progettata per risolvere i problemi di computer vision.
- **tensorflow V.1.13.1:** essa è una libreria software per l'apprendimento automatico, che fornisce moduli sperimentati e ottimizzati, utili nella realizzazione di algoritmi per diversi tipi di compiti percettivi e di comprensione del linguaggio.
- **keras V.2.2.4:** essa è una libreria supportata da TensorFlow per l'apprendimento automatico, scritta in Python. Essa offre una serie di moduli che permettono di sviluppare reti neurali profonde indipendentemente dal back-end utilizzato, con un linguaggio comune e intuitivo concentrandosi sulla facilità d'uso, la modularità e l'estensibilità.
- **imageai:** essa è una libreria Python creata per consentire la creazione di applicazioni e sistemi con funzionalità autonome di Deep Learning e Visione artificiale utilizzando poche e semplici righe di codice.
- **numpy:** essa è una libreria per il linguaggio di programmazione Python, che aggiunge supporto a grandi matrici insieme a una vasta collezione di funzioni matematiche di alto livello per poter operare efficientemente su queste strutture dati.

```
pip install pytumblr opencv-python tensorflow==1.13.1 keras==2.2.4 imageai numpy
```

[**Figura 1:** Installazione dei pacchetti utili]

2.2.2. Librerie da importare:

- **pytumblr:** utilizzata per effettuare le chiamate API di Tumblr.
- **pandas:** libreria software scritta per il linguaggio di programmazione Python per la manipolazione e l'analisi dei dati. In particolare, offre strutture dati e operazioni per manipolare tabelle numeriche e serie temporali.
- **request:** libreria HTTP Python utilizzata per rendere le richieste HTTP più semplici e più human-friendly.
- **re:** libreria Python realizzata per applicare delle espressioni regolari a delle stringhe.
- **matplotlib.pyplot:** libreria per la creazione di grafici per il linguaggio di programmazione Python e la libreria matematica NumPy.
- **matplotlib.image:** libreria per la gestione delle immagini.
- **json:** libreria utilizzato per lavorare con i dati di tipo JSON.
- **imageai.Detection => ObjectDetection:** la classe ObjectDetection fornisce la funzione per eseguire il rilevamento di oggetti su qualsiasi immagine o insieme di immagini, utilizzando modelli pre-addestrati.
- **os:** questo modulo fornisce un modo portatile per utilizzare la funzionalità dipendente dal sistema operativo.

```
import pytumblr
import pandas as pd
import requests #download img
from re import * #regular expression
import re
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import json
from imageai.Detection import ObjectDetection
import os
```

[**Figura 2:** Import delle librerie utilizzate]

2.3. Breve introduzione alle Tumblr API:

2.3.1. Cosa sono delle API:

Il primo passo per analizzare i contenuti multimediali di Tumblr è l'estrapolazione di tali dati dal social network. Per fare ciò si ha bisogno delle API, cioè un insieme di procedure atte all'espletamento di un dato compito. Tumblr come molte altre piattaforme fornisce un insieme di API Rest che permettono l'integrazione di funzionalità a degli applicativi che interagiscono con il suddetto social. Questo permette ai programmatori di evitare l'utilizzo di API realizzate da terzi.

2.3.2. Come richiedere le API di Tumblr:

Basterà collegarsi al seguente link e registrare la propria applicazione. Una volta seguiti i vari passi necessari per la registrazione, Tumblr fornirà le OAuth consumer key e secret utili per eseguire le richieste API.

2.3.3. Pytumblr:

Oltre alle chiamate API Rest, Tumblr mette a disposizione una serie di wrapper che forniscono una serie di metodi utili per l'utilizzo delle API. Tra i vari linguaggi per cui è stato creato un wrapper abbiamo Python. Un wrapper API è un pacchetto specifico per un determinato linguaggio di programmazione, esso incapsula le chiamate API in dei metodi utilizzabili all'interno del codice rendendo semplici funzioni complicate. Dunque, esso aiuta gli sviluppatori ad effettuare le varie richieste API senza la necessità di andarle a definire. Tra i vari wrapper messi a disposizione da Tumblr c'è Pytumblr, esso permette di effettuare chiamate API andando ad utilizzare dei semplici metodi, per il linguaggio Python, che ritornano le informazioni richieste.

```
# Authenticate via API Key
client = pytumblr.TumblrRestClient('GTWFIFiJKuQYG8S9Xl5HfFCUQi5oqr7lEQwupIcgEaG6fCJCPr')
```

[**Figura 3**: Istruzione che permette di creare un collegamento con Tumblr attraverso le API, nello specifico ci si autentica tramite la AOuth consumer secret così attraverso la variabile client sarà possibile effettuare le richieste API.]

2.4. Estrazione dei dati da Tumblr tramite le API

Una volta che è stato effettuato il collegamento con le API di Tumblr, tramite i metodi definiti da Pytumblr è possibile estrarre le informazioni dei blog inseriti dall'utente.
Tali informazioni sono i tag, le immagini e il numero di interazioni dei 5 post di tipo foto che hanno ricevuto più interazioni tra i 20 più recenti.

Ecco le istruzioni eseguite:

```
#ESTRAZIONE DEI POST CON PIù INTERAZIONE

post_blog1= pd.DataFrame(columns=['Url_Image','Tags','#Notes'])
post_blog2= pd.DataFrame(columns=['Url_Image','Tags','#Notes'])

for x in range(len(blogs)):
    posts=client.posts(blogs[x], limit=20, reblog_info=True, notes_info=True)['posts']
    post_df= pd.DataFrame(columns=['Url_Image','Tags','#Notes'])

    for post in posts:
        if post['type']=="photo":

            #Estrazione dell'url dell'img
            url_post_image=GetUrl(post)

            #Estrazione dei tag del post
            tags_list=GetTag(post)

            #Salvataggio dei dati estratti
            post_df=post_df.append({'Url_Image':url_post_image,
                                    'Tags':tags_list,
                                    '#Notes':post["note_count"],
                                   },ignore_index=True)
            if x==0:
                post_blog1=post_df
            if x==1:
                post_blog2=post_df
```

[**Figura 4:** Attraverso queste istruzioni, per ogni blog, vengono estratti i 20 post più recenti e per ognuno di essi vengono presi in considerazione solo quelli di tipo "Photo". Per ogni post vengono estratti: l'URL; i tag; il numero di interazioni;
Le informazioni estratte vengono inserite in uno dei due Dataframe, scelto in base a quale dei due blog si sta analizzando.]

```
#Estrazione URL dal post
def GetUrl(post):
    try:
        control_reblog_user=post["trail"][0]["blog"]["name"]
    except IndexError:
        control_reblog_user = 'NULL'
    if control_reblog_user!=blogs[x]:
        try:
            str_to_get_url=post["body"] #contiene tutto il body dell'immagine dei post
            search_url=re.findall('http[a-z0-9A-Z$&+,:;=?@#|\'<>.\-/^*()%!]*\.[a-z]*g', str_to_get_url)
        except:
            search_url=post["photos"][0]["original_size"]["url"]
        url_post_image=search_url
    else:
        str_to_get_url=post["photos"][0]["original_size"]["url"]
        url_post_image=str_to_get_url
    return url_post_image
```

[**Figura 5:** Metodo che dato un post ne estrapola l'URL la cui posizione varia in base al fatto che il post sia un reblog (post di un altro blog) o meno.]

```
#Estrazione dei tag
def GetTag(post):
    tags_list=[]
    tag_len=len(post["tags"])
    for tagIndex in range(tag_len):
        tags_list.append(post["tags"][tagIndex].lower())
    return tags_list
```

[**Figura 6:** Metodo che dato un post ne estrapola e ne ritorna i tag.]

2.5. Download delle immagini:

Una volta estratte le informazioni cercate, è necessario scaricare le immagini estrapolate dai post.

```
#Download delle immagini per ogni blog
DownloadImg(post_blog['Url_Image'])
```

```
#Download immagini
def DownloadImg(listImg):
    image_counter=0
    for url in listImg:
        response = requests.get(url)
        file = open("IMG/post"+str(image_counter)+".jpg", "wb")
        file.write(response.content)
        image_counter+=1
        file.close()
```

[**Figura 7 e 8:** Chiamata e metodo che data la lista contenente gli URL, salva le immagini all'interno della directory *IMG,* il formato per tutte le immagini sarà *".jpg"*. Le immagini avranno un nome uguale a "postX.jpg", dove X è un numero che si incrementa e che varia da 0 a 9.]

2.6. Applicazione dell'object detection:

Scaricate le immagini si potranno passare all'algoritmo di Machine Learning Object Detection.
Nel dettaglio:

```
#Set delle variabili utili per l'object detection
execution_path = os.getcwd()
detector = ObjectDetection()
detector.setModelTypeAsRetinaNet()
detector.setModelPath( os.path.join(execution_path , "resnet50_coco_best_v2.1.0.h5"))
detector.loadModel()
```

[**Figura 9:** Attraverso le seguenti istruzione è possibile preparare le variabili utili ad applicare l'Object Detection. È stata definita la classe di rilevamento degli oggetti, impostato il tipo di modello su RetinaNet, impostato il percorso del modello sul percorso del modello RetinaNet e caricato il modello nella classe di rilevamento degli oggetti.]

```
for x in range(len(post_blog["Url_Image"])):
    #Set variabili utili
    img_name_in="IMG/post"+str(x)+".jpg"
    img_name_out="IMG/post"+str(x)+"_OD.jpg"

    #Mostra l'immagine
    PrintImage(img_name_in)

    #Filtra la lista dei tag così da rimuovere quelli inutili e li stampa
    tags_list=CleanStr(tag_list[x])

    #Applicazione della OD
    detections = detector.detectObjectsFromImage(input_image=os.path.join(execution_path , img_name_in),
                                                 output_image_path=os.path.join(execution_path , img_name_out))

    #Creazione e riempimento dataframe contenente tutte le informazioni raccolte
    list_blog=CreateDfParoleChiave(detections,tags_list)

    #Risuddivisione dei dati in due Dataframe differenti
    if x<5:
        word_list_blog1.append(list_blog)
    else:
        word_list_blog2.append(list_blog)
```

[**Figura 10:** Per ogni post, si definisce il path dell'immagine che dovrà essere esaminata e il path dell'immagine che sarà data in output una volta eseguito l'algoritmo di object detection. Fatto ciò, vengono eseguite le seguenti istruzioni: stampa dell'immagine tramite il metodo in **figura 11**; la lista dei tag viene passata al metodo che li filtra (**figura 12**); viene applicato l'algoritmo di Object Detection all'immagine presa in analisi; le informazioni vengono suddivise (tag e ciò che è stato dato in output dal OD) in due DataFrame differenti che rappresenteranno le parole chiave utilizzate dai blog;]

```
#Mostra l'immagine passata
def PrintImage(img):
    plt.imshow(mpimg.imread(img))
    plt.show()
```

[**Figura 11:** Metodo che prende in input una immagine e la stampa attraverso il metodo imshow() della libreria plot.]

```
#Filtro i Tag
def CleanStr(listStr):
    listRemove=["tumblr", "like", "follow", "instagram","likes", "photooftheday"]
    for rem in listRemove:
        for row in listStr:
            if row.find(rem) != -1:
                listStr.remove(row)
    print(listStr)
    return listStr
```

[**Figura 12:** Metodo che prende in input una lista contenente i tag di un post e controlla se al suo interno ci sono dei tag che contengono una delle parole non utili al fine di analizzare la similitudine.]

2.7. Calcolo delle parole simili:

Definite le due liste contenenti le parole chiave composte dai tag univoci filtrati e dagli oggetti riconosciuti dall'algoritmo di Object Detection, esse sono passate al metodo che ne rileva il numero di elementi uguali.

```
#Ricerca e stampa delle parole chiave trovate
print("\n-------------------------------------------------------------------------------------------------------------------\n")
counter=SearchWord(word_dict_blog1,word_dict_blog2)

print("\n-------------------------------------------------------------------------------------------------------------------\n")
print ("Punteggio parole chiave= ",counter)
```

```
#Ricerca e stampa delle parole chiave trovate
def SearchWord(dict1,dict2):
    counter=0
    for d1 in dict1:
        for d2 in dict2:
            if d1==d2:
                print("Coppia uguale: ",d1,"-",d2)
                counter=counter +1
                if d1[0:1]=="$":
                    counter=counter +1
    return counter
```

[**Figura 13 e 14:** Chiamata e definizione del metodo che prese due liste contenenti le parole chiave dei due blog, ne ritorna il numero di elementi presenti in entrambe le liste, dando un peso doppio agli elementi che sono stati predetti dall'algoritmo di Object Detection riconoscibili poiché in fase di inserimento nelle liste è stato aggiunto un carattere speciale "$" antecedente alla stringa.]

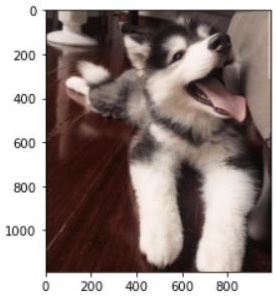
2.8. Valore finale della similitudine:

Ritornato il valore rappresentante il numero di parole chiave uguali dei due blog, esso è stato normalizzato è stampato sotto forma di percentuale.

```
min_value=min(len(word_dict_blog1),len(word_dict_blog2))
if counter == 0:
    print("I blog inseriti sono incompatibili")
else:
    counter_normalize=(counter/min_value)*100
    print ("Similitudine finale= ",counter_normalize,"%")
```

[**Figura 15:** Attraverso le seguenti istruzioni, è stato preso il numero di parole chiave uguali e il valore minimo tra le cardinalità delle due liste contenente le parole chiave, ne è stata fatta la divisione e il risultato è stato moltiplicato per 100 così da avere il risultato sotto forma di percentuale.
È stato preso il valore minimo tra le due cardinalità perché si vuole che se la lista più piccola è sottoinsieme dell'altra il risultato della normalizzazione sia 100%. Inoltre, considerando che gli oggetti predetti se presenti in entrambe le liste incrementeranno di 2 il contatore di parole uguali, il risultato finale potrebbe per assurdo superare il 100%.]

2.9. Esempio output:




```
['siberiian', 'siberian husky', 'husky', 'dog', 'puppy', 'puppies', 'indonesia', 'clothing line', 'huskies']
dog  :  81.98124170303345
dog  :  55.83466291427612
```

[**Figura 16:** In questa immagine è possibile vedere l'output del codice presente nella **Figura 10**, nello specifico, l'immagine del post, i tag utilizzati che sono stati filtrati e gli oggetti trovati all'interno dell'immagine dall'algoritmo di Object Detection.]

```
----------------------------------------------------------------------------------------------------

Parole chiave Blog 1

['$remote', '$dog', 'puppy', 'dog', 'puppies', 'cute', 'animals', 'adorable', '$cat', '$sports ball']

----------------------------------------------------------------------------------------------------

Parole chiave Blog 2

['$cat', 'siberiian', 'siberian husky', 'husky', 'dog', 'puppy', 'puppies', 'indonesia', 'clothing line', 'huskies', '$teddy bear', '$dog', '$bed', '$clock']

----------------------------------------------------------------------------------------------------

Coppia uguale:  $dog - $dog
Coppia uguale:  puppy - puppy
Coppia uguale:  dog - dog
Coppia uguale:  puppies - puppies
Coppia uguale:  $cat - $cat

----------------------------------------------------------------------------------------------------

Punteggio parole chiave=  7
Similitudine finale=  70.0 %
```

[**Figura 17:** In questa immagine è possibile vedere l'output del codice presente nelle **Figure 13-14-15**, nello specifico, la lista delle parole chiave per ogni blog, le coppie di stringhe uguali, il punteggio delle parole chiave e il valore della similitudine.]

3. Considerazioni finali

Attraverso l'applicativo appena realizzato, si è riusciti a definire il grado di similitudine tra due blog, dove la similitudine è definita come un valore che rappresenta il numero di parole chiave uguali usate nei post di tipo foto più recenti.
Inoltre, è stato introdotto il wrapper Pytumblr utile ad effettuare richieste API per il social Tumblr, con lo scopo di estrarre informazioni dai blog.
Ed infine, è stato spiegato come utilizzare l'algoritmo di Object Detection per la rilevazione degli oggetti interni alle immagini dategli in input.

In futuro potrebbero essere apportate migliorie al progetto, soprattutto nella parte che riguarda l'algoritmo di Object Detection, poiché il modello utilizzato non è sempre precisissimo.

[**Figura 16:** Anche se molti affermano che gli animali hanno atteggiamenti simili all'uomo, questa immagine è un esempio di come anche gli algoritmi possano sbagliare.]