Post Scriptum

Un'analisi linguistica del quotidiano online Il Post in un'applicazione Flask

Tommaso Barbato
tommaso.barbato@studio.unibo.it

Abstract

Questo progetto si propone di raccogliere i titoli degli articoli del giornale italiano online Il Post negli anni dal 2018 al 2023 via scraping, effettuare alcune analisi linguistiche di stampo statistico con l'aiuto di tool di natural language processing, visualizzare questi dati in grafici e infine presentarli in una web app usando Flask.

1 Introduzione

L'idea del progetto nasce dal tentativo di impratichirsi su più tecnologie possibili di quelle trattate a lezione, di modo da avere una visione ampia dei vari strumenti e maturarne una competenza, seppur superficiale, su ciascuno.

A tale scopo, abbiamo cominciando decidendo di non usare nessuno dei dataset già disponibili come base per un qualche tipo di analisi, ma di costruirlo autonomamente da zero sfruttando la tecnica del *web scraping*. La nostra attenzione si è rivolta, dopo una breve analisi dei possibili target, verso i titoli di giornale, sui quali sarebbe stata possibile compiere un'analisi quantitativa delle parole utilizzate, impiegando strumenti di *natural language processing*, un'altra delle tecnologie incontrate su cui fare pratica.

Da qui, il resto del progetto sarebbe conseguito abbastanza naturalmente: con la mole di dati quantitativi disponibili, il passo successivo più logico è stato realizzare grafici per rappresentare facilmente le informazioni statistiche ottenute in forma visuale.

Abbiamo finire deciso di esibire dati e grafici con un piccolo sito web, ritenendo che questa fosse la soluzione più efficiente per presentare ordinatamente le informazioni raccolte e che al contempo offriva l'opportunità di sperimentare il framework *Flask* per la creazione di web app.

2 Scraping

Identificare dove e come poter reperire un numero sufficientemente ampio di titoli di giornale attraverso il web scraping è stata la parte più complessa del progetto. La maggior parte dei giornali online utilizza una struttura web complessa, che non permette semplicemente di navigare all'indietro per recuperare, ad esempio, tutte le notizie di un determinato anno. L'accesso agli archivi, inoltre, è spesso a pagamento e nella maggior parte dei casi ostile a tecniche di scraping non sofisticate.

Dopo alcuni giorni di ricerca, tuttavia, abbiamo notato che Il Post[1], pur organizzando la propria homepage in maniera similmente problematica ad altri giornali online, disponeva di sezioni tematiche dove gli articoli erano paginati diversamente, in ordine temporale uno di seguito all'altro, seguendo una struttura HTML razionale che poteva essere sfruttata dallo scraper. Abbiamo pertanto deciso di orientarci su di esso, nello specifico sulla sezione tematica relativa all'Italia[2] in quanto quella più ricca di articoli, che arrivavano fino al 2017.

Sfruttando quindi le librerie standard per questa operazione, *requests* e *Beautiful Soup*, abbiamo identificato l'elemento HTML contenitore principale e iniziato lo scraping con le seguenti accortezze:

- per costruire un dataset il più coerente possibile abbiamo deciso di evitare gli anni non completi; pertanto abbiamo saltato tutti gli articoli del 2024, ancora in corso, e quelli del 2017, che si interrompevano ad un certo punto raggiunto il limite massimo delle pagine mostrate dal giornale, 500

- la data dell'articolo era scritta in termini rel-

[1] https://www.ilpost.it
[2] https://www.ilpost.it/italia

ativi (es. ”2 giorni fa”) per gli articoli più recenti, rendendo necessario o un calcolo complesso o di navigare nella pagina di ogni singolo articolo completo per estrarre da lì la data integrale. Abbiamo tuttavia scoperto che l'URL dell'articolo completo, invece, conteneva la data espressa in termini assoluti[3] e quindi, usando una regex, lo abbiamo estratto da lì

- abbiamo aggiunto un piccolo sleep timer alla fine di ogni scrape di pagina per evitare di tempestare il server di richieste e rischiare un potenziale blocco

Alla fine del processo, il nostro dataset per gli anni 2018-2023 è stato salvato in formato JSON, pronto per essere analizzato.

3 Analisi NLP

La decisione su che tipo di analisi linguistica effettuare si è fondata principalmente sulle capacità e la facilità d'uso dei modelli di *natural language processing* per la lingua italiana. Dopo aver brevemente esplorato gli strumenti a nostra disposizione, abbiamo deciso di utilizzare la libreria *spaCy*[4] in quanto, dopo alcuni test, più efficiente e pratica.

Nello specifico, le due funzioni su cui abbiamo basato la costruzione dell'analisi quantitativa sono state la *lemmatizzazione*, ovvero la riduzione della forma flessa di una parola alla sua forma canonica (es. ”andando” $\rightarrow$ ”andare”, ”dottoresse” $\rightarrow$ ”dottore”) e il *POS tagging*, ovvero l'assegnazione di categorie grammaticali a ciascuna parola (es. ”notizie” $\rightarrow$ sostantivo, ”indagano” $\rightarrow$ verbo).

Sfruttando queste due funzioni, siamo andati ad analizzare l'intero dataset costruendo le seguenti metriche:

- una lista globale di tutte le parole usate ordinata per numero di occorrenze, ottenuta iterando ogni token nel dataset e lemmatizzandolo, escludendo i segni di interpunzione

- una lista globale di sostantivi, verbi e nomi propri, ciascuna ordinata per numero di occorrenze e lemmatizzata, ottenuta sfruttando il *POS tagging*

- una lista per ciascuno degli anni del dataset (2018-2023) dei sostantivi usati, ordinata per numero di occorrenze, ottenuta allo stesso modo con l'aggiunta della divisione per anno sfruttando le date salvate durante lo scraping

- una lista, creata dopo aver creato le prime 3, del numero di occorrenze di alcuni sostantivi (lemmatizzati) selezionati tra i più comuni nel corso degli anni, per vedere in che modo mutava il loro uso nel tempo

Ciascuna di queste liste è stata poi salvata in un file JSON distinto.

3.1 Pulizia manuale dei dati

Nonostante l'ottimo lavoro che il modello NLP di spaCy ha fatto senza necessità di particolare tweaking, i dati generati dall'analisi risultavano un po' sporchi, con alcune parole non perfettamente raggruppate, e talvolta poco interessanti, in quanto piazzavano termini semanticamente irrilevanti in cima alle classifiche. Ad esempio la parola ”dato”, nonostante fosse il secondo sostantivo più usato in tutto il dataset, non dava nessuna indicazione tematica interessante per via della sua genericità (altri esempi: ”caso”, ”anno”, ”lunedì”, ecc...).

Per questo motivo, abbiamo deciso di non usare i dati grezzi per le fasi successive del progetto ma di ripulirli manualmente in modo da far emergere dei termini più interessanti ed eliminare quelli più trascurabili, pur rispettando fedelmente la classifica delle occorrenze. Dalla lista dei nomi propri, troppo casuale per avere qualche utilità statistica, abbiamo estratto solo i nomi delle città italiane, costruendo una lista di quelle più menzionate.

Abbiamo pertanto creato due nuovi JSON, identificati dal suffisso *clean*, per raccogliere i dati manualmente rifiniti che avremmo usato al posto di quelli grezzi.

4 Plotting

Con le 4 liste generate dall'analisi quantitativa alla mano, lo step successivo è stato ragionevolmente di rappresentare questi dati in grafici, sfruttando la classica libreria *Matplotlib*.

Il processo è stato piuttosto lineare: per i dati annualizzati abbiamo generato un grafico a barre per ciascun anno. Lo stesso abbiamo fatto per ogni POS tag (sostantivi, verbi e nomi propri divenuti città). Per la lista parole seguite nel tempo abbiamo invece usato un grafico a linee che

[3] Ad esempio: https://www.ilpost.it/2021/02/13/giuramento-governo-draghi-diretta-foto/ da cui si poteva estrarre la data 2021-02-13.

[4] https://spacy.io/

mostrasse facilmente l'oscillazione delle menzioni negli anni.

L'ultimo gruppo di dati a nostra disposizione era quello relativo alla lista globale di tutte le parole: per sfruttarlo abbiamo deciso di verificare che la legge di Zipf[5], secondo la quale la frequenza statistica delle parole in ogni corpus di testi segue la curva del decadimento esponenziale, fosse rispettata anche in questo caso, usando un grafico a linee.

Tutti i grafici generati sono poi stati esportati come immagini PNG, pronte per essere incorporate in un sito web.

5 Web application

L'ultimo passaggio del progetto è stato quello di presentare dati e grafici in una forma il più intuitiva possibile. Abbiamo subito considerato un sito web statico come il candidato migliore per questo scopo, in quanto ci avrebbe permesso di personalizzarne l'estetica a piacimento e di accedervi facilmente con un browser.

Tuttavia, esplorando le tecnologie disponibili nel contesto Python, siamo stati attirati dal framework *Flask*, che permetteva di creare web application dinamiche, anche semplicissime, con poche righe di codice. Interessati dalla possibilità di apprendere le basi di questo software, abbiamo pertanto scelto di sfruttare alcune delle funzionalità dinamiche del framework, come il *routing*, il *templating* e i *query arguments*, per realizzare qualche funzione aggiuntiva invece di un ipertesto base.

Per quanto riguarda il lato estetico, ci siamo avvalsi del framework CSS Bootstrap, in modo da avere elementi grafici gradevoli già pronti senza dover scrivere ulteriore codice e soprattutto avere accesso al sistema di layout della libreria per rendere le pagine ben strutturate e consultabili sia da desktop che da telefono con pochissime righe di HTML.

6 Conclusioni

Il progetto ci ha permesso di approcciarci ad un buon numero di tecnologie molto diffuse nell'ecosistema di Python e di comporre, con relativamente poco sforzo, un processo fondazionale di costruzione, analisi e presentazione di dati statistici, a cui senza dubbio attingere nuovamente nel futuro come modello iniziale per task simili.

Le maggiori difficoltà incontrate sono state primariamente nella definizione dello scope del progetto, e nella comprensione delle possibilità a nostra disposizione per trattare i dati raccolti. Sfruttando la dinamicità del linguaggio e i suoi strumenti di prototipazione veloce (come la standard shell e i Jupyter notebook), abbiamo però potuto validare o cestinare idee rapidamente, indirizzando il progetto con un misto di pratica e progettazione che ci ha permesso di iterare lentamente su visioni di base fino a perfezionarle.

References

Beautiful Soup documentation, https://www.crummy.com/software/BeautifulSoup/bs4/doc

spaCy documentation, https://spacy.io/usage/linguistic-features

Matplotlib documentation, https://matplotlib.org/stable/users/index.html

Flask documentation, https://flask.palletsprojects.com/en/3.0.x

Bootrap documentation, https://getbootstrap.com/docs/5.3/

[5] https://en.wikipedia.org/wiki/Zipf%27s_law