

# UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



Tesina per il progetto di  
Fondamenti di Data Science e Machine Learning

## Social Mapper

**Marco MARCHIONNO**

Mat. 0552501098

**Gaetano CASILLO**

Mat. 0522501057

ANNO ACCADEMICO 2020/2021

# Introduzione

Oggi è quasi impossibile trovare qualcuno che non abbia almeno un account di un qualche social network, ed è molto comune che chi lo abbia pubblici anche foto e video con annesse didascalie sul proprio profilo. Inoltre, social network come Instagram, Linkedin, Facebook, danno la possibilità di dare una descrizione di sè stessi, talvolta con informazioni personali e sensibili, questa descrizione è chiamata *Bio*. Tutti questi elementi creano un *one-way mirror* sulla vita degli utenti, che permette a persone esterne di raccogliere informazioni su di essi senza che quest'ultimi se ne accorgano. La conseguenza di una società sempre più indirizzata verso i social è che ci sono miliardi e miliardi di utenti che condividono informazioni per un totale decine e decine di yottabyte di dati. Avere a disposizione una così grande quantità di informazioni ha fatto sì che compagnie di marketing si siano specializzate in pubblicità mirate o che agenti malevoli possano rubare l'identità di alcuni utenti attraverso tecniche di *social engineering*. A causa di questa miniera d'informazioni, le varie aziende hanno adottato rigidi protocolli di sicurezza a tutela dei dati sensibili, tuttavia, spesso sono gli stessi utenti a rendere pubbliche tali informazioni rendendo la vita più semplice a chi vuole raccogliere dati. La privacy è un argomento molto ignorato dagli utenti nonostante le pagine e pagine di consigli da parte dei social network sulle politiche di privacy. Da queste basi, il nostro progetto si pone l'obiettivo di generare un codice di fiscale di un utente randomico attraverso le informazioni da lui fornito e nel processo fare una statistica di quanti utenti abbiano condiviso informazioni sensibili. Le informazioni sono state raccolte da Instagram attraverso l'uso di social mapper e instaloooter per un totale di 108 account con circa 15 media per account.

# Capitolo 1

## Related work

Una parte del progetto in questione ha previsto il crawling di instagram per poter scaricare tutte le informazioni che ci servivano dai profili selezionati casualmente e dopo di analizzarle. Abbiamo quindi studiato dei lavori che ci aiutassero a capire come poter portare a termine questo compito nel migliore dei modi.

- **Yolo detection:** Da questo progetto abbiamo osservato com'è facile da usare e com'è potente il framework YOLO, per questo si è deciso di usare YOLO nel progetto in quanto già pronto e funzionante per risparmiare nei tempi di sviluppo del progetto [1].
- **Web Scraping Instagram with Selenium Python:** Avendo poca esperienza con il framework di selenium questo articolo è stato fondamentale nel dare spiegazioni su come usare framework e come usarlo per fare crawling su instagram [2].
- **Tips For Web Scraping:** Quest'articolo è stato utile in quanto ha permesso di non creare ancor più account di quanto non sia stato già fatto poichè le politiche di bot detection di Instagram hanno spesso bannato gli account creati per effettuare crawling. Seguendo questo sito si è riusciti a ridurre il numero di account da creare [3].

- **Survey on NLP:** L'articolo presenta cinque tipologie di NLP in maniera approfondita con esempi di utilità. Alla fine di un accurato brainstorming si è arrivati alla conclusione che il NER (*Named Entity Recognition*) sia la tipologia di NLP che più si avvicinava allo scopo del progetto [4].
- **An Exercise In Identifying Fake Instagram Profiles With TensorFlow:** L'articolo presenta alcuni parametri interessanti sul come riconoscere se un account è fake o meno, da qui si sono prese alcune considerazioni sul come fare per ritenere autentico un account [5]

# Capitolo 2

## Tencologie utilizzate

In questo capitolo verranno illustrate le tencologie utilizzate quali linguaggi, librerie e framework.

### 2.1 Social Mapper

Social Mapper, implementato dal ricercatore Jacob Wilkin, permette di ricercare le persone su differenti social network quali: Facebook, Linkedin, Instagram, VKontakte, Twitter, Pinterest, Weibo e Douban. Il progetto è strutturato nelle seguenti componenti:

- **social\_mapper.py:** Componente principale del progetto, in cui ‘e definito il parser per l’immissione dei parametri da linea di comando e sono presenti le funzioni per ricercare le persone attraverso le varie modalità di input sui differenti social. Questa componente, infine, si occupa della formattazione dei risultati ottenuti e della costruzione del file di output, sia in formato HTML che in formato .csv;
- **un modulo differente per ogni social network:** come visibile in Figura 2.1 sono presenti 8 componenti differenziate che si occupano di effettuare il login sul sito web, di effettuare la ricerca del profilo e restituire il risultato in output.

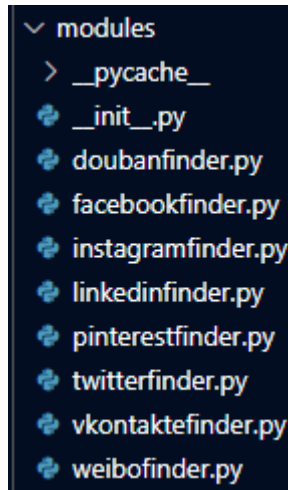


Figura 2.1

L'esecuzione può essere lanciata da linea di comando con diverse modalità di input:

- prendendo in input una cartella contenente le immagini delle persone da cercare denominate con il nome ed il cognome dell' individuo;
- tramite un file csv che ha come primo campo il nome ed il cognome dell' individuo e come secondo campo presenta il link ad un'immagine del soggetto da ricercare;
- tramite la ricerca delle persone per azienda su Linkedin salvando tutte quelle trovate o in un'apposita cartella con il nome dell' azienda oppure in un csv inserendo il link dell'immagine trovata.

Social Mapper utilizza il riconoscimento facciale per individuare correttamente la persona sui diversi social network utilizzando le funzioni di libreria di Face Recognition. L'accuratezza nella ricerca può essere specificata tramite un parametro dato in input da linea di comando, utilizzando "fast" per ricerche più veloci ma meno accurate ed "accurate" per ricerche più accurate ma più lente.

Al termine delle ricerche Social Mapper restituisce due differenti output:

- un output in formato csv con il nome ed il cognome della persona da ricercare ed link a tutti i social network in cui l'individuo è stato trovato;

- un output in formato HTML con le foto prese in input con il nome e cognome nella prima colonna e le foto ritrovate sui differenti social network nelle altre colonne con i relativi link

Il progetto originale su GitHub, inizialmente scaricato, non funzionava su nessun social network poichè i siti web risultano in costante aggiornamento. Ciò comporta che le posizioni degli elementi e le classi dei “div”, dove risiedono le informazioni, cambino continuamente. In seguito alla natura mutevole delle pagine web, ciò comporta che i moduli di Social Mapper debbano essere costantemente aggiornati per rimanere in linea ai social network e per garantirne il corretto funzionamento.

Non è stato possibile però aggiornare i moduli dedicati ai social Weibo e Douban, poichè non ci è stato possibile creare un account fittizio per svolgere le nostre estrapolazioni.

In tali social la registrazione richiede obbligatoriamente un numero di telefono per l'autenticazione, ma il suffisso italiano non risulta accettato. Non potendo quindi accedere alle informazioni dei profili su tali siti abbiamo preferito tralasciarli per concentrarci sui restanti social.

### 2.1.1 Funzionalità di crawling aggiunte e migliorate

Per poter adempiere all'obiettivo del nostro progetto, e farlo nel modo migliore possibile, abbiamo dovuto allargare le feature di Social Mapper. In particolare, avevamo la necessità di poter fare crawling fornendo al crawler soltanto una lista di nomi, nient'altro. Inoltre, abbiamo riscontrato la necessità di riscrivere parzialmente la funzione di login all'account Instagram.

- **login:** Il metodo *doLogin* è stato parzialmente riscritto poichè come scritto nella descrizione di social mapper, non solo è cambiata la pagina html, ma sono stati anche aggiunti controlli sui cookies da dover considerare, altrimenti la homepage di instagram non veniva caricata.
- **csv-nophoto:** Questa è l'opzione aggiunta al crawler che permette, attraverso un csv di nomi e cognomi di persona, di effettuare crawling sui

profili collegati a tali nomi. Questo ci è servito per poter creare velocemente un dataset di profili Instagram, in quanto la necessità di avere una foto del soggetto avrebbe rallentato di parecchio la generazione.

- **getInstaProfilePhotos:** Questo è il metodo che ha permesso di scaricare 15 post utente da ogni profilo che rispettava, il metodo è stato utilizzato assieme alla libreria *Instalooter*.

Di seguito verranno elencate altre funzionalità di crawling create, ma non aggiunte a Social Mapper

- **bio.py:** Questo è lo script che ha permesso di scaricare le biografie generate dagli utenti senza far segnalare l'account.
- **didascalial.py:** Questo è lo script che partendo dalle immagine scaricate dagli utenti, tramite il mediaid ricrea il link del post e scarica informazioni quali: luogo, utenti taggati, descrizione, hashtag e la didascalia.
- **music\_recognizer.py:** Tramite questo script si è riusciti a separare i video dall'audio e mandare quest'ultimo alle RestAPI di *audd.io* che ha restituito le musiche riconosciute.

### 2.1.2 Librerie utilizzate

Di seguito verranno elencate e spiegate altre librerie utilizzate oltre a Social Mapper

- **pycorenlp:** Questa libreria [6] è un wrapper python per il modulo di NLP realizzato dalla *Stanford University*, detto appunto *Stanford CoreNLP*, sviluppato in Java. CoreNLP permette agli utenti di ricavare annotazioni linguistiche nei testi, inclusi: *token boundaries* e *sentence boundaries*, porzioni del discorso, named entities *Named Entity Recognition*, valori numerici e temporali, analisi delle dipendenze e dei raggruppamenti, analisi dei riferimenti, sentiment analysis e quote attribution.



CoreNLP supporta attualmente 6 lingue: Arabo, Cinese, Inglese, Francese, Tedesco e Spagnolo.

Pycorenlp utilizza degli *annotators*, attraverso i quali specifici alla libreria quali task di NLP deve eseguire. Gli annotator utilizzati in questo progetto sono:

- **ssplit:** Sta per *Sentence Splitting* ed è il processo di dividere il testo in frasi.
  - **ner:** Sta per *Named Entity Recognition*. Consiste nel riconoscere named entities (come persona, lavoro, luoghi, etc) nel testo. Principalmente, questo annotatore usa uno o più modelli Machine Learning per etichettare le entità, ma può anche chiamare componenti basati su regole specializzate, come per l’etichettatura e l’interpretazione di orari e date. Le entità numeriche che richiedono la normalizzazione, ad esempio le date, hanno il loro valore normalizzato memorizzato. Le named entities che abbiamo ricercato sono le seguenti: *person, location, organization, number, date, email, url, city, state\_or\_province, country, nationality, religion, title, ideology*
  - **depparse:** Sta per *Dependency Parsing*. Fornisce un parser di dipendenze sintattiche veloce. Venogno generati tre output basati su dipendenze, come segue: *Basic Dependencies*, non codificate; *Enhanced Dependencies*, le dipendenze migliorate; e infine *Enhanced++ Dependencies*, le dipendenze ulteriormente migliorate.
- **YOLOV5:** Questo framework [7] è stato utilizzato per fare object recognition all’interno di immagini e video. YOLO sta per ”you only look once” ed è basato sulle CNN, come suggerisce il nome, è infatti necessario solo una iterazione attraverso una rete neurale per identificare gli oggetti. I motivi per cui si è scelto questo framework sono: Velocità, Precisione, è molto facile da riaddestrare aggiungendo altri oggetti oltre a quelli già presenti ed infine è Open Source. L’idea di fondo dell’algo-

ritmo è che divide il frame in una griglia SxS (dipende dalla versione dell'algoritmo YOLO) e produce delle bounding boxes che racchiudono gli oggetti identificati con la loro confidenza. Si è deciso di utilizzare la versione v5 in quanto è quella più recente ed è stata resa molto più facile da utilizzare rispetto alle altre.

- **Instalooter:** Questa libreria [8] python ha permesso di scaricare egregiamente (nonostante alcuni problemi legati a piccoli bug) foto e video dagli account di Instagram. Sono necessari soltanto l'username dell'utente e il numero di quante foto e/o quanti video si vuole scaricare. Sono presenti anche moduli per scaricare informazioni riguardanti i post quali le didascalie o gli hashtag di un determinato utente.
- **MoviePy:** MoviePy[9] è un modulo python per il video editing, che può essere utilizzato per operazioni di base (come tagli, concatenazioni, inserimento di titoli), compositing video (a.k.a. editing non lineare), elaborazione video, o per creare effetti avanzati. Può leggere e scrivere i formati video più comuni, tra cui GIF. In questo progetto è stata usata per convertire i video mp4 in file audio mp3 da dare in pasto ad audd.
- **fuzzywuzzy:** E' una libreria [10] python che tramite la *Levenshtein Distance* calcola la similarità tra due parole. E' stata utilizzata per fare statistica su musiche e nomi.
- **audd.io:** È una RestAPI per la music recognition, ovvero rileva e identifica la musica presente in un qualsiasi contenuto, nel nostro caso un file mp3. Dalla homepage del sito web è possibile sottoscrivere per ottenere un token per l'utilizzo del servizio. Il piano gratuito comprende un massimo di 300 richieste al mese.
- **names:** È un modulo python che permette di generare dei nomi casuali. È stato utilizzato inizialmente nel nostro progetto per creare un csv di nomi da dare in pasto all'opzione *csv-nophoto* di social mapper.

# Capitolo 3

## Dataset

Parte fondamentale di questo progetto è la creazione di un dataset. Grazie alla nuova funzionalità di social mapper *csv-nophoto*, dando in pasto al crawler un *CSV* di nomi di persona, ha dato in output un dataset cartelle in cui ogni folder rappresenta un account e al suo interno sono presenti foto e video. La generazione ha coinvolto un totale di 108 account divisi equamente in maschi e femmine. Per ogni nome di persona sono stati considerati un massimo di 5 account collegati a tale nome e per ognuno di essi sono stati presi al massimo 10 post di foto e 5 post di video. Tuttavia, la presenza dei *carousel*, ha fatto in modo che il numero di foto e di video recuperati per ogni utente non sia sempre lo stesso.

Di seguito verranno elencati gli script utilizzati e i file ottenuti per creare il dataset.

- **getInstaProfilePhotos:** Il metodo ha generato circa 15 media per account tra foto e video prese da instagram
- **didascalia.py:** Lo script ha generato il file *info.json* contenente utenti taggati, didascalia e location, ove possibile anche musica e descrizione dei media.
- **bio.py:** Lo script ha generato il file *bio.json* contente la biografia degli utenti.

- **music\_recognizer.py** Lo script ha generato il file *music\_info.json* contenente le musiche riconosciute nei video.
- **generayolo.py** e *leggiimamgini.py* hanno generato il file *dentrofoto.json* contenenti il numero e il nome degli oggetti riconosciuti presenti nei media con una confidenza superiore al 55% nei video e al 45% nelle immagini
- **interessi.py** e *normalizza.py* hanno categorizzato gli interessi degli utenti in una scala da 1 a 5 cercando di carpirli osservando gli oggetti più postati sui loro social.
- **bio\_nlp.py** e *descrizione\_nlp.py* hanno preso rispettivamente in input i file *info.json* e *bio.json* per fare un'analisi di NLP e generare i file *nlp.json* e *didascalia\_nlp.json* contenenti gli output degli annotator descritti in 2.1.2.

Dai file generati si è deciso di creare un unico json contenente le informazioni ritenute più importanti. Grazie a *merge\_json.py* che univa tutti i json per un account e generava il file *general.json* e *generate\_complete\_dataset.py* che univa tutti i *general.json* in *dataset\_completo.json*. I campi presenti nel *general.json* sono:

- **username:** l'username dell'account in questione.
- **info\_name:** il nome della persona che gestisce tale account presente nel file *info.json*.
- **info\_locations:** tutte le locations legate all'account presenti nel file *info.json*.
- **music:** la musica ottenuta dal file *info.json* e quella data in output dallo script *music\_recognizer.py*.
- **didascalia\_location:** tutte le location ottenute dal file *info.json*
- **didascalia\_hashtags:** tutti gli hashtags ottenuti dal file *info.json*

- **nlp\_name:** nomi riconosciuti tramite NER ottenuti dal file *nlp.json*
- **didascalìa\_nlp\_name:** nomi riconosciuti tramite NER ottenuti dal file *info.json*.
- **nlp\_location:** le location riconosciute dagli algoritmi di NER presenti nel file *nlp.json*.
- **didascalìa\_nlp\_location:** le location riconosciute dagli algoritmi di NER presenti nel file *didascalìa\_nlp.json*.
- **nlp\_organization:** le organizzazioni riconosciute dagli algoritmi di NER presenti nel file *nlp.json*
- **didascalìa\_nlp\_organization:** le organizzazioni riconosciute dagli algoritmi di NER presenti nel file *didascalìa\_nlp.json*.
- **nlp\_number:** i numeri riconosciuti dagli algoritmi di NER presenti nel file *nlp.json*.
- **didascalìa\_nlp\_number:** i numeri riconosciuti dagli algoritmi di NER presenti nel file *didascalìa\_nlp.json*.
- **nlp\_date:** le date riconosciute dagli algoritmi di NER presenti nel file *nlp.json*.
- **didascalìa\_nlp\_date:** le date riconosciute dagli algoritmi di NER presenti nel file *didascalìa\_nlp.json*.
- **nlp\_email:** le email riconosciute dagli algoritmi di NER presenti nel file *nlp.json*.
- **didascalìa\_nlp\_email:** le email riconosciute dagli algoritmi di NER presenti nel file *didascalìa\_nlp.json*.
- **nlp\_url:** gli url riconosciuti dagli algoritmi di NER presenti nel file *nlp.json*.

- **didascalialnp\_url:** gli url riconosciuti dagli algoritmi di NER presenti nel file *didascalialnp.json*.
- **nlp\_city:** le città riconosciute dagli algoritmi di NER presenti nel file *nlp.json*.
- **didascalialnp\_city:** le città riconosciute dagli algoritmi di NER presenti nel file *didascalialnp.json*.
- **nlp\_state\_or\_province:** gli stati o le province riconosciute dagli algoritmi di NER presenti nel file *nlp.json*.
- **didascalialnp\_state\_or\_province:** gli stati o le province riconosciute dagli algoritmi di NER presenti nel file *didascalialnp.json*.
- **nlp\_country:** gli stati riconosciuti dagli algoritmi di NER presenti nel file *nlp.json*.
- **didascalialnp\_country:** gli stati riconosciuti dagli algoritmi di NER presenti nel file *didascalialnp.json*.
- **nlp\_nationality:** nazionalità riconosciute dagli algoritmi di NER presenti nel file *nlp.json*.
- **didascalialnp\_nationality:** le nazionalità riconosciute dagli algoritmi di NER presenti nel file *didascalialnp.json*.
- **nlp\_religion:** le religioni riconosciute dagli algoritmi di NER presenti nel file *nlp.json*.
- **didascalialnp\_religion:** le religioni riconosciute dagli algoritmi di NER presenti nel file *didascalialnp.json*.
- **nlp\_titles:** i lavori riconosciute dagli algoritmi di NER presenti nel file *nlp.json*.
- **didascalialnp\_titles:** i lavori riconosciute dagli algoritmi di NER presenti nel file *didascalialnp.json*.

- **nlp\_ideology:** le ideologie riconosciute dagli algoritmi di NER presenti nel file *nlp.json*
- **didascalia\_nlp\_ideology:** le ideologie riconosciute dagli algoritmi di NER presenti nel file *didascalia\_nlp.json*.
- **interessi:** gli interessi degli utenti divisi per categoria in una scala da 1 a 5.
  - *sociale\_category:* categoria che indica quanto è socievole una persona indicata da quanto spesso pubblica foto e video con altre persone
  - *kitchen\_category:* categoria che indica quanto è appassionata alla cucina una persona, indicata da quanto spesso pubblicata foto e video contenenti oggetti appartenenti alla cucina
  - *classy\_category:* categoria che indica quanto è appassionata al mondo dell'alta classe una persona, indicata da quanto spesso pubblicata foto e video contenenti oggetti costosi
  - *sport\_category:* categoria che indica quanto è appassionata allo sport una persona, indicata da quanto spesso pubblicata foto e video contenenti oggetti appartenenti al contesto sportivo quali racchette da tennis, snowboard, scarpe con tacchetti e altro
  - *gamer\_category:* categoria che indica quanto è appassionata al mondo del gaming una persona, indicata da quanto spesso pubblicata foto e video contenenti oggetti appartenenti al gaming quali pc, joystick e altro
  - *tvaddicted\_category:* categoria che indica quanto è appassionato alle serie tv/film una persona, indicata da quanto spesso pubblicata foto e video contenenti oggetti quali televisori, telecomandi, divani e altro

- *safefood\_category*: categoria che indica quanto è appassionata al mangiar bene una persona, indicata da quanto spesso pubblicata foto e video contenenti cibi quali verdure, frutta e altro
- *animali\_category*: categoria che indica quanto è appassionata agli animali una persona, indicata da quanto spesso pubblicata foto e video contenenti animali
- *junkfood\_category*: categoria che indica quanto spesso una persona mangia cibi non particolarmente sani, indicata da quanto spesso pubblicata foto e video contenenti oggetti appartenenti al fast food, patatine e altro
- *road\_category*: categoria che indica quanto spesso una persona esce di casa, indicata da quanto spesso pubblicata foto e video contenenti oggetti appartenenti ad ambienti della strada quali auto, cartelli, panchine, parchi e altro
- *travel\_category*: categoria che indica quanto è appassionata al viaggiare una persona, indicata da quanto spesso pubblicata foto e video contenenti oggetti riguardanti il viaggio come aerei, treni, valige e altro



# Capitolo 4

## Grafici sui dati

In questo capitolo verranno mostrati e spiegati i grafici che si sono ottenuti grazie ai dati raccolti durante il progetto. Tali grafici mostrano quanto spesso gli utenti condividono dati personali o elementi che possono aiutare nel fare pubblicità mirate.

### 4.1 Grafici

#### 4.1.1 Dati semplici

In questa sezione verranno mostrati quanto spesso gli utenti scrivono nella propria biografia o nella propria didascalia elementi quali: Ideologia politica, religione, lavoro attuale, email personali, url e altri.

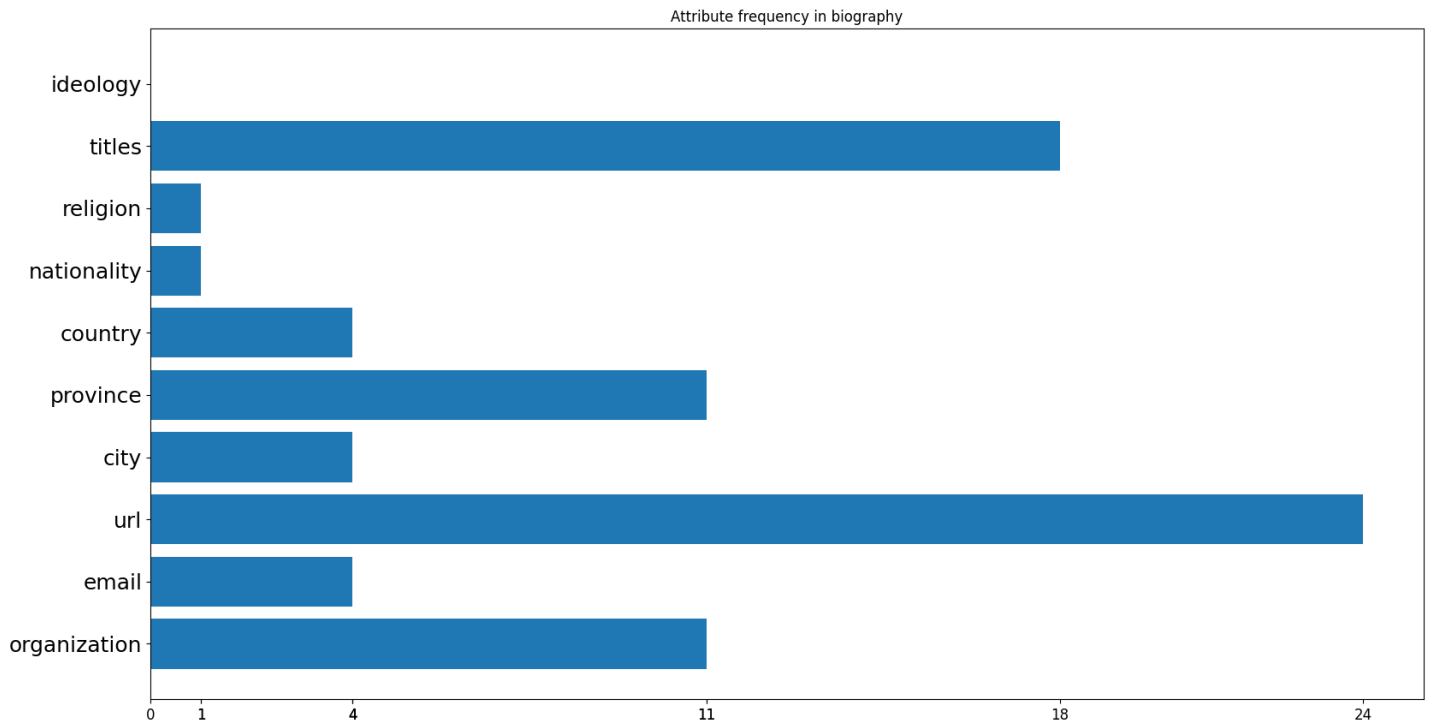


Figura 4.1: Frequenza degli attributi nella bio

Nella biografia, l'elemento più presente è un link ad un sito personale, al proprio canale youtube o talvolta ad un sito per delle raccolte fondi. Il secondo elemento più pubblicato è il proprio lavoro, seguito a pari merito da quelli che nlpcore ha riconosciuto come organizzazione (luogo di lavoro) e come il luogo in cui si abita. Nlpcore, ancora, ha riconosciuto che 4 utenti hanno pubblicato la propria email, la propria città e il proprio stato di appartenenza, mentre quasi nessuno ha pubblicato informazioni sulla propria religione e nazionalità. Infine, nessuno ha pubblicato informazioni riguardo la propria ideologia.

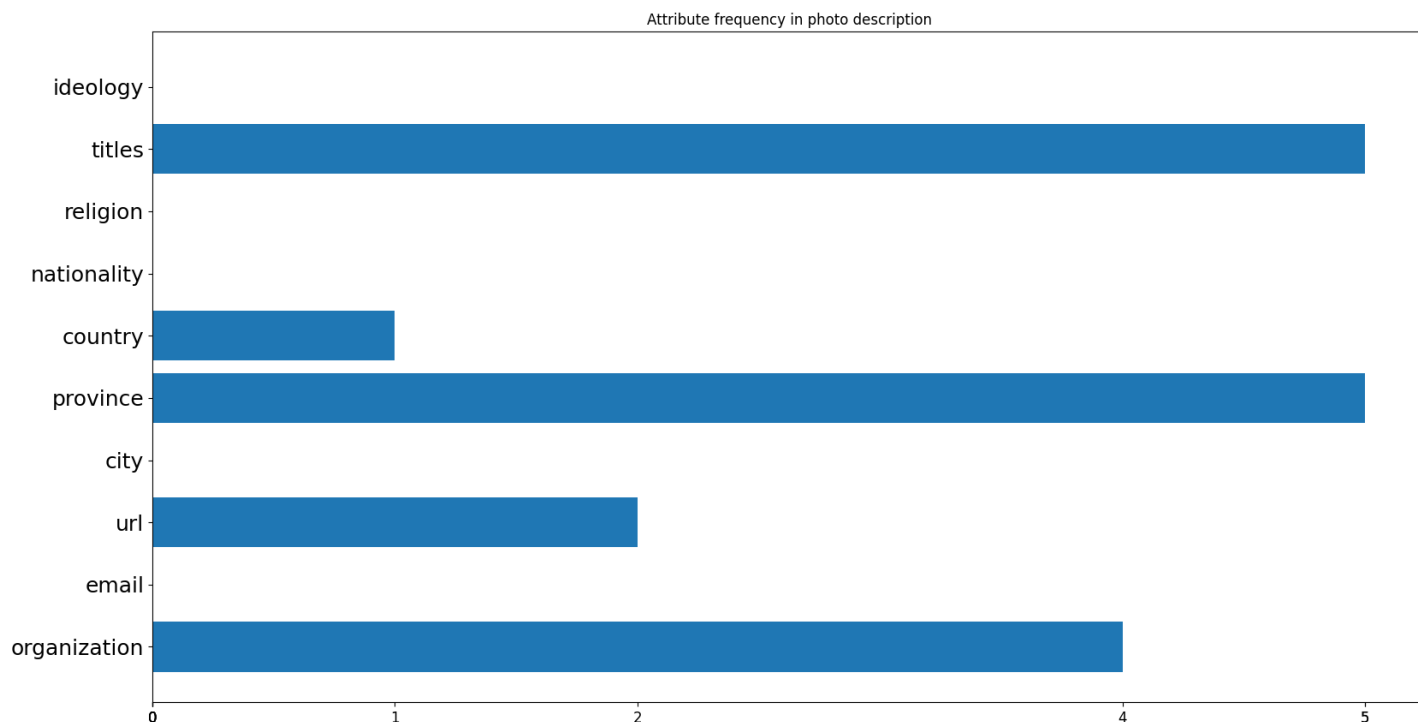


Figura 4.2: Frequenza degli attributi nella didascalia

Nelle descrizioni delle foto invece nessuno ha pubblicato elementi riguardo la propria religione, email o ideologia, mentre 5 utenti hanno scritto la propria provincia di appartenenza o il lavoro che stavano svolgendo in quella foto. Sono 4 le persone che hanno pubblicato informazioni sulla propria organizzazione, mentre solo una ha scritto nella propria didascalia informazioni sullo stato in cui si trovavano.

Per le informazioni sulla geolocalizzazione se ne approfondirà meglio in seguito. Infine notiamo che nessuno ha scritto informazioni riguardanti città, nazionalità, email, religioni e/o ideologie.

#### 4.1.2 Grafici sulle canzoni

In questa sezione verranno mostrati le statistiche sulle canzoni e gli artisti pubblicati dagli utenti nel proprio feed.

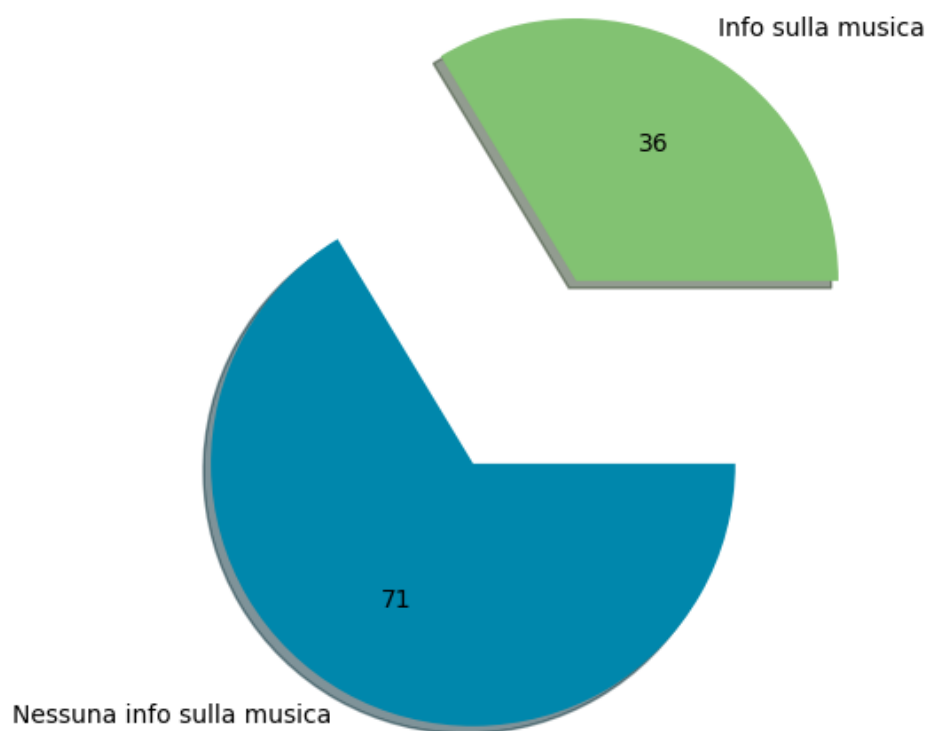


Figura 4.3: Tipologie di date presenti nella bio

Come si può facilmente evincere da questo grafico sono 36 le persone su cui siamo riusciti a reperire informazioni riguardo ai propri gusti musicali. Vediamole nel dettaglio.

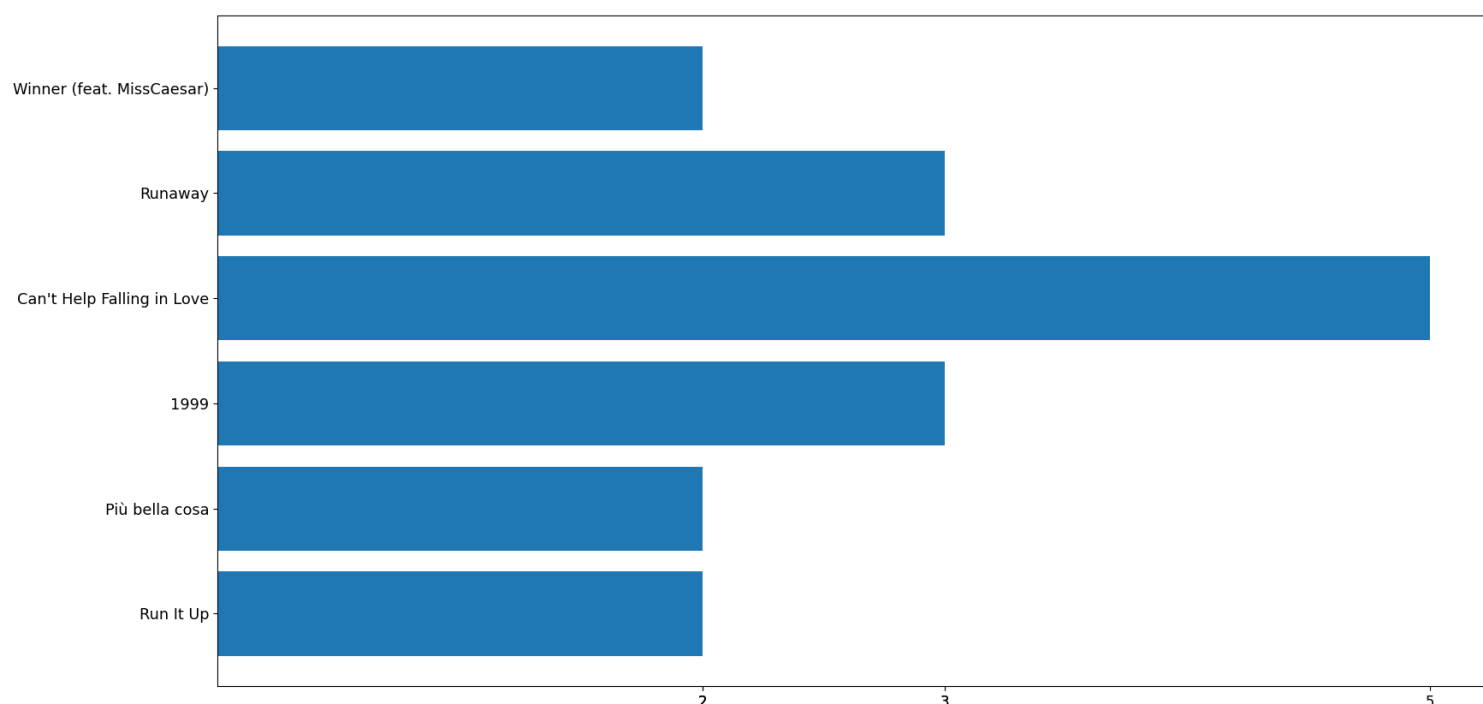


Figura 4.4: Canzoni più ascoltate dagli utenti

La canzone più condivisa tra i 36 utenti è *Can't help falling in love*, classico inno all'amore di Elvis Presley seguita da *1999* di Charli XCX & Troye Sivan a pari merito con *Runaway* di Aurora. Infine, tra le canzoni più ascoltate abbiamo *Winner*, *Run It Up* e *Più bella cosa* di Eros Ramazzotti. Possiamo osservare, quindi, che l'unica canzone facente parte dei grandi classici è *Can't help falling in love*, tutte le altre sono state pubblicate principalmente tra il 2015 e 2021, con solo *Più bella cosa* di Eros Ramazzotti pubblicata nel 1996. Questo testimonia il fatto che gli utenti del dataset sono per lo più giovani i cui interessi vanno per lo più al passo con i tempi, tuttavia tra gli interessi musicali degli utenti restano comunque i grandi classici.

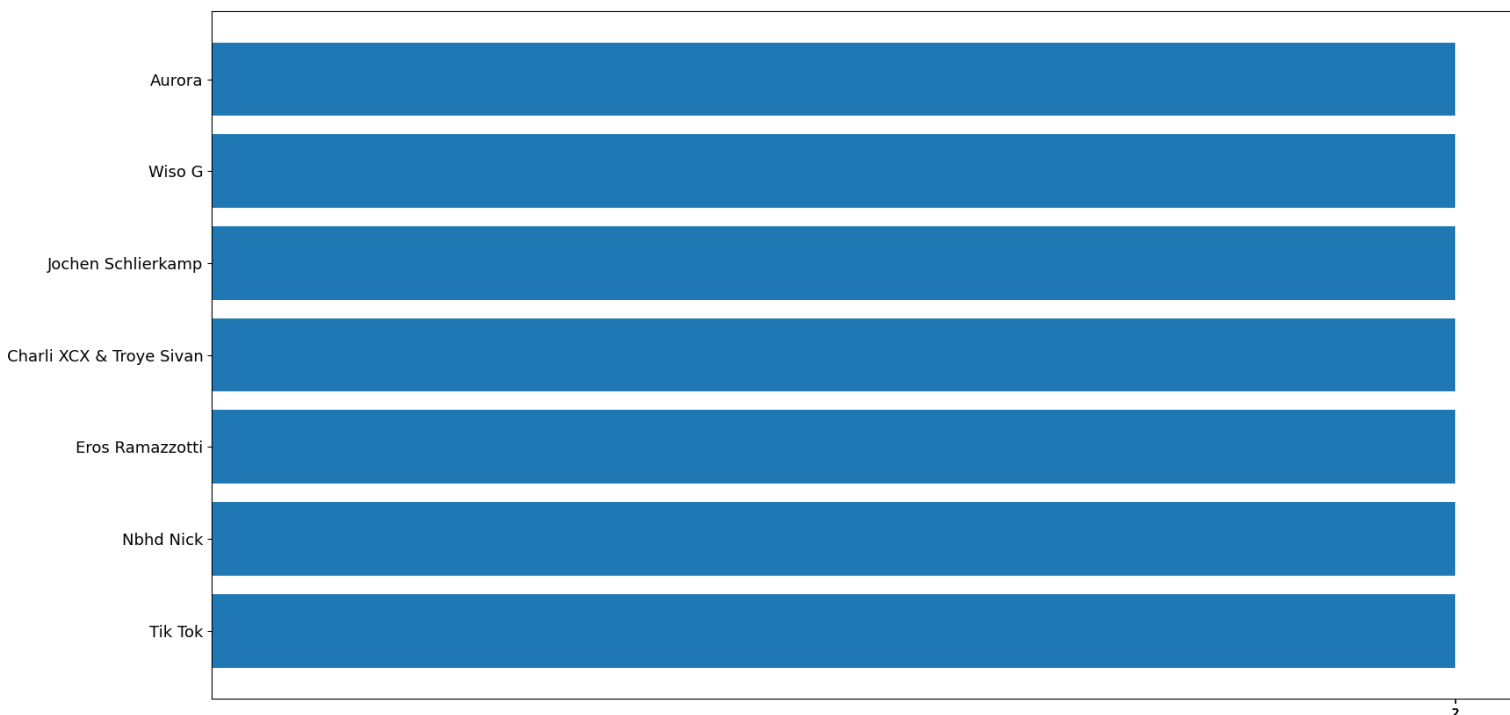


Figura 4.5: Canzoni più ascoltate dagli utenti

Una cosa che si può notare dagli utenti di Instagram è che nonostante la canzone di Elvis Presley appaia più volte, non è detto che sia stata cantata da lui quella pubblicata in foto, sono infatti presenti più e più cover di una canzone. Le cover sono un ottimo modo per suscitare interesse ai più giovani verso i grandi classici della musica. Gli artisti maggiormente condivisi sono quelli presenti nel grafico. Onore all'Italia che si ritrova Eros Ramazzotti grazie alle cover in italiano e spagnolo di *La Cosa Più Bella*

### 4.1.3 Grafico sugli hashtags

In questa sezione verranno mostrati quali sono gli hashtags più utilizzati dagli utenti del dataset.

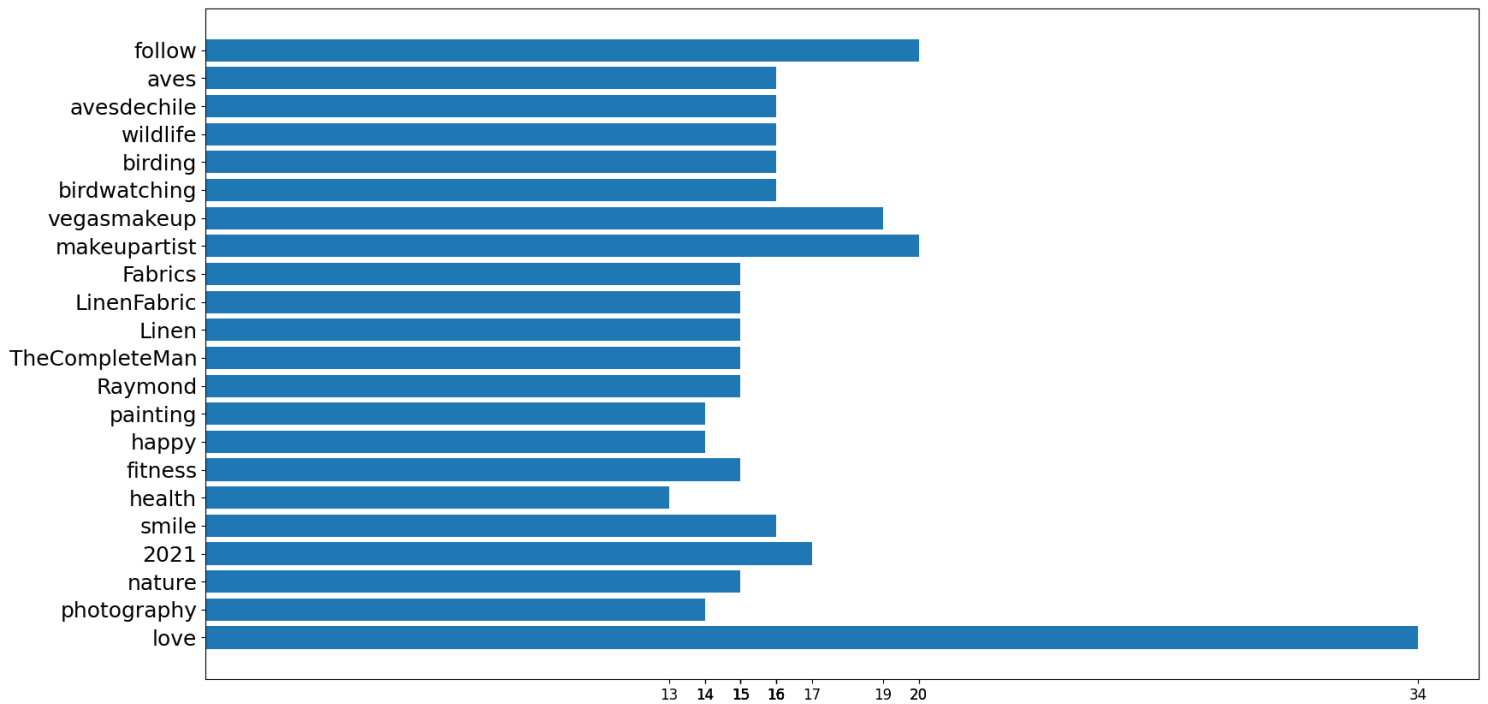


Figura 4.6: Frequenza degli hashtags nella didascalia

Si può notare che il lavoro di makeupartist sia quello più sponsorizzato su instagram. Le foto raffiguranti momenti di amore sono quelle più gettonate come si può evincere dall'hashtag *love* presente in 34 foto. Come ci si aspetterebbe da un social come instagram ci sono richieste di follow (20), foto di fitness, di vestiti, personaggi del momento o media del momento.

#### 4.1.4 Grafici sulle date

In questa sezione verranno mostrati quanti utenti hanno pubblicato date sul proprio profilo e quanto sono potenzialmente utilizzabili le informazioni mostrate per operazioni quali il social engineering

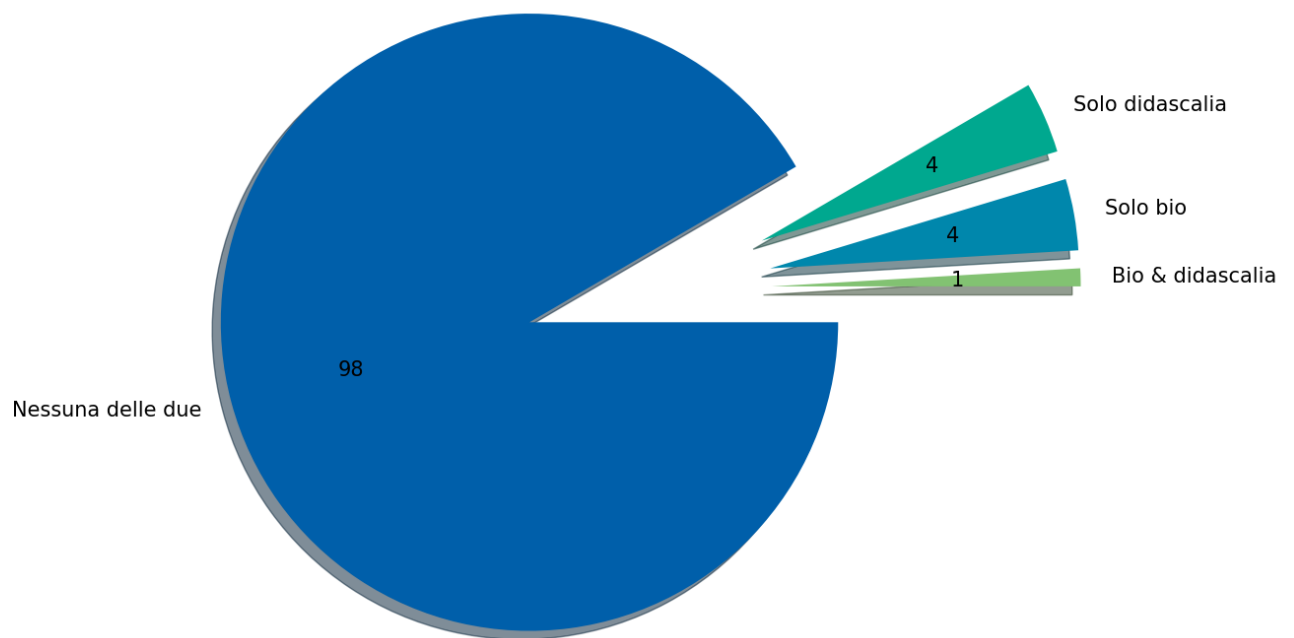


Figura 4.7: Date pubblicate dagli utenti

Su 108 utenti, la stragrande maggioranza non ha pubblicato date (98 utenti) negli ultimi 15 post o nella propria biografia, mentre 4 utenti hanno pubblicato date nella didascalia o nella biografia e solo 1 utente ha pubblicato date sia nel feed che nella bio.



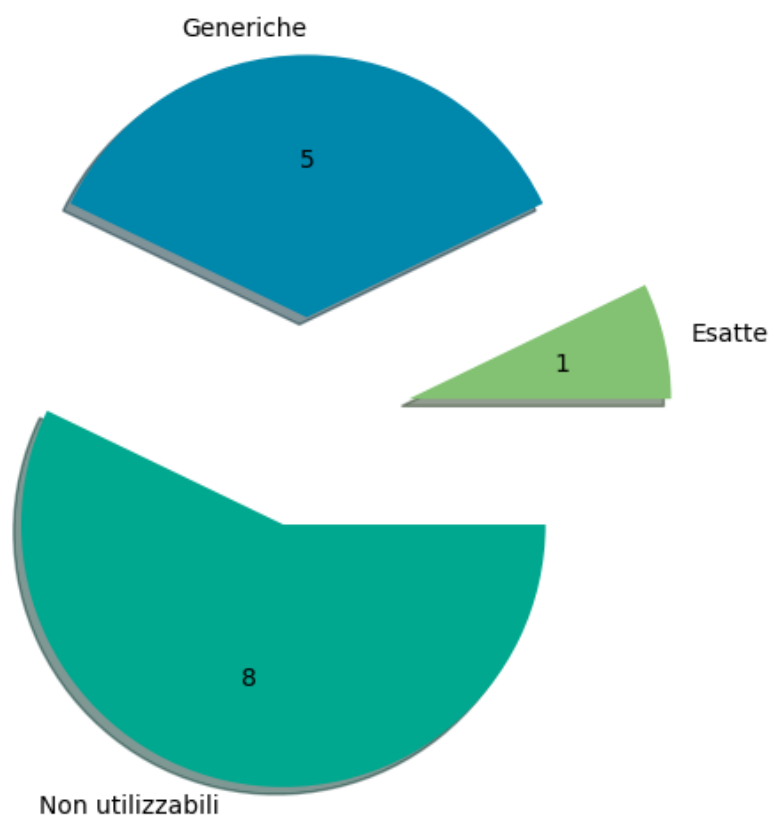


Figura 4.8: Tipologie di date presenti nella bio

Nella bio, è stata pubblicata una sola data completa con giorno, mese e anno, 5 date sono state pubblicate non complete e 8 date sono indicazioni temporali riconosciute come *now*, *tomorrow* e *altro*. E' possibile che le *date generiche* e *non utilizzabili* diventino più precise con un'analisi contestuale.

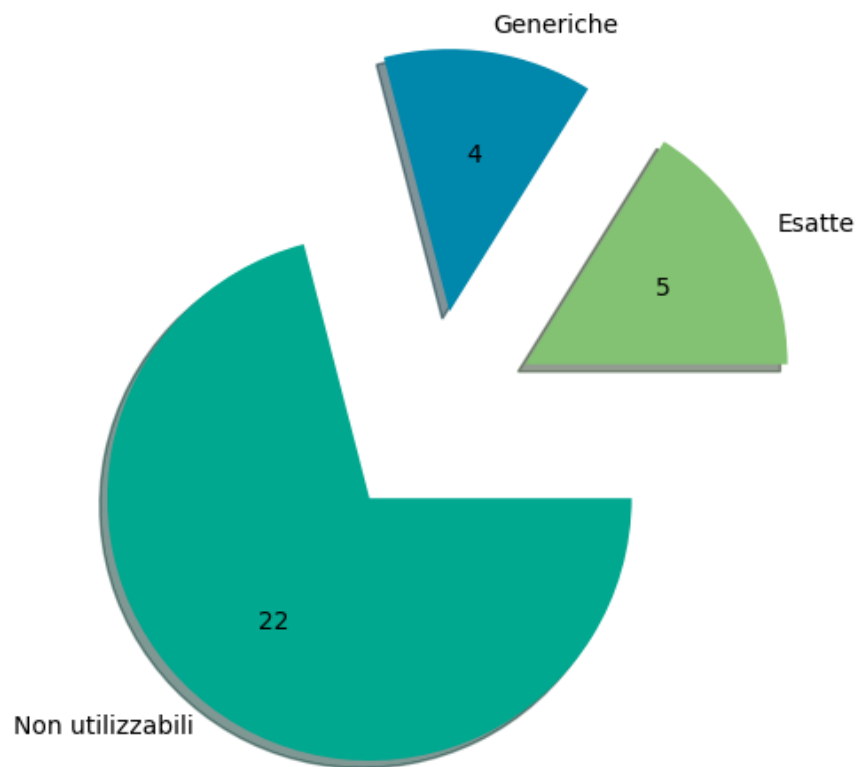


Figura 4.9: Tipologie di date presenti nella didascalia

Sono 5 le date esatte nelle didascalie delle foto, utilizzabili per identificare univocamente un avvenimento, mentre generiche sono 4 e non utilizzabili sono 22. E' possibile che come le date nelle biografie, anche qui *generiche* e non *utilizzabili* con analisi contestuale possano fornire informazioni precise.

### 4.1.5 Grafico sui numeri

In questa sezione si sono osservati i numeri presenti nella bio e si è analizzato quali di questi potesse essere utile per fare social engineering.

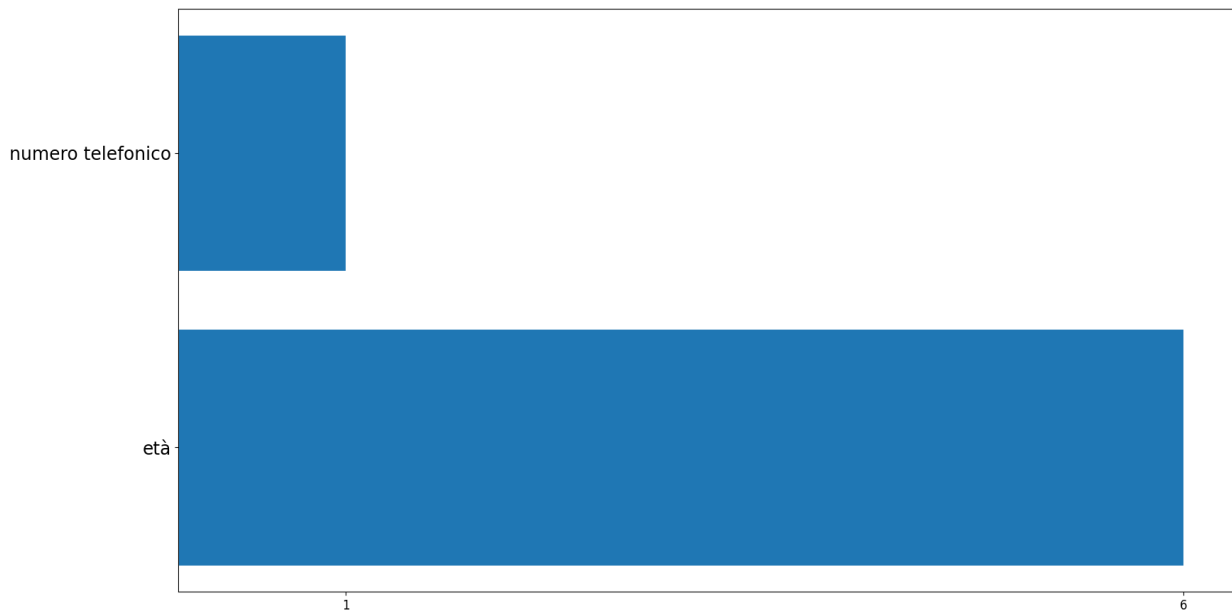


Figura 4.10: Nomi inseriti nella propria bio

Su 108 utenti si è trovato 1 numero telefonico (di un fotografo) e 6 utenti che hanno pubblicato l'età nella bio.

### 4.1.6 Nomi pubblicati

In questa sezione verrà mostrato quanti utenti hanno inserito il proprio nome all'interno della bio di Instagram. Per fare questo confronto si è usato il file *Nomi.csv* che è stato impiegato per ricercare gli account sul social e prendendo come assodato che i nomi di tutti gli utenti siano presenti nel file. Si è confrontato con *fuzzywuzzy* se il nome che l'utente ha inserito nella bio figurava o fosse simile a uno di quelli in *Nomi.csv*.

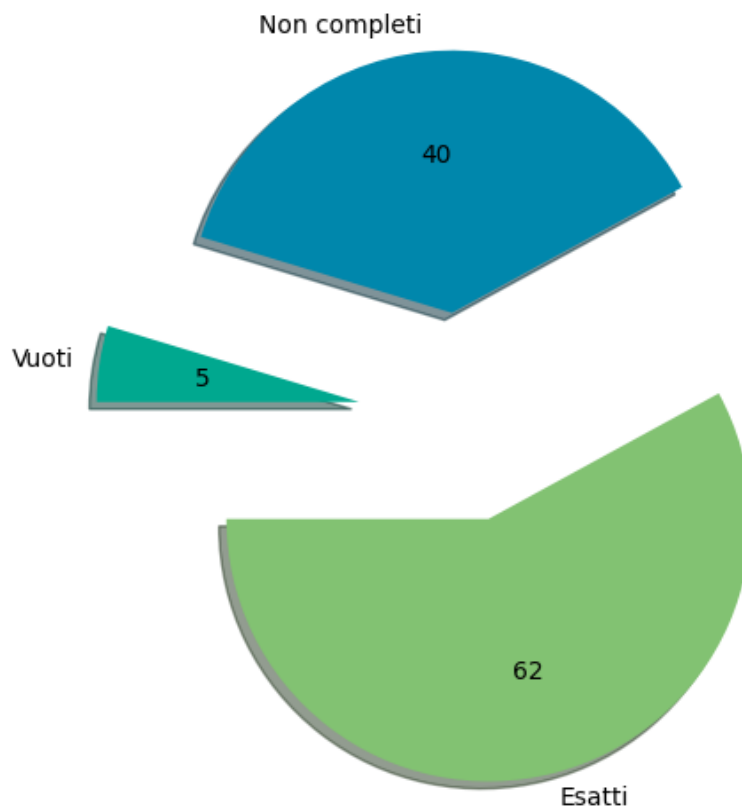


Figura 4.11: Nomi inseriti nella propria bio

Il risultato dell'esperimento è che 62 utenti hanno inserito il proprio nome e cognome nella bio, 40 hanno inserito abbreviazioni del proprio nome o cognome, o soltanto uno dei due, solo 5 utenti non hanno inserito nessun nome nella propria bio.

#### 4.1.7 Grafico sulle locations

In questa sezione verrà mostrato quanto spesso gli utenti pubblicino la propria posizione attraverso la funzione di geotagging e in seguito come questa funzione, assieme ad altre informazioni possono essere utilizzate per scopi malevoli.

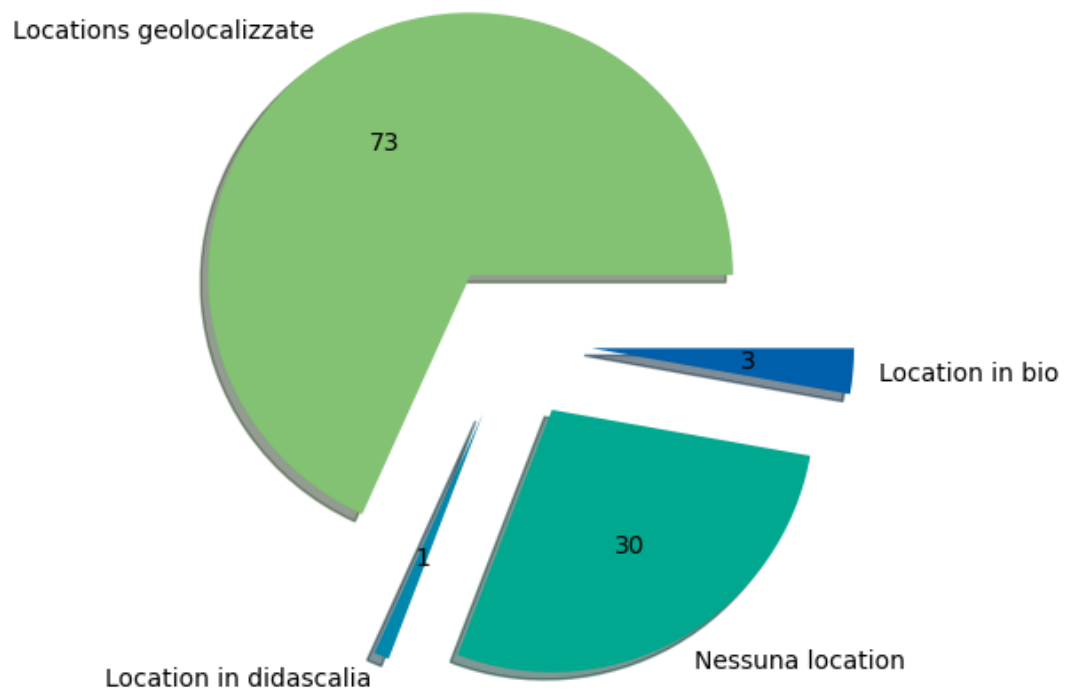


Figura 4.12: Utenti che utilizzano la funzione di geotagging

In foto viene mostrato che 73 utenti su 108 hanno impostato la funzione di geotagging nel pubblicare foto, 3 utenti hanno indicato dove si trovano nella biografia e un solo utente ha descritto dove si trovasse nel momento in cui ha scattato una foto. In totale, 77 utenti hanno dato indicazione del luogo contro i 30 che non hanno mai fornito indicazioni su dove si trovassero.

# Capitolo 5

## Cosa fare con i dati

In questo capitolo verrà mostrato come è stato possibile recuperare un probabile codice fiscale di uno degli utenti.

### 5.1 Codice Fiscale

E' stato scelto come primo utente, un utente randomico tale *Daniel Brassi*, brasiliano. E' stato difficile recuperare il suo codice fiscale, tale *CPF*, in quanto si sarebbe dovuto imparare un nuovo sistema giuridico e non conoscendo il portoghese sarebbe stato più complicato del necessario. Per questo motivo si è scelto di prendere un utente italiano dal dataset, tale *danidanycabrass*. Sapendo come funziona il codice fiscale in Italia e come calcolarlo bisogna trovare:

- Nome e cognome
- Genere
- Luogo di nascita
- Data di nascita

### 5.1.1 Nome e cognome

Il nome e cognome sono rintracciabili all'interno del file *dataset\_completo.json*:

Daniele Cabras

```
    "danidanycabrass": {  
      "index": 16,  
      "username": "danidanycabrass",  
      "info_name": "Daniele Cabras",
```

Figura 5.1: Nome di danidanycabrass

### 5.1.2 Genere

Dal nome si può risalire al genere: Maschio

### 5.1.3 Luogo di nascita

L'utente in questione ha pubblicato più post con geolocalizzazione attiva che lo posizionavano attorno a Domus de Maria, Sardegna. Si è ipotizzato quindi che l'utente possa essere di questa città viste l'alto numero di post. Da qui si è scoperto che Domus de Maria è provincia di Sardegna Sud dal 2014, ex provincia di Cagliari. Usando Google, il posto più vicino in Sardegna e quello più comune per partorire è l'ospedale di Cagliari. Con abbastanza sicurezza si può ipotizzare che quindi Daniele Cabras sia nato a Cagliari(CA).

```

"info_locations": [
  {
    "location": "Tuerredda",
    "address_json": {
      "street_address": "tuerredda SP 71 Chia Domus de Maria (CA)",
      "zip_code": "",
      "city_name": "Domus de Maria",
      "region_name": "",
      "country_code": "IT",
      "exact_city_match": false,
      "exact_region_match": false,
      "exact_country_match": false
    },
    "occurrences": 5
  },

```

Figura 5.2: Luogo di nascita di Daniele Cabras

#### 5.1.4 Data di nascita

Negli ultimi 15 post non si è riuscito a trovare nessun dato riguardante il giorno di nascita. Quindi si è dovuto modificare il dataset mettendosi nei panni di un utente malevole che vuole rubare l'identità dell'utente e ha quindi scaricato l'intero profilo. Usando nlp core, si è trovato un numero 26 pubblicato come didascalia. Ipotizzando fosse l'età e vedendo la data di pubblicazione della foto si è potuto risalire alla data di nascita: 4 ottobre 1994.






Figura 5.3: Luogo di nascita di Daniele Cabras

### 5.1.5 Risultato

Utilizzando le informazioni recuperate e un calcolatore di codice fiscale online si è ottenuto questo codice fiscale:



## Calcolo Codice Fiscale

Calcola il codice fiscale online

**CODICE FISCALE**

CBRDNL94R04B354I

**COGNOME**

CABRAS

**NOME**

DANIELE

**SESSO** M ▾

**LUOGO DI NASCITA**

CAGLIARI

**PROVINCIA (SIGLA)**

CA

**DATA DI NASCITA** 04 ▾ 10 ▾ 1994 ▾

Figura 5.4: Codice Fiscale di Daniele Cabras

# Capitolo 6

## Futuri Sviluppi

Per motivi tempistici ci si è concentrati di più su tematiche di sensibilizzazione degli utenti sul social engineering.

Possibili sviluppi futuri potrebbero riguardare:

- *Pubblicità mirata:* Avendo gli interessi degli utenti si potrebbero clusterizzare questi ultimi e capire quali elementi sarebbero più felici di acquistare contattando i diretti interessati.
- *Aumentare Dataset:* Ampliare il dataset aggiungendo altri utenti è sempre utile. In fase di creazione, è stata questa la fase più difficile che per motivi di tempo non ha permesso di aggiungere altri utenti.
- *Generatore automatico di codice fiscale:* Creare un generatore automatico di codice fiscale con le informazioni del dataset

# Indice

<b>1</b>	<b>Related work</b>	<b>2</b>
<b>2</b>	<b>Tencologie utilizzate</b>	<b>4</b>
2.1	Social Mapper . . . . .	4
2.1.1	Funzionalità di crawling aggiunte e migliorate . . . . .	6
2.1.2	Librerie utilizzate . . . . .	7
<b>3</b>	<b>Dataset</b>	<b>10</b>
<b>4</b>	<b>Grafici sui dati</b>	<b>16</b>
4.1	Grafici . . . . .	16
4.1.1	Dati semplici . . . . .	16
4.1.2	Grafici sulle canzoni . . . . .	18
4.1.3	Grafico sugli hashtags . . . . .	21
4.1.4	Grafici sulle date . . . . .	22
4.1.5	Grafico sui numeri . . . . .	26
4.1.6	Nomi pubblicati . . . . .	26
4.1.7	Grafico sulle locations . . . . .	27
<b>5</b>	<b>Cosa fare con i dati</b>	<b>29</b>
5.1	Codice Fiscale . . . . .	29
5.1.1	Nome e cognome . . . . .	30
5.1.2	Genere . . . . .	30
5.1.3	Luogo di nascita . . . . .	30
5.1.4	Data di nascita . . . . .	31

5.1.5 Risultato . . . . .	32
<b>6 Futuri Sviluppi</b>	<b>34</b>
<b>Bibliografia</b>	<b>37</b>

# Bibliografia

- [1] *YOLOv3 Object Detection with OpenCV*. URL: <https://github.com/yash42828/YOLO-object-detection-with-OpenCV>.
- [2] *Web Scraping Instagram with Selenium Python*. URL: <https://medium.com/analytics-vidhya/web-scraping-instagram-with-selenium-python-b8e77af32ad4>.
- [3] *5 Tips For Web Scraping Without Getting Blocked or Blacklisted*. URL: <https://www.scraperaapi.com/blog/5-tips-for-web-scraping/>.
- [4] *Survey on Nlp*. URL: <https://blog.aureusanalytics.com/blog/5-natural-language-processing-techniques-for-extracting-information>.
- [5] *An Exercise In Identifying Fake Instagram Profiles With TensorFlow*. URL: <https://medium.com/@karnsaheb/ig-profile-classification-914d3352f8ec>.
- [6] *Overview CoreNLP*. URL: <https://stanfordnlp.github.io/CoreNLP/>.
- [7] *Yolov5*. URL: <https://github.com/ultralytics/yolov5>.
- [8] *Instalooter home*. URL: <https://instalooter.readthedocs.io/en/latest/>.
- [9] *moviepy*. URL: <https://zulko.github.io/moviepy/>.
- [10] *fuzzywuzzy*. URL: <https://pypi.org/project/fuzzywuzzy/>.