2018

MACHINE LEARNING
SISTEMI INTELLIGENTI
Progetto

Angelo del Re 441476
Stefano Silvi 461993

Codice del progetto disponibile su GitHub:
https://github.com/mrsambo93/MLSII2018.git

Introduzione

Il progetto riguarda la costruzione di un sistema di raccomandazione di "post" provenienti da più social network, nel nostro caso Instagram e Flickr. Ha due fasi principali: la prima consiste nell'estrazione di post dai due siti a partire da "tag"; la seconda, invece, nello sfruttare le informazioni scaricate per effettuare, dato un input, delle raccomandazioni.

Tecnologie

Le tecnologie usate sono: Selenium, API di Flickr, MongoDB, Pandas e Scikit-learn.

Selenium, un framework software-testing per applicazioni web, fornisce uno strumento di riproduzione per la creazione di test senza la necessità di apprendere un linguaggio di scripting di prova. Inoltre, permette di testare un dominio di uno specifico linguaggio tramite i più famosi linguaggi di programmazione. Tra le sue componenti, abbiamo usato il WebDriver, una classe che permette di interfacciarsi con Google Chrome, o altri web browser, e simulare una sessione utente su un sito. La particolarità più rilevante è che il browser può essere avviato in modalità "headless", ovvero senza aprire l'interfaccia. Sostanzialmente le pagine vengono renderizzate in memoria.

Flickr mette a disposizione delle API che consentono di accedere liberamente e gratuitamente al proprio contenuto, a differenza di Instagram che è "leggermente" più limitato.

MongoDB è il più famoso database NoSQL al mondo e si adatta perfettamente ai nostri scopi.

Pandas è una libreria che mette a disposizione delle strutture dati, veloci e flessibili, che ben si adattano alle applicazioni Machine Learning. La struttura dati principale utilizzata è il DataFrame, che è una struttura tabulare bidimensionale.

Scikit-learn è una libreria open source di apprendimento automatico per il linguaggio di programmazione Python. Contiene algoritmi di Classificazione, regressione e clustering (raggruppamento) e SVM, regressione logistica, Classificatore bayesiano, k-mean e DBSCAN, ed è progettato per operare con le librerie NumPy e SciPy.

Esecuzione

Avviato lo scraping su uno specifico “hashtag”, nel nostro caso abbiamo preso in considerazione #sport e #food, si hanno due classi separate per l’interazione con Instagram e Flickr: InstagramScraper e FlickrScraper. Nella prima, innanzitutto, viene usato Selenium per stabilire la connessione con il driver di Chrome, esplicitando l’esecuzione in modalità “headless”. Una volta ottenuto il WebDriver, si passa ad effettuare il login sul portale di Instagram, passo obbligatorio per accedere a qualsiasi tipo di contenuto sul social. Grazie ai selettori offerti da Selenium si riempie la form di accesso e si effettua il login, il tutto automaticamente.

```
def login(self, driver):
    driver.get("https://www.instagram.com/accounts/login/?source=auth_switcher")

    WebDriverWait(driver, 10).until(EC.visibility_of_element_located(
        (By.CSS_SELECTOR, 'input[name="username"]'))).send_keys(self.credentials["user"])
    WebDriverWait(driver, 10).until(EC.visibility_of_element_located(
        (By.CSS_SELECTOR, 'input[name="password"]'))).send_keys(self.credentials["password"])
    button = WebDriverWait(driver, 10).until(EC.visibility_of_element_located(
        (By.CSS_SELECTOR, 'button[class="_5f5mN       jIbKX KUBKM      yZn4P    "]')))
    if button.is_enabled():
        button.click()
        print("Logged in")
```

Una volta fatto l’accesso si può ottenere la lista dei link appartenenti alla categoria specificata.

```
def get_posts_by_tag(self, driver, tag, number):
    self.close_pop_up(driver)
    driver.get("https:/www.instagram.com/explore/tags/" + tag.replace('#', ''))

    print("Collecting posts for " + tag)
    posts = set()
    pbar = tqdm(total=number)
    previous_len = 0
    while len(posts) < number:
        new_posts = WebDriverWait(driver, 10).until(EC.visibility_of_all_elements_located(
            (By.XPATH, '//*[@id="react-root"]/section/main/article/div[2]/div/div/div/a')))
        for el in new_posts:
            if len(posts) < number:
                posts.add(el.get_attribute('href'))
            if previous_len < len(posts):
                pbar.update(1)
                previous_len = len(posts)
        driver.execute_script('window.scrollTo(0, document.body.scrollHeight);')
    pbar.update(number - pbar.n)

    posts_list = list()
    for elem in posts:
        posts_list.append(elem)
    return posts_list
```

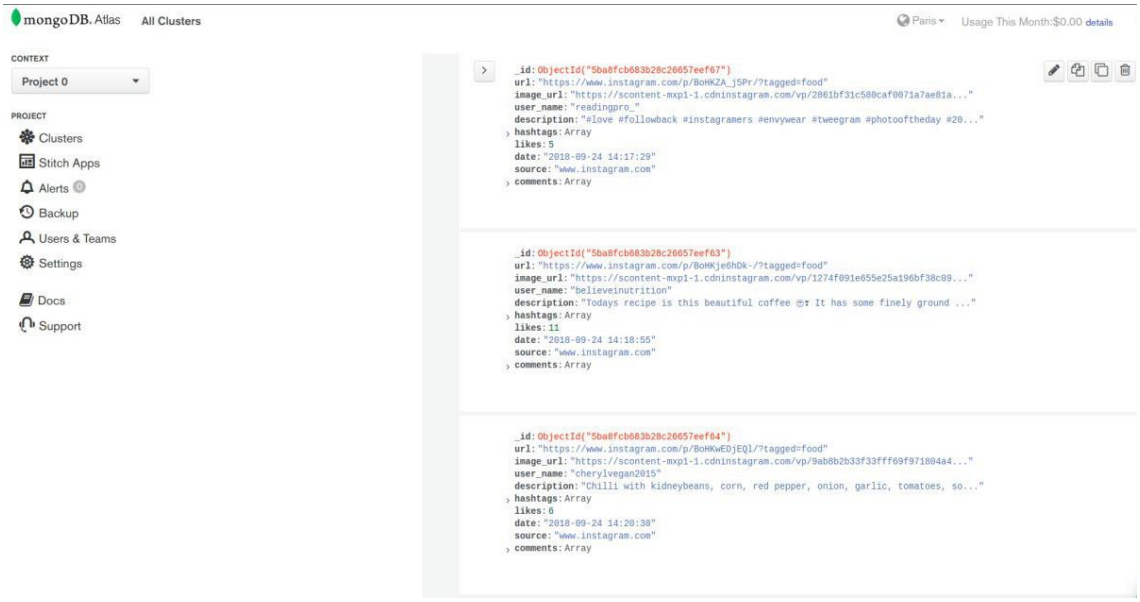
In output si avrà la lista degli url dai quali verranno estratti i singoli “post”. La fase di scraping, vera e propria, prevede la creazione di un oggetto “post” con i seguenti attributi: url, url dell’immagine o del video, nome dell’utente, descrizione, hashtag usati, numero di “likes”, data di caricamento, lista di commenti e hashtag contenuti in quest’ultimi. Questi oggetti “post” saranno salvati sul database.

```
    post["url"] = post_link
    user_name = WebDriverWait(driver, 10).until(EC.visibility_of_element_located(
        (By.CSS_SELECTOR, 'a[class="FPmhX notranslate TlrDj"]'))).get_attribute('title')
    description = WebDriverWait(driver, 10).until(EC.visibility_of_element_located(
        (By.XPATH,
            '//*[@id="react-root"]/section/main/div/div/article/div[2]/div[1]/ul/li[1]/div/div/div/span')))\
        .text
    hashtags = list({tag.replace('#', '') for tag in description.split() if tag.startswith('#')})
    likes_selector = self.get_likes(driver)
    likes = 0
    if likes_selector:
        likes_text = likes_selector.text
        likes = int(''.join(filter(str.isdigit, likes_text)))
    date = WebDriverWait(driver, 10).until(EC.visibility_of_element_located(
        (By.CSS_SELECTOR, 'time[class="_1o9PC Nzb55"]'))).get_attribute('datetime')
    image_url = self.get_image_url(driver)
    if image_url:
        post["image_url"] = image_url
    video_url = self.get_video_url(driver)
    if video_url:
        post["video_url"] = video_url
    post["user_name"] = user_name
    post["description"] = description
    post["hashtags"] = hashtags
```

Per quanto riguarda Flickr, la prima operazione e quella di collegarsi alle API tramite la libreria fornita. Il sito non necessita di effettuare un login vero e proprio, quindi si può passare direttamente alla fase di scraping. Flickr considera la foto come un “post”, ma, di fatto, non differisce di molto da uno di Instagram. Anche in questo caso creiamo degli oggetti “post” e avremo, in aggiunta agli altri, un attributo titolo, ma non avremo gli “hashtag” nei commenti, perché il social network non ne consente.

```
def scrape_hashtag(self, flickr, tag):
    print('Scraping on flickr')
    for i in range(1, self.PAGES + 1):
        print('\nPage ' + str(i))
        photos = flickr.photos.search(text=tag.replace('#', ''), per_page=self.PAGE_LIMIT + 1, page=i,
                                      extras=self.extras, sort='interestingness-desc')
        posts = list()
        for photo in tqdm(photos['photos']['photo']):
            comments_full = flickr.photos.comments.getList(photo_id=photo['id'])
            post = dict()
            url = 'https://www.flickr.com/photos/' + photo['owner'] + '/' + photo['id']
            post['url'] = url
            post['source'] = 'www.flickr.com'
            post['title'] = photo['title']
            post['user_name'] = photo['ownername']
            sizes = flickr.photos.getSizes(photo_id=photo['id'])['sizes']['size']
            image_url = sizes[-1]['source']
            post['image_url'] = image_url
            post['description'] = self.strip_tags(photo['description']['_content'])
            post['hashtags'] = photo['tags'].split()
            post['likes'] = photo['count_faves']
            post['date'] = datetime.utcfromtimestamp(int(photo['dateupload'])).strftime('%Y-%m-%d %H:%M:%S')
            comments = list()
            try:
                for c in comments_full['comments']['comment']:
```

Ottenuti tutti i “post” avremo nel database dei documenti di questo tipo:



Ci siamo appoggiati su MongoDB Atlas per l’hosting del database, ma l’applicazione funziona correttamente anche su un’istanza locale.

Per la nostra esecuzione prendiamo, all’incirca, 1.000 “post” per ogni “tag” sottomesso da entrambi i social network, quindi poco meno di 2.000 per “tag”.

Per quanto riguarda i tempi di scraping, su Instagram sono evidentemente più lunghi rispetto a Flickr, poiché le API permetto un accesso più immediato e, comunque, il rendering di pagine, che spesso hanno contenuti caricati dinamicamente, richiede del tempo.

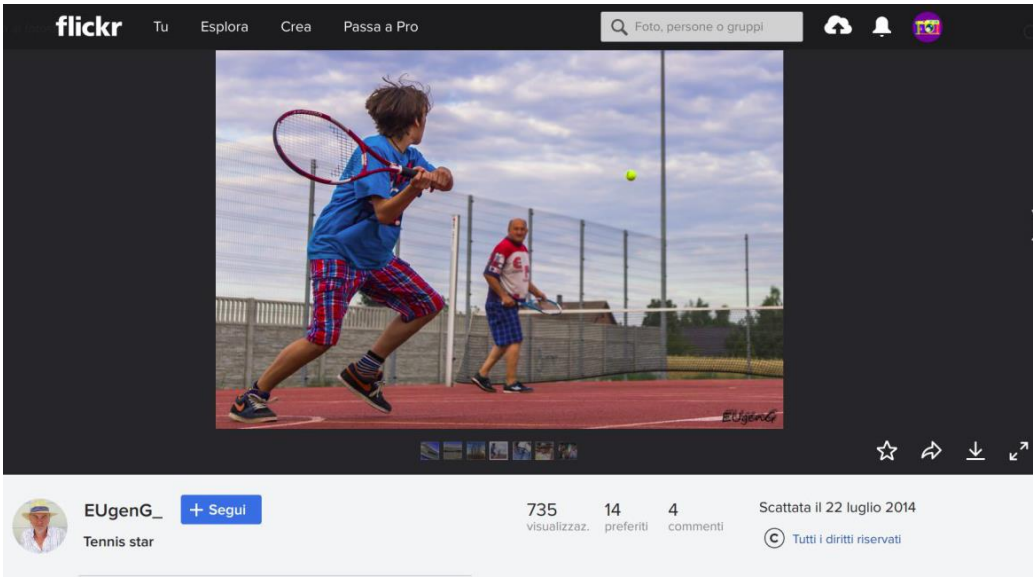
Recommender System

Per le raccomandazioni viene usata la classe Recommender: prende in input l'url di un nuovo "post", appartenente alle categorie di riferimento, invocando l'opportuno scraper ottiene le informazioni di quel "post". Quindi scarica tutti i "post" salvati nel database e li accorpa, sia l'input che quelli scaricati, in un DataFrame. Prima di effettuare il calcolo del tf-idf, la classe provvede a fare una pulizia dei dati, come, ad esempio, la rimozione di simboli particolari, "emoji", ecc. Dopodiché combina le feature sulle quali andrà ad effettuare i calcoli. Nel nostro caso, utilizziamo titolo (se presente), descrizione e "hashtag". Su questa combinazione calcola tf-idf e la coseno similarità tra gli elementi nel Data Frame. A questo punto il Recommender ritorna i 10 elementi più somiglianti all'input.

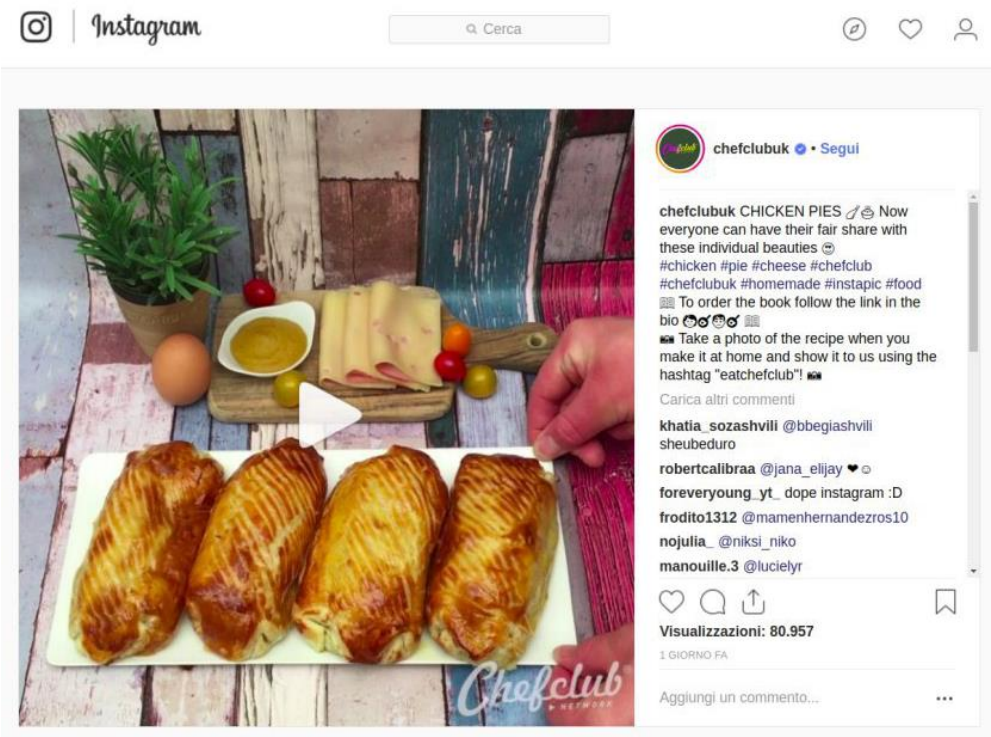
```
tfidf = TfidfVectorizer(stop_words='english')

df = pd.DataFrame(posts)
df['title'] = df['title'].fillna('')
df['title'] = df['title'].apply(MongoConnector.clean_string).apply(remove_other_languages)
df['comments_hashtags'] = df['comments_hashtags'].fillna('')
df['comments'] = df['comments'].apply(get_comments)
df['comments'] = df['comments'].apply(remove_from_all)
df['description'] = df['description'].apply(MongoConnector.clean_string)
df['description'] = df['description'].apply(remove_other_languages)
df['soup'] = df.apply(create_soup, axis=1)
tfidf_matrix = tfidf.fit_transform(df['soup'])
indices = pd.Series(df.index, index=df['url']).drop_duplicates()
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
index = indices[post['url']]
sim_scores = list(enumerate(cosine_sim[index]))
sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
sim_scores = sim_scores[1:11]
movie_indices = [i[0] for i in sim_scores]
return df['url'].iloc[movie_indices].values.tolist()
```

Input:

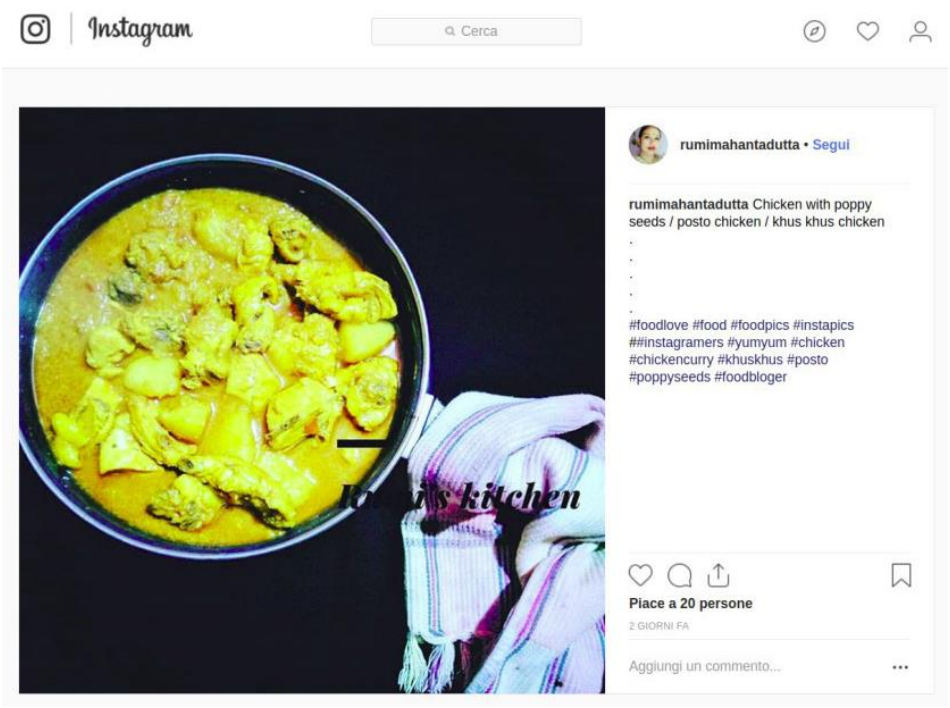
Output raccomandazione:

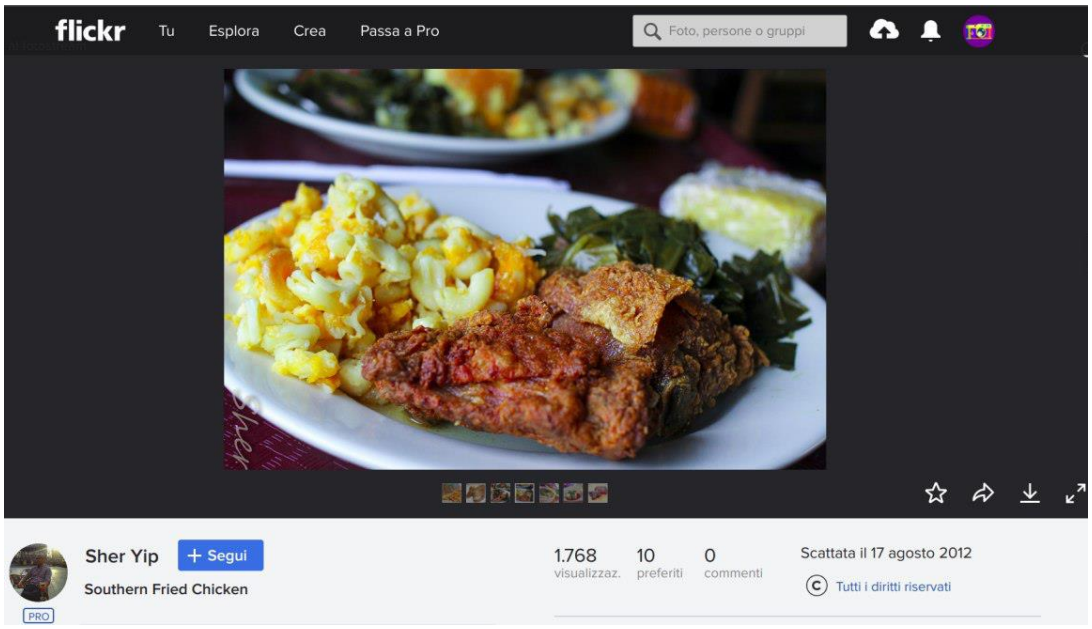

flickr
Tu
Esplora
Crea
Passa a Pro
Foto, persone o gruppi
EUgenG_
+ Segui
Tennis star
735
visualizzaz.
14
preferiti
4
commenti
Scattata il 22 luglio 2014
Tutti i diritti riservati

Input:


Instagram
Cerca
chefclubuk • Segui
chefclubuk CHICKEN PIES Now everyone can have their fair share with these individual beauties
#chicken #pie #cheese #chefclub #chefclubuk #homemade #instapic #food
To order the book follow the link in the bio
Take a photo of the recipe when you make it at home and show it to us using the hashtag "eatchefclub"!
Carica altri commenti
khatia_sozashvili @bbegiashvili sheubeduro
robertcalibraa @jana_elijay
foreveryoung_yt_ dope instagram :D
frodito1312 @mamenhernandezros10
nojulia_ @niksi_niko
manouille.3 @lucielyr
Visualizzazioni: 80.957
1 GIORNO FA
Aggiungi un commento...
Chefclub

Output Raccomandazione:





Conclusione

In conclusione, abbiamo raggiunto lo scopo prefissato, ottenendo risultati soddisfacenti. Per migliorare le previsioni, per possibili sviluppi futuri, si potrebbe cercare di tradurre tutto il testo in inglese, tramite una libreria che si appoggia sulle API di Google Traduttore, del quale la nostra applicazione ha già l'implementazione, ma la versione attuale della libreria (2.3.0) non funziona (sono previsti dei fix a breve). Un ulteriore miglioramento della previsione, si potrebbe ottenere cercando di estrarre informazioni anche dalle immagini, ad esempio Clarifai offre delle API adatte allo scopo, che, però, sono a pagamento dopo una certa soglia. L'applicazione, al momento, gira solo da terminale, si potrebbe introdurre un notebook Jupyter per facilitarne l'utilizzo. Inoltre, si potrebbe estendere il funzionamento su altri social network.