UNIVERSITÁ DEGLI STUDI DI MILANO-BICOCCA

BOOKING & TRIVAGO SCRAPING – ANALISI DELLE RECENSIONI, VOTI E PREZZI

Autori:

Luca Gabellini 777786
Matteo Provasi 782922
Lorenzo Vita 784681

Sommario

Introduzione e obiettivi	2
Scraping	2
Selenium webdriver	2
Scraping recensioni Booking	3
Scraping prezzi Booking	5
Scraping Trivago	5
Sharding	6
La nostra configurazione	7
Matching dei dataset	9
Analisi e risultati	10
Analisi di Booking	10
Analisi di Trivago	11
Analisi del matched dataset	12
Conclusioni	13

Introduzione e obiettivi

Booking.com è un sito di viaggi di origine olandese, che aggrega tariffe di strutture alberghiere provenienti da fonti web differenti proponendo all'utente una vasta gamma di offerte. Viene classificato come OTA (Online Travel Agency), che riprende il vecchio modello delle agenzie turistiche adattandolo al web 2.0. Dal 2005 è di proprietà della compagnia americana Booking Holdings, mentre il quartier generale dell'azienda ha sede ad Amsterdam.[1]

Il sito web presenta offerte relative a più di 1'534'024 strutture presenti in 226 nazioni differenti. Vengono rese disponibili all'utente svariate funzionalità, che permettono di fare confronti multicampo (in base al periodo, in base alla struttura, al prezzo etc.) per individuare la struttura preferita e prenotare l'offerta migliore. A differenza di altri siti simili, Booking.com risulta essere quello più completo a livello di informazioni ed anche il più comodo per effettuare scraping per via delle informazioni organizzate in maniera strutturata.

Trivago.com è un metamotore che colleziona le informazioni di hotel da più di 20 siti diversi, li confronta per offrire all'utente il prezzo migliore oltre che a riportare anche delle recensioni e giudizi tratti dai siti stessi. Con sede a Dusseldorf, Germania, Trivago è diventata negli ultimi anni una delle aziende con la maggiore crescita economica in Germania, con un fatturato annuo che nel 2017 ha toccato il miliardo di euro. [2]

Con oltre 5 milioni di accessi mensili al sito (da parte degli utenti italiani), Trivago è il primo concorrente di Booking.com in Italia. La differenza principale tra i due siti sta nel fatto che Booking è una OTA che svolge primariamente una funzione di aggregatore, mentre Trivago, come detto prima, è un metasearch che presenta una struttura molto simile a quella di tripadvisor. Entrambe ottengono la maggioranza dei propri guadagni grazie alle commissioni generate dal numero di prenotazioni on-site.

I dati di interesse sono stati ricavati in prima battuta da Booking.com, e poi integrati con quelli ottenuti da Trivago. Ciò ha permesso di ottenere informazioni il più possibile esaustive sulla singola struttura: recensioni (da Booking), prezzi, punteggi e numero di stelle (da Trivago).

Data la grande mole di dati ottenuta, è stato designato uno sharded cluster in modo tale da gestirla al meglio e supportare efficacemente le successive elaborazioni.

Gli obiettivi del nostro progetto sono:

- Evidenziare l'andamento del rapporto tra voto assegnato alla camera e costo dell'alloggio, condizionato al numero di stelle della struttura;
- Comprendere quali siano i termini/espressioni più utilizzati nelle recensioni, e come cambiano a seconda del sentiment della stessa;

[1] Bogenimaging.it: Booking, Trivago, Expedia
[2] Wikipedia: Trivago

- Verificare eventuali differenze tra i punteggi assegnati alla stessa struttura a seconda del sito.

Per tutti i punti elencati precedentemente verranno fornite analisi esplorative affiancate da indagini più approfondite.

Scraping

I dati di interesse sono stati ottenuti tramite scraping dal sito Booking.com.
L'algoritmo di scraping è stato creato in codice Python: in seguito sarà presentata l'architettura in maniera dettagliata. Sono state scaricate le recensioni (scritte in inglese) insieme ad una serie di metadati e informazioni legate all'hotel e al recensore, memorizzati in formato JSON e CSV.

Selenium webdriver

Selenium è un framework per testing di applicazioni web. Consente di scrivere test in linguaggi di programmazione popolari (C, Java, Python etc). È un software rilasciato da Apache: gli sviluppatori web possono scaricarlo ed utilizzarlo gratuitamente vista la sua natura open source.

Consente non solo di fare testing, ma anche di automatizzare task ripetitivi su applicazioni web: implementando uno script in uno dei linguaggi supportati, questo utilizzerà specifici comandi per emulare le azioni eseguibili dall'utente.

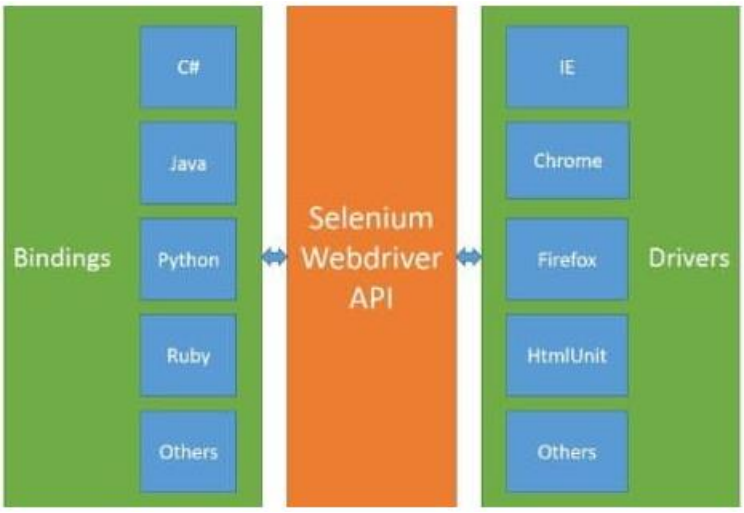

Fig.1: *Funzionamento di selenium webdriver*

Selenium webdriver, una delle componenti della famiglia selenium, si basa su 3 componenti architetturali principali (***Fig.1***):

- Lo script implementato in uno dei codici sopra elencati (Python nel nostro caso) interagisce con l'API di selenium webdriver;

- L'API, grazie ad un set di librerie comuni, interpreta e traduce lo script inviandolo al rispettivo driver;
- Il driver (Chrome driver nel nostro caso) riceve i comandi dal livello precedente eseguendo i task richiesti (cliccare su un link, passare alla pagina successiva, acquisire dati etc) sul browser.

Scraping recensioni Booking

I dati delle recensioni sono stati memorizzati all'interno di un file JSON, per via della sua struttura flessibile che ha permesso l'inserimento delle caratteristiche di ogni hotel e di ogni recensione in base alla loro presenza.

Per scaricare i dati è stato settato il *webdriver* di Chrome (inizialmente i primi tentativi di scraping sono stati realizzati mediante Mozilla Firefox che però è risultato sia più lento, sia eccessivamente dispendioso a livello di RAM utilizzata) con la modalità in incognito, scegliendo di non caricare le immagini. Il driver è stato indirizzato (tramite la funzione *get*) alla pagina dove venivano elencati gli hotel inglesi, che presenta un contatore per poter passare alla pagina successiva di hotel. Booking elenca un massimo di 30 hotel per pagina (per ogni città), salvo i casi in cui l'ultima pagina non comprenda abbastanza hotel.

Caratteristica peculiare su cui si è basato lo scraping è stata la struttura dell'url che poteva essere di due tipi:

- Tradizionale (per la prima pagina)
- Presenza di "*offset=30&*" alla fine dell'url per la seconda pagina

Il valore "*offset*" nell'url indica al sito da quale hotel visualizzare i risultati in pagina. In questo caso un valore di offset uguale a 30 imponeva al sito di visualizzare i risultati dal 31esimo hotel. È possibile modificare questo valore a piacimento, settandolo (ad esempio) a 0 per poter creare lo stesso risultato della prima pagina tradizionale ma con un url standardizzato rispetto alle altre.

Successivamente sono stati individuati gli elementi HTML contenenti il link delle recensioni per ogni hotel tramite il comando:

driver.find_elements_by_partial_link_text('See more reviews of ')

"*See more reviews of*" è un testo che compare per ogni hotel che, se selezionato, porta alla pagina delle recensioni di quello specifico hotel.

Inizialmente la selezione della pagina delle recensioni veniva effettuata tramite la funzione *click()* di Selenium, la quale, tuttavia, non attendeva il caricamento della pagina per passare alla riga di codice successiva portando ad un errore. È stata quindi utilizzata la funzione *get*, che risolve il problema descritto e grazie all'attributo *href* degli elementi trovati precedentemente rende possibile l'indirizzamento alla pagina desiderata.

L'individuazione degli elementi originariamente veniva effettuata tramite le funzioni di Selenium che estraevano l'elemento HTML direttamente all'interno del browser. Lo scraping tuttavia

rallentava esponenzialmente col passare del tempo: se per la prima pagina con 30 hotel il tempo impiegato era di circa 20 minuti, alla fine del 90esimo il tempo trascorso risultava di 3 ore e mezza.

Grazie alla funzione *BSoup(driver.page_source, 'html.parser')* della libreria bs4 è stato possibile scaricare direttamente il codice sorgente HTML della pagina del webdriver. Dato che gli elementi venivano individuati direttamente all'interno di un testo senza dover interagire con il browser, lo scraping si velocizzava notevolmente (30 hotel in 10 minuti senza rallentamenti successivi).

Le caratteristiche di ogni recensione e di ogni hotel sono state scaricate tramite la funzione *find* individuando la tag e l'attributo corrispondente dell'elemento HTML:

- Caratteristiche dell'hotel (Nome, Indirizzo, Numero di recensioni, Voto Totale, Voto Pulizia, Voto comodità, Voto location, Voto servizi, Voto staff, Voto rapporto qualità-prezzo, Voto Wi-Fi gratuito).
- Caratteristiche delle recensioni (Data della recensione, Nome utente, Nazionalità, Fascia d'età, Numero di giudizi precedenti dell'utente, Voto, Titolo, Recensione positiva, Recensione negativa, Data soggiorno, Tags).

Inoltre, per ogni hotel è stato creato un ID univoco formato da "Nome_Città".

All'interno di questo codice è stata utilizzata anche la funzione *try-except-pass* per gli elementi non presenti in tutte le recensioni, che porterebbero ad un errore e all'interruzione dello scraping.

Una volta scaricati i dati venivano fatti dei controlli di preprocessing per poter evitare che ci fossero hotel ripetuti o con nessuna recensione, in modo da poter correggere immediatamente l'errore e riscaricare la parte mancante.

Alla fine dello scraping abbiamo raggiunto un dataset di 2.4 GB.

Scraping prezzi Booking

Su un arco temporale di un anno, cambiando il check-in ogni 3 giorni, sono stati selezionati i prezzi di tutti gli hotel inglesi considerando i soggiorni di 2 notti. Questi sono stati salvati come dataset in formato csv, composto da 10 attributi (Hotel, City, Room, Max Guests, Price, Room reinforcements, Check-in Date, Booking in last 6 hours, Scraping Date, ID).

La selezione del check-in e check-out è resa possibile modificando l'url in cui viene indicato mese, giorno e anno dell'inizio e della fine del soggiorno. Individuando il numero dell'ultima pagina degli hotel per queste date e moltiplicandolo per il numero di strutture per ogni pagina (15), è stato possibile definire l'ultimo numero da inserire nel meccanismo "*offset=30&*", illustrato per lo scraping precedente.

Dopo aver estratto il codice sorgente HTML sono stati individuate e memorizzate, sempre tramite la funzione *find*, le informazioni utili (Nome hotel, Città, Numero di prenotazioni nelle ultime 6 ore, Tipo di camera, Numero massimo di persone per camera, Prezzo, Lista dei servizi inclusi nel prezzo).

Questo codice tuttavia presenta una criticità: il browser aumenta esponenzialmente l'utilizzo della RAM col passare del tempo, bloccando il processo. Per poter eludere questo problema è stato inserito un comando che, dopo ogni 150 hotel, chiudeva il browser e lo riapriva.

In conclusione, è stata svolta una pulizia del dataset ottenuto:

- La variabile Room reinforcements è stata suddivisa in 4 variabili binarie (Breakfast included, Dinner included, Free cancellation, No prepayment) che indicano la presenza o meno di un certo servizio nel prezzo.
- Gli hotel ripetuti con la stessa data di check-in sono stati eliminati, dato che differivano soltanto per un euro nel prezzo o per il numero di prenotazioni nelle ultime 6 ore.
- I record contenenti missing value nel prezzo sono stati rimossi, poiché rappresentavano soltanto il 5% del dataset.

Scraping Trivago

La struttura del codice per lo scraping dei prezzi su Trivago è uguale a quella usata per Booking, eccetto qualche piccola variazione per l'individuazione di determinati elementi. I risultati, tuttavia, sono riportati in maniera differente fra i due siti, da qui la necessità di un diverso approccio per ottenere i prezzi da Trivago.

Booking non mette restrizioni sul numero di hotel visualizzati ed il codice poteva essere lasciato andare senza interventi esterni. Trivago propone i risultati della ricerca in massimo 20 pagine e non è in alcun modo possibile aumentare questo limite; anche modificando l'url per inserire un numero di pagine maggiore la visualizzazione si ferma in ogni caso alla ventesima pagina. Questa limitazione riduce sensibilmente il numero di hotel di cui si riesce a ricavare il prezzo. Fortunatamente è possibile modificare tramite l'url alcuni parametri di ricerca: la soluzione trovata per massimizzare il numero di hotel è stata quella di effettuare 5 diversi scraping ognuno in corrispondenza del numero di stelle. Con questo metodo si è potuto aumentare di 5 volte il volume di dati ottenuto.

Lo scraping su Trivago è poi risultato molto più complicato di quello su Booking: oltre al fatto di non porre limitazioni sul numero di hotel visualizzati, Booking lasciava che il codice interagisse con il sito senza limitazioni, mentre su Trivago, appena iniziato lo scraping, veniva rilevato che un bot stava effettuando le ricerche. Il reCAPTCHA visualizzato doveva essere sbloccato su una nuova finestra, in quanto nella finestra di scraping, anche dopo aver spuntato la casella, la pagina non si ricaricava. Per risolvere questo problema è stato necessario fermare lo scraping, salvare i dati ottenuti, riabilitare l'accesso alla pagina e ricominciare con lo scraping nuovamente. Per semplificare il procedimento si è provato a rallentare la velocità dello scraping: mediante il codice *time.sleep()* si creava un'attesa di qualche secondo fra il caricamento di una pagina e quella successiva. Anche questa soluzione non ha portato miglioramenti in quanto il sito, anche se con un po' di ritardo, riusciva ad individuare il bot. Per completare lo scraping si è dovuto intervenire manualmente diverse volte per riavviare la procedura, combinato con la suddivisione degli hotel per stelle, ci sono voluti diversi giorni per ottenere i prezzi da Trivago.

Sharding

MongoDB (da "humongous", enorme) è un DBMS non relazionale, orientato ai documenti. Classificato come un database di tipo NoSQL, MongoDB si allontana dalla struttura tradizionale basata su tabelle dei database relazionali in favore di documenti in stile JSON con schema dinamico (MongoDB chiama il formato BSON), rendendo l'integrazione di dati di alcuni tipi di applicazioni più facile e veloce. [3]

MongoDB supporta lo scaling di tipo orizzontale tramite lo sharding. Questo metodo permette di distribuire i dati e il carico di lavoro su più server (teoricamente allocati su più macchine differenti), in modo da aumentare la capacità complessiva del sistema e renderlo più performante.

Uno sharded cluster è sostanzialmente composto da 3 componenti: [14]

- Shard: può essere o una singola istanza di MongoDB (mongod) oppure un intero Replica Set. Lo shard ha la funzione di memorizzare uno o più subset dei dati acquisiti. L'implementazione del replica set scongiura la perdita permanente di dati rispetto all'architettura basata su un singolo nodo.
- Config servers: i config servers memorizzano metadati e altre importanti informazioni sull'architettura del cluster.
- Mongos: L'istanza mongos agisce da router, facendo da interfaccia tra il client e il cluster.

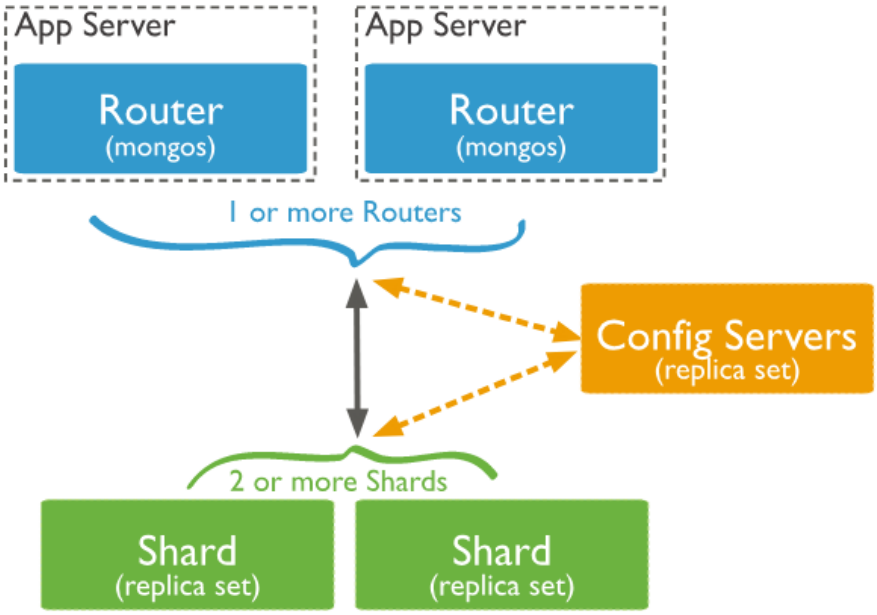

Fig.2: *Architettura di un tipico sharded cluster*

MongoDB partiziona la collezione basandosi sulla shard key, un campo immutabile e univoco presente in ogni documento. La chiave di shard viene scelta in fase di abilitazione dello sharding

[3] Wikipedia: MongoDB
[4] MongoDB.com: Sharding Manual

sulla collezione di interesse, e non potrà essere cambiata in seguito. La scelta della chiave condizionerà inevitabilmente la performance, l'efficienza e la scalabilità del cluster.

Mongo partiziona i dati shardati in chunks, dei subset della collezione originale. Questi vengono organizzati e spostati nei chunks grazie allo sharded cluster balancer.

Per poter gestire in maniera efficiente la grande mole di recensioni e dati ottenuti tramite scraping, si è scelto di implementare un'architettura di questo tipo, che verrà presentata più nel dettaglio nel successivo paragrafo.

La nostra configurazione

La nostra versione di MongoDB (3.4) vincola l'utente a inizializzare il config server come replica set. Nel nostro caso quindi il set si compone di 3 repliche, ovvero 3 istanze di *mongod*. Una volta completata la configurazione del server con il comando rs.initiate(), sono stati specificati i 2 nodi di shard, anche questi corrispondenti a 2 replica set, ciascuno composto di 3 istanze *mongod*. Una nuova istanza *mongos* viene poi collegata al *config server*; successivamente, con un mongo connesso al mongos, gli shard vengono aggiunti al cluster.

Il dataset principale contenente le recensioni di booking viene acquisito dal sistema con il comando *mongoimport*. Dopo aver creato un opportuno indice per supportare l'hashed sharding, il database e la collezione specificati con il precedente comando vengono shardati nel sistema. Dal check emerge che la collezione, formata da 6262 documenti (uno per ogni hotel scrapato), viene suddivisa in 58 chunks, equamente distribuiti nei due shards:

```
mongos> db.hotel.getShardDistribution()

Shard shard1.1 at shard1.1/localhost:27414,localhost:27415,localhost:27416
 data : 942.6MiB docs : 3131 chunks : 29
 estimated data per chunk : 32.5MiB
 estimated docs per chunk : 107

Shard shard1.2 at shard1.2/localhost:27417,localhost:27418,localhost:27419
 data : 920.18MiB docs : 3131 chunks : 29
 estimated data per chunk : 31.73MiB
 estimated docs per chunk : 107

Totals
 data : 1.81GiB docs : 6262 chunks : 58
 Shard shard1.1 contains 50.6% data, 50% docs in cluster, avg obj size on shard : 308KiB
 Shard shard1.2 contains 49.39% data, 50% docs in cluster, avg obj size on shard : 300KiB
```

Fig.3: *Sharding status intermedio*

In seguito si ripete lo stesso procedimento per i dataset utilizzati nella fase di analisi, in particolare per le collezioni "hotelfinal", "hotelprice" e "hotelfinal_filter", che sono tutte di dimensioni rilevanti.

La conformazione finale del cluster sarà la seguente:

```
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("5b9fd9689b11b337a527207d")
  }
  shards:
        {  "_id" : "shard1.1",  "host" : "shard1.1/localhost:27414,localhost:27415,localhost:27416",  "state" : 1 }
        {  "_id" : "shard1.2",  "host" : "shard1.2/localhost:27417,localhost:27418,localhost:27419",  "state" : 1 }
  active mongoses:
        "3.4.10" : 1
  autosplit:
        Currently enabled: yes
  balancer:
        Currently enabled:  yes
        Currently running:  no
NaN
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                79 : Success
                1 : Failed with error 'aborted', from shard1.2 to shard1.1
```

```
databases:
        {  "_id" : "england",  "primary" : "shard1.2",  "partitioned" : true }
                england.hotel
                        shard key: { "_id" : "hashed" }
                        unique: false
                        balancing: true
                        chunks:
                                shard1.1        29
                                shard1.2        29
                        too many chunks to print, use verbose if you want to force print
                england.hotelfinal
                        shard key: { "_id" : "hashed" }
                        unique: false
                        balancing: true
                        chunks:
                                shard1.1        30
                                shard1.2        30
                        too many chunks to print, use verbose if you want to force print
                england.hotelfinal_filter
                        shard key: { "_id" : "hashed" }
                        unique: false
                        balancing: true
                        chunks:
                                shard1.1        20
                                shard1.2        20
                        too many chunks to print, use verbose if you want to force print
                england.hotelprice
                        shard key: { "_id" : "hashed" }
                        unique: false
                        balancing: true
                        chunks:
                                shard1.1        29
                                shard1.2        29
                        too many chunks to print, use verbose if you want to force print
```

Figg. 4,5: *Sharding status finale*

L'implementazione dello sharding ha permesso di gestire in maniera ottimale la grande quantità di dati a nostra disposizione. Si è rivelato un ottimo strumento a supporto delle analisi con Python: il collegamento tra i due software è stato possibile grazie al package PyMongo, driver di Python per MongoDB.

Matching dei dataset

Volendo fare delle analisi incrociate fra i due siti, è stato necessario effettuare un matching fra i tre dataset. Si è iniziato confrontando gli ID del dataset in formato JSON di Booking con quello dei prezzi di Trivago. Come previsto il dataset di Booking conteneva quasi il doppio degli hotel di Trivago, 6262 contro 3424. Senza effettuare nessuna modifica sulle stringhe si otteneva una corrispondenza di 666 ID.

Per poter ottenere più corrispondenze si è iniziato a lavorare sui nomi delle città, modificando elementi ambigui come *St.* e *Saint* e la rimozione di trattini. Con queste modifiche si è passati da 418 a 442 città in comune su un massimo possibile di 510. Si è voluto quindi indagare sul perché alcune città non corrispondessero: Trivago è molto più preciso rispetto a Booking nell'individuazione delle località, ad esempio per Londra venivano riportati come città il nome del quartiere e non Londra stessa. La maggior parte degli ID non corrispondenti erano relativi a delle città molto piccole (definiti addirittura villaggi per via della bassa popolazione) che su Booking rientravano sotto la denominazione della città più grande nei paraggi. L'ultima differenza era nel fatto che su Trivago venivano anche trovate le città appartenenti all'Isola di Man, mentre queste su Booking erano completamente assenti.

L'ID è stato quindi aggiornato sostituendo il nome della città originale con il nome di matching nei casi in cui è stata necessaria una modifica.

Successivamente si è cercato di trovare delle corrispondenze tra hotel: in questa fase le modifiche sono state più profonde con la rimozione di tutti i caratteri speciali e di alcune parole comuni come *hotel* e *house*. Mediante questa pulizia si è potuto quasi raddoppiare il numero di hotel in comune passando da 683 a 1235.

Questo numero è comunque solamente un terzo degli hotel totali e si è quindi voluto indagare sommariamente sul perché ci fossero tutti questi matching mancati. Moltissimi nomi di hotel differivano in maniera sostanziale fra i due siti, il che lascia pensare a due strutture alberghiere diverse e quindi più ad una impossibilità oggettiva nel trovare una corrispondenza piuttosto che delle limitazioni nella procedura di matching.

Analisi e risultati

Analisi delle recensioni

Dai documenti nel file JSON sono state estratte le recensioni negative e positive. I file di testo hanno subìto un processo di tokenizzazione (tecnica mediante la quale da una frase si ottengono delle singole parole individuate tramite la separazione di uno spazio) propedeutico alla successiva creazione della word cloud. Si è reso poi necessario un processo di normalizzazione dei token ottenuti. Volendo analizzare le differenze fra i termini adottati a livello di stelle e recensioni positive e negative, sono state create un totale di 10 liste di token.

Dopo aver ricavato ogni parola sono state effettuate le seguenti modifiche:

- Le parole sono state rese tutte minuscole (dato che Python è case sensitive)
- Sono stati rimossi i segni di punteggiatura e caratteri speciali
- Un processo di lemming su nomi, aggettivi e verbi. Questo passaggio comporta l'estrazione della radice di una parola: in particolare si risolvono i problemi di singolare/plurale, desinenza verbale e aggettivi di comparazione.
- Le parole con meno di tre caratteri sono state rimosse
- Le parole trovate dopo questo processo sono state confrontate con una lista di stopwords, un elenco di parole molto frequenti, specialmente articoli e preposizioni, che non sono utili per ottenere delle informazioni dal testo tokenizzato.

Le parole così ottenute sono state memorizzate con le relative frequenze in dizionari.

Volendo cercare informazioni sul linguaggio utilizzato nelle recensioni, non sono state analizzate le singole parole, ma coppie di parole, dette bigrammi. Dopo aver trovato le coppie di parole, queste sono state ordinate in quanto due bigrammi con le stesse parole ma in ordine invertito erano considerate differenti ma per i fini delle nostre analisi sono da considerarsi uguali.

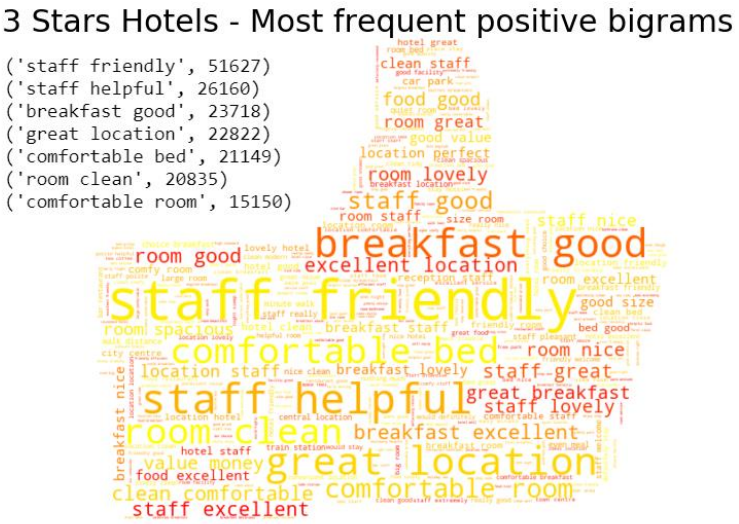

Fig.6: *Bigramma parole positive*

Per quanto riguarda i bigrammi positivi, non esiste una grande differenza a livello di stelle: in tutte le word cloud il bigramma più frequente è *staff friendly,* seguito da altri commenti positivi sulla colazione o la stanza.

Per quanto riguarda i bigrammi negativi si è ritenuto necessario effettuare un'ulteriore analisi in quanto utilizzando solo due parole non era possibile integrare anche la parola "no" o "not" anteposta al bigramma ottenuto: si è reso quindi necessario creare dei trigrammi.

In questo caso esiste una differenza netta fra i termini utilizzati nei commenti negativi, in particolare sono riportate di seguito le due word cloud per le recensioni degli hotel con una stella e quelli con cinque:

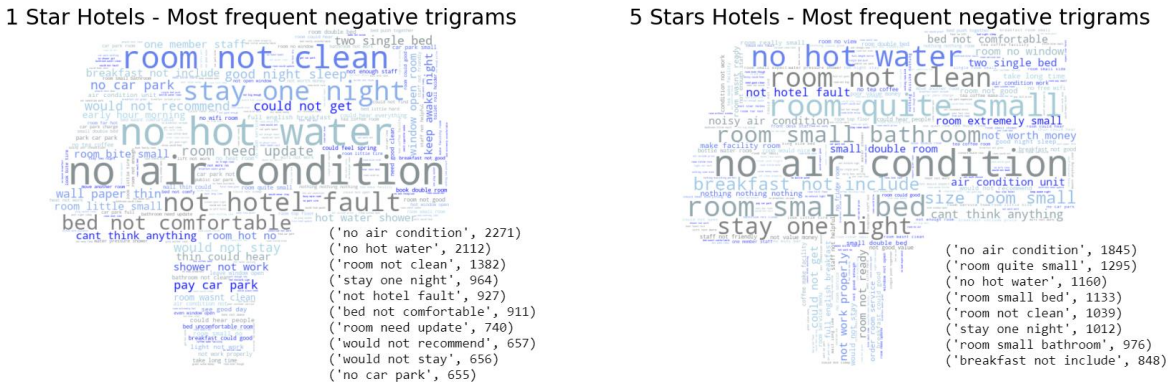

Fig.7: *Confronto trigrammi nelle recensioni di hotel a 1 e a 5 stelle*

Entrambe le word cloud hanno dei trigrammi in comune, come ad esempio la mancanza dell'aria condizionata, ma la differenza più importante si rileva a livello della parola "room": benché sia presente anche nelle recensioni negative degli hotel ad una stella, per quelli da cinque ci sono diversi trigrammi riguardanti la dimensione ristretta della camera. Da sottolineare infine come il termine "condition" era duplicato in quanto l'abbreviazione informale "con" era utilizzata quasi in ugual misura nelle recensioni, ma non essendo un termine propriamente corretto, il processo di lemming non riusciva a convertire il termine in quello originale.

Analisi di Trivago

A differenza del dataset precedente, il CSV di Trivago non contiene recensioni proprie dato che è soltanto un sito di comparazione.

Abbiamo deciso, pertanto, di analizzare quali possano essere le differenze nel rapporto prezzo-voto a seconda delle stelle dell'hotel. Dall'infografica successiva si può notare come all'aumentare delle stelle la distribuzione degli hotel passi da un range di valori più ampio sull'asse X (Voto) ad un intervallo più ampio sull'asse Y (Prezzo). Ciò sta a indicare che per gli hotel con poche stelle c'è alta variabilità sul voto e bassa variabilità sul prezzo. Per gli hotel di 4 o 5 stelle vale il contrario.

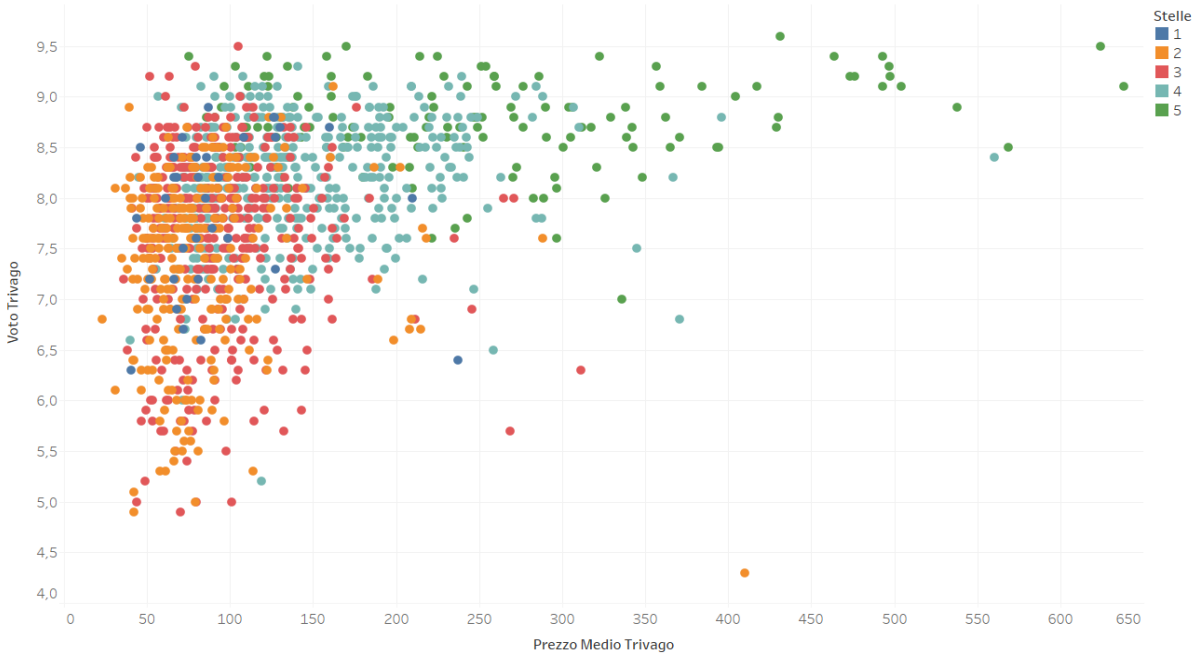

Fig.8: *Voto vs prezzo condizionato per n° stelle*

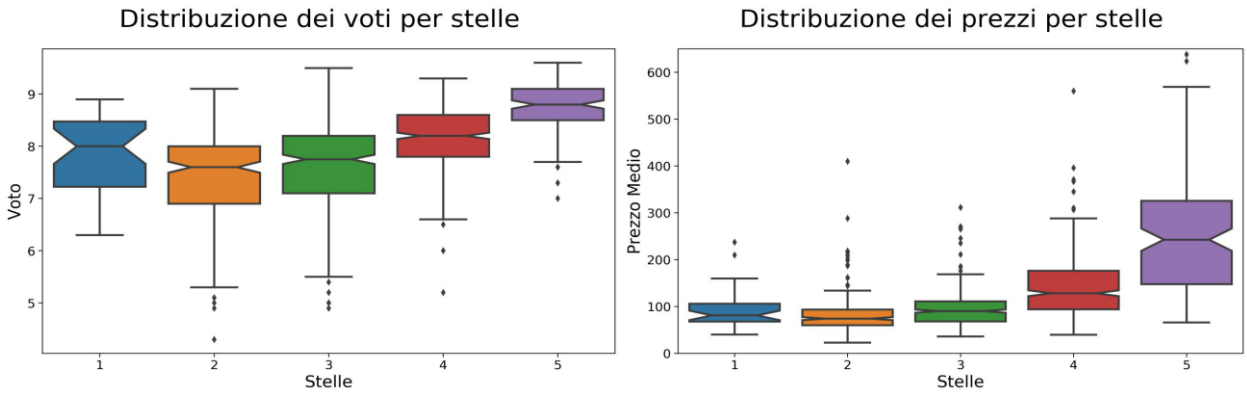

Fig 9: *Confronto stelle-voto per gli hotel di Trivago*

Dai boxplot si nota una tendenza: il rating aumenta all'aumentare del numero di stelle, ad eccezione degli hotel a una stella che hanno un voto medio paragonabile a quello degli hotel a 3 o 4 stelle.

Per quanto riguarda i prezzi l'andamento è simile al precedente. Spicca un'elevata variabilità dei prezzi relativi agli hotel a 5 stelle rispetto agli altri

Analisi del matched dataset

Una volta ottenuto il dataset con gli hotel in comune sono state effettuate delle analisi riguardanti la valutazione media della struttura alberghiera. Prima di parlare dei risultati è necessario fare alcune premesse sui sistemi di valutazione dei due siti. Booking non riporta nei dettagli come

calcola lo score di una struttura alberghiera, però è intuibile supporre che sia la media di diversi parametri (pulizia, posizione, servizi, staff…) calcolati a loro volta dalla recensione di un utente.

Su Trivago non è possibile scrivere direttamente delle recensioni in quanto è un sito di comparazione di hotel. Trivago prende recensioni e giudizi da diversi siti per creare il trivago Rating Index (tRI). La necessità di sviluppare questo indice nasce dal fatto che l'interazione di giudizi da fonti diverse fra di loro è problematica e non è possibile effettuare una semplice media. Come riportato nel sito lo score tiene in considerazione diverse variabili fra cui: il numero di recensioni della fonte, il numero di fonti, la data in cui è stato rilasciato il giudizio ed altri parametri.

Non è possibile, ovviamente, dire quale dei due giudizi sia il più preciso, in quanto non esiste un punto fisso da cui fare paragoni, ma mettendosi nell'ottica di un utente che vuole prenotare un hotel è interessante valutare se il giudizio medio a livello di hotel è sostanzialmente differente fra i due siti.

Il grafico successivo rappresenta le differenze medie per tutti gli hotel divisi per stelle dove la grandezza di ogni punto è proporzionale al numero totale di giudizi di quel determinato hotel. I giudizi su Booking e Trivago considerano solamente le situazioni in cui ci sono almeno 10 recensioni; per rendere i risultati ancora più robusti si è deciso di tenere solamente gli hotel con almeno 100 giudizi (perdendo solamente una quarantina di hotel dal numero di partenza).

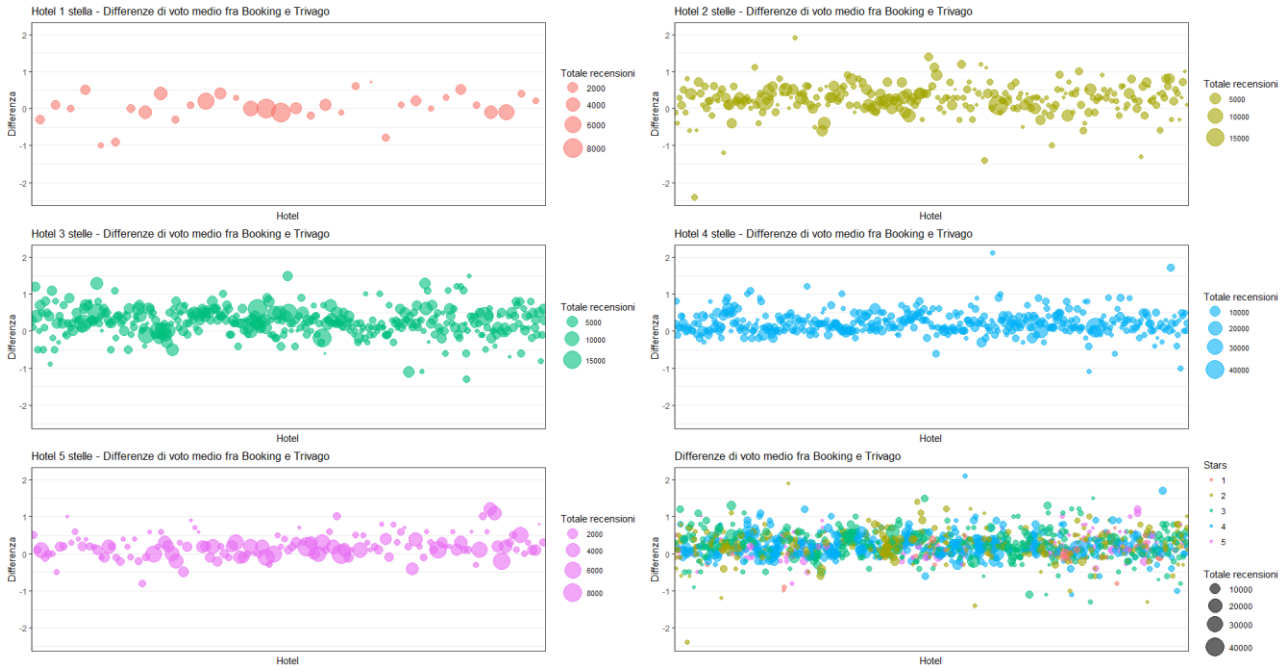

Fig.10: *Confronto score fra Trivago e Booking*

Analizzando attentamente si notano delle differenze per tutte le stelle, anche se alcune sono più evidenti di altre: gli hotel a 4 e 5 stelle hanno delle differenze minori rispetto alle altre categorie, un risultato prevedibile anche in relazione alla visualizzazione precedente dove si evidenziava una correlazione fra il giudizio medio ed il numero di stelle. Gli hotel con una sola stella sono molto meno numerosi di tutte le altre categorie, mentre quelli con 2 e 3 stelle sono numerosi e la differenza risulta evidente per un numero considerevole di hotel. Dalla rappresentazione finale in

basso a destra che raccoglie i cinque grafici precedenti si nota come nella maggior parte dei casi la differenza è compresa all'interno dell'unità di voto.

La distribuzione della differenza di voti è asimmetrica con la mediana posta a 0.2 in favore di Booking. Il 59.07% delle differenze erano comprese nel range $[-0.3, 0.3]$ e l'83.39% fra $[-0.5, 0.5]$. Si sono trovate sporadiche differenze sostanziali di giudizio, passando da un minimo di -2.4 fino ad un massimo di 2.1. Questi casi unici presentavano un numero di recensioni sostanzialmente superiore al minimo di 100 impostato precedentemente (tranne un singolo hotel che aveva 103 recensioni su Booking) e la differenza, per quanto anomala, è da attribuirsi al sistema tRI. Le possibili cause sono:

- In diverse situazioni Booking ha un numero di recensioni molto più basso rispetto a quelle di Trivago, di conseguenza altre fonti hanno un peso maggiore sul voto finale.
- Come specificato precedentemente il fattore temporale gioca un ruolo importante nella media finale.
- Altre particolarità nell'algoritmo utilizzato che non sono riportate esplicitamente.

In conclusione, su questo aspetto non esiste una differenza sostanziale di giudizio a parte casi sporadici, a seconda delle preferenze dell'utente è consigliabile o meno fare un raffronto fra i due siti: se l'utente in questione è interessato al giudizio più che alle recensioni scritte, dato che esiste da entrambe le parti l'opzione di filtrare gli hotel per fascia di voto, un confronto fra Trivago e Booking è raccomandabile.

Conclusioni

Dalla word cloud sono emersi alcuni aspetti interessanti: la gentilezza e la cordialità dello staff è l'aspetto più importante per gli utenti, ma a sorpresa lo staff non viene citato nei commenti negativi che si soffermano più sulla mancanza o il funzionamento difettoso di apparecchi della camera, come l'aria condizionata, per quanto riguarda gli hotel di fascia bassa, mentre per gli hotel più rinomati e costosi le lamentele principali si concentrano più sulla camera in sé.

Dall'analisi dell'andamento dei rating contro i prezzi emerge una correlazione positiva; tendenzialmente gli alberghi a 4 o 5 stelle ricevono voti più elevati rispetto a quelli con un numero minore di stelle.

Dall'analisi dei voti medi si è notato che generalmente non esiste una grande distinzione di voto di una struttura proposta su Booking e su Trivago: a parte casi isolati la differenza si attesta in un range bassissimo o comunque difficilmente influente sulla decisione finale per la prenotazione.

Bibliografia:

1. Bogenimaging.it: Booking, Trivago, Expedia
2. Wikipedia: Trivago
3. Wikipedia: MongoDB
4. MongoDB.com: Sharding Manual