

UNIVERSITÀ DEGLI STUDI DI TORINO



SCUOLA DI SCIENZE DELLA NATURA

DIPARTIMENTO DI INFORMATICA

CORSO DI LAUREA TRIENNALE IN INFORMATICA

TESI TRIENNALE

TIME FILTERING

Relatore

Prof. Marino SEGNAN

Candidato

Fabrizio SANINO

925515

Anno Accademico 2021–2022

Un uomo che osa sprecare anche
solo un'ora del suo tempo non ha
scoperto il valore della vita

Charles Darwin

Dichiarazione di originalità

Dichiaro di essere responsabile del contenuto dell'elaborato che presento al fine del conseguimento del titolo, di non avere plagiato in tutto o in parte il lavoro prodotto da altri e di aver citato le fonti originali in modo congruente alle normative vigenti in materia di plagio e di diritto d'autore. Sono inoltre consapevole che nel caso la mia dichiarazione risultasse mendace, potrei incorrere nelle sanzioni previste dalla legge e la mia ammissione alla prova finale potrebbe essere negata.

Indice

1	Introduzione	4
1.1	Problemi di ricerca sulle piattaforme di E-commerce	4
1.1.1	Esempio ricerca	4
1.2	Soluzione	6
1.2.1	Diagramma di contesto	6
2	Raccolta dei requisiti	8
2.1	Requisiti utente	8
2.2	Casi d'uso	9
3	Architettura	11
3.1	Front-end	11
3.1.1	MVVM	11
3.1.2	Android	12
3.2	Back-end	12
4	Front-end	14
4.1	Home - Saved searches	14
4.2	Specifiche tecniche	16
5	Back-end	18
5.1	Server	18
5.1.1	Python	18
5.1.2	Flask	18
5.1.3	Database	20
5.2	Scraping	20
5.2.1	Analisi struttura HTML	21
5.2.2	Beautiful Soup	22
5.2.3	Selenium	23
5.2.4	Filtraggio	23
5.3	Sentiment Analysis	25
5.3.1	NLTK	26
5.3.1.1	Ripulitura	26
5.3.1.2	Tokenization	26
5.3.1.3	Eliminazione delle "stopwords"	27
5.3.1.4	Enrichment - POS tagging	27

5.3.1.5	Ottenimento delle "steam words"	27
5.4	Rule-based	29
5.4.1	VADER	29
5.5	Machine Learning based	29
5.5.1	Classificatore Bayesiano	29
5.5.2	Implementazione	31
6	Ottimizzazione	32
6.0.1	Parallelizzazione ricerca	32
6.0.2	Fast transmit	32
7	Conclusioni	34
7.1	Difficoltà riscontrate	34
7.1.1	Problema del filtraggio	34
7.2	Limitazioni	34
7.3	Sviluppi futuri	35
7.3.1	Miglioramento filtraggio	35
7.3.2	Multiutenza	35
7.3.3	Miglioramento barra di ricerca	35
7.4	Applicazioni simili	35
7.5	Link GitHub	36

Capitolo 1

Introduzione

1.1 Problemi di ricerca sulle piattaforme di E-commerce

Al giorno d'oggi l'acquisto e la vendita di prodotti sulla rete è una pratica ampiamente utilizzata da parte delle persone. Esistono molte piattaforme che permettono di comprare un qualsiasi tipo di oggetto on-line. Siccome queste aziende presentano sul mercato un'offerta molto ampia, l'utente deve cercare il prodotto che vuole acquistare tra tantissimi altri: deve quindi compiere un'operazione di **filtraggio**. Questa operazione, teoricamente, dovrebbe essere fatta dai motori di ricerca all'interno degli **E-commerce**. Spesso non è così... vengono mostrati i prodotti che sono acquistati più frequentemente, quelli sponsorizzati, quelli che l'azienda vuole vendergli sulla base delle sue preferenze. Infatti, i motori di ricerca, quando l'utente inserisce un determinato prodotto con delle caratteristiche precise, non risponde con solo prodotti aventi quelle caratteristiche ma risponde in base ai gusti generali degli utenti. Questo porta quindi l'utente a dover filtrare i risultati di un filtraggio, operazione onerosa in termini di tempo perchè l'uomo è più lento di un computer.

1.1.1 Esempio ricerca

Facendo una ricerca su un qualsiasi sito di E-commerce possiamo osservare questo fenomeno. (Il prodotto che cerchiamo è arbitrario)

Proviamo a cercare su Amazon un Notebook ASUS con 16GB di RAM. Questo è il risultato

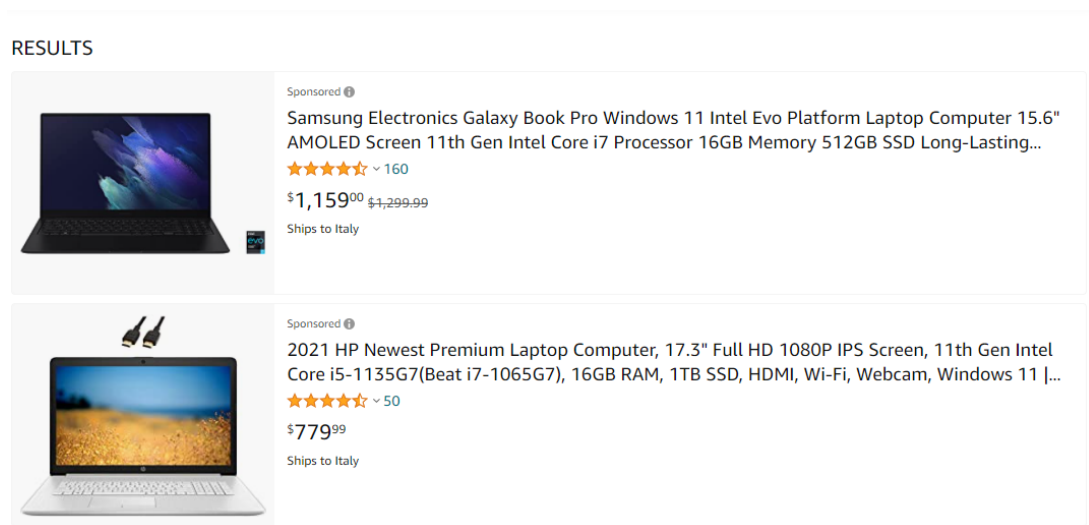


Figura 1.1: Esempio di ricerca

Ci vengono mostrati, come primi due risultati, prodotti che non sono nemmeno della marca corretta, infatti, il primo è un Samsung mentre il secondo un HP. Dopo una serie di ricerche, è stato stimato quanto segue:

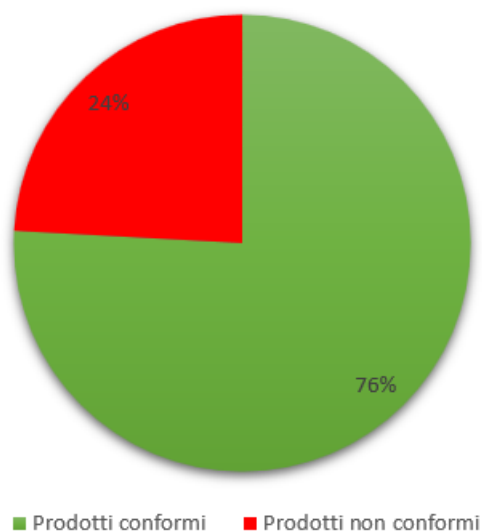


Figura 1.2: Confronto prodotti

cioè in ogni ricerca che l'utente effettua, ogni circa 4 prodotti, è presente un prodotto che non rispecchia le caratteristiche. Più la ricerca diventa precisa, più questa probabilità aumenta.

In questa tesi è stata discussa una soluzione di come può essere risolto il problema.

1.2 Soluzione

Per risolvere il problema è stata sviluppata un'applicazione Android che permette di **filtrare** ciò che la ricerca su un qualsiasi sito di E-commerce fornirebbe agli utenti, ovvero non mostrando tutti quei prodotti che non sono conformi rispetto alle specifiche dettate dall'utente. Inoltre l'applicazione permette, tramite una Supervised Machine Learning di classificare il prodotto in base alle recensioni di questo, fornendo all'utente la percentuale di recensioni positive e negative, aiutandolo a scegliere il prodotto migliore.

Nell'applicazione da me creata è stato fornito un esempio di filtraggio sulla piattaforma di E-commerce Amazon.

1.2.1 Diagramma di contesto

In questo diagramma è rappresentato, in modo sintetico, le relazioni che sono presenti tra i vari sistemi utilizzati e l'utente.

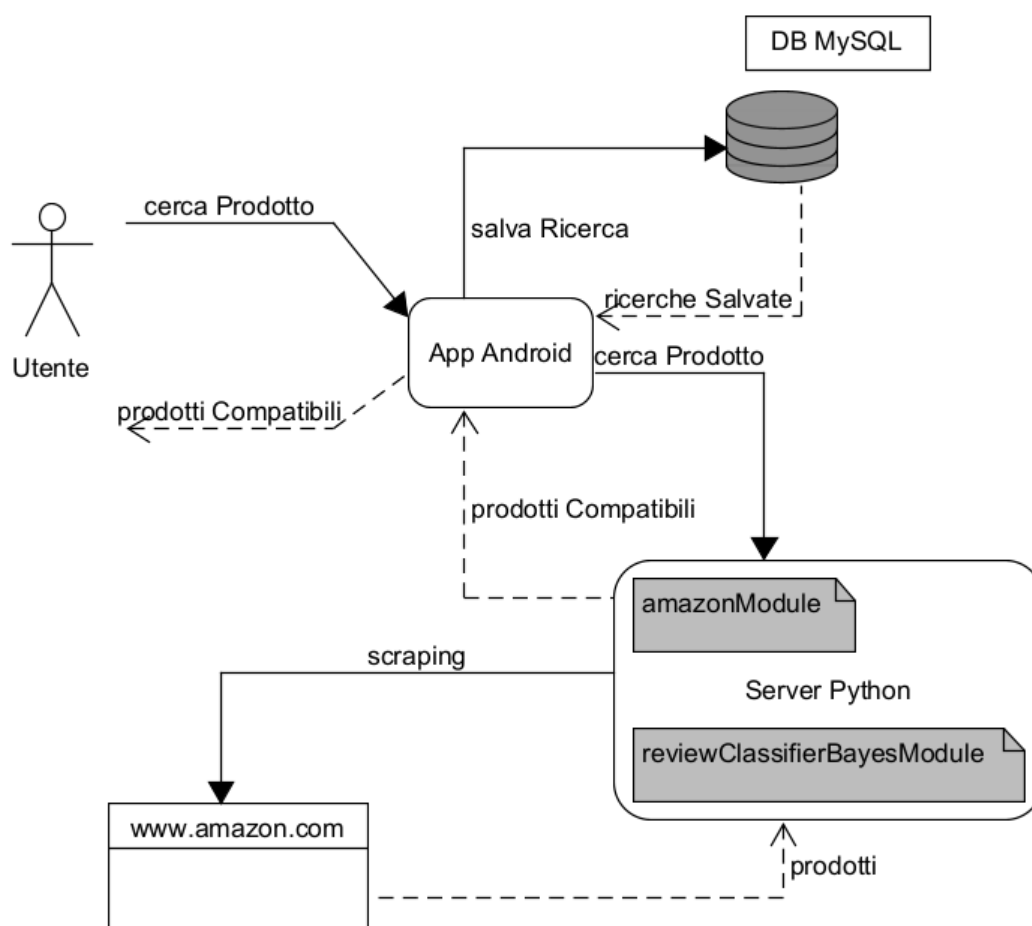


Figura 1.3: Diagramma di contesto

L'utente, utilizzando l'applicazione, inserisce il prodotto che vuole cercare (invece che cercarlo direttamente sul sito di E-commerce) all'interno di una barra di ricerca. Dopo di che, l'applicazione si collega ad un server **Python** che è funziona grazie a due moduli:

- **amazonModule**: si collega al sito di E-commerce permettendo di effettuare lo scraping dei prodotti risultanti da una ricerca.
- **reviewClassifierBayesModule**: permette di analizzare le recensioni di un prodotto e di restituire l'analisi effettuata.

Capitolo 2

Raccolta dei requisiti

2.1 Requisiti utente

In questa sezione sono descritti, tralasciando i dettagli implementativi, i servizi che il sistema offre. Questo passo è stato fatto prima di passare alla fase di sviluppo vero e proprio in modo tale da aver ben chiaro quali fossero gli obbiettivi che l'utente voleva raggiungere, prima di passare ad implementarli.

Dopo un attento e dettagliato studio, i requisiti individuati sono:

- Consentire all'utente di effettuare una ricerca su Amazon tramite un'apposita barra di ricerca.
- Filtrare i risultati considerando in AND tutte le parole chiave inserite dall'utente nella ricerca. Quindi, un prodotto sarà mostrato SE E SOLO SE contiene tutte le parole chiavi.
- Salvataggio ed eliminazione di una ricerca effettuata in modo da averla sempre a disposizione.
- Consentire all'utente di ordinare i prodotti in base sulla base di diversi ordinamenti.
- Consentire all'utente di ottenere una valutazione del prodotto in base alle recensioni di esso.

2.2 Casi d'uso

In questa sezione sono descritti gli scenari interessanti che il sistema software dovrà implementare. Per la stesura è stato deciso di utilizzare il formato breve.

Effettua ricerca

L'utente inserisce il prodotto e le caratteristiche che il prodotto deve avere all'interno di una barra di ricerca. Il sistema filtra i risultati prodotti dal sito di E-commerce e presenta la lista all'utente. L'utente consulta i risultati e naviga sulla pagina di un prodotto.

Salva ricerca

L'utente preme sul pulsante per salvare la ricerca. Il sistema carica su un sistema di memorizzazione la ricerca e conferma al cliente che l'operazione è avvenuta.

Ordina prodotti

L'utente preme sul pulsante per ordinare i prodotti decidendo l'ordine che essi devono avere. Il sistema ordina i prodotti secondo il criterio deciso dall'utente e mostra il risultato.

Valutazione prodotto

L'utente preme sul pulsante per valutare il prodotto. Il sistema consulta la recensione e decide la valutazione da assegnare al prodotto mostrandola all'utente.

Dopo aver analizzato nel dettaglio ciascuno caso d'uso, si è passati alla rappresentazione, tramite i *Diagrammi di sequenza (SSD)*, di ciascun caso d'uso in modo da evidenziare tutte le azioni che l'utente può fare e le eventuali eccezioni scatenate dal sistema.

Per brevità verrà riportato un solo diagramma di sequenza riferito al caso d'uso "Effettua ricerca" che rappresenta lo scenario principale di successo.

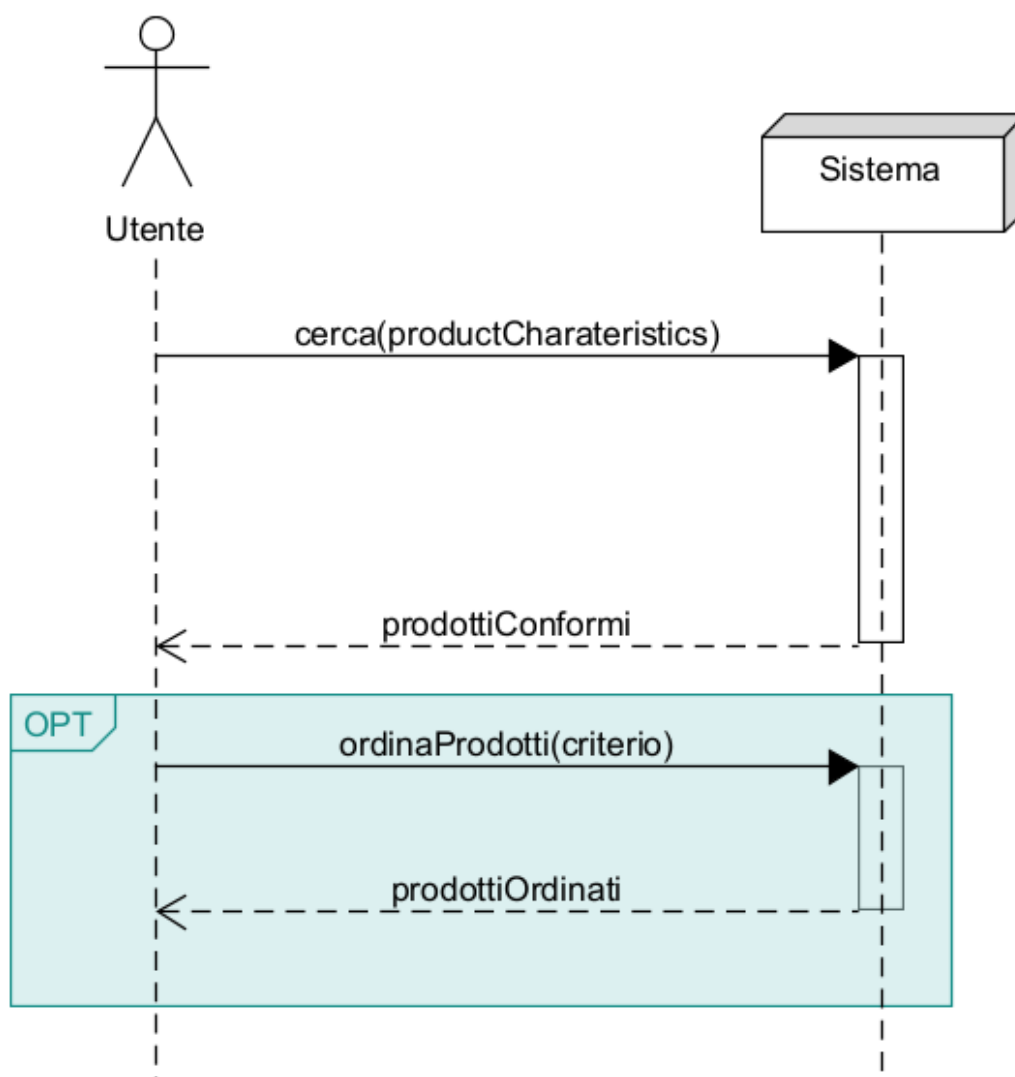


Figura 2.1: Diagramma di sequenza "Effettua ricerca"

Capitolo 3

Architettura

3.1 Front-end

Per sviluppare i casi d'uso sopra citati, è stata realizzata un'applicazione per il sistema operativo **Android** in cui l'utente può facilmente effettuare la proprie ricerche senza dover utilizzare direttamente il sito di E-commerce.

L'applicazione è organizzata mediante l'utilizzo del pattern **MVVM**.

3.1.1 MVVM

E' un **Pattern Architeturale**. Il pattern è una *Soluzione progettuale generale ad un problema ricorrente*. Nello specifico, MVVM, è uno schema che permette di organizzare la struttura di un sistema software suddividendo in modo chiaro le parti che lo compongono. In MVVM sono presenti 3 elementi fondamentali:

- **Model**: in cui sono presenti i dati che utilizza l'applicazione.
- **View**: contiene l'interfaccia utente, che è formata da 2 Fragment (Home e Saved searches).
- **ViewModel**: fa da link tra il Model e le View ed è utilizzato per manipolare i dati che arrivano dal Server.

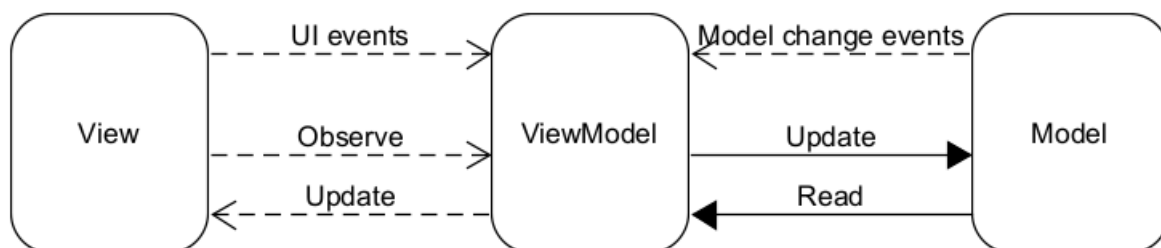


Figura 3.1: Pattern MVVM

3.1.2 Android

E' un sistema operativo **open source** basato su **Linux** che viene eseguito su smartphone, tablet e computer. Offre un approccio unificato nello sviluppo di applicazioni per i dispositivi mobili cioè gli sviluppatori possono sviluppare solo applicazioni per Android e queste verranno eseguite solo su dispositivi Android. Per lo sviluppo di applicazioni si utilizza spesso **Java** ma è anche possibile utilizzare **Kotlin**.



Figura 3.2: Logo dell'applicazione

3.2 Back-end

L'applicazione si collega ad un server **Python** il cui compito principale è quello di navigare sul sito di E-commerce, effettuare la ricerca del prodotto richiesto dall'utente e filtrare i risultati in modo tale che i prodotti restituiti siano il più conformi possibili a quelle che sono state le richieste dell'utente.

Per poter analizzare i prodotti che un E-commerce restituisce a fronte di una richiesta, è stato utilizzato il **Web Scraping** (Capitolo 5.2). Ogni sito ha una sua struttura, quindi lo Scraping deve essere effettuato "ad hoc" per ciascuno. A titolo di esempio è stato effettuato per Amazon ma può essere fatto per qualsiasi.

E' possibile modificare il sito di E-commerce su cui l'applicazione lavora definendo un apposita classe (con il nome *sitoDiE-commerceModule*). La classe creata deve necessariamente estendere tutti i metodi che sono presenti nella classe astratta *scrapingModuleAbstract* in modo tale che il server, e quindi il filtraggio, possa continuare a funzionare. L'estensione dei metodi e quindi il codice scritto all'interno, dipende da come il sito web è strutturato.

La classificazione dei prodotti, ovvero la raccolta delle recensioni e l'analisi di esse, è fatta in un modulo (*reviewClassifierBayes*) utilizzando il **Classificatore Bayesiano**. Siccome ci sono più tecniche utilizzabili, il classificatore è definito in un modulo a parte in modo tale che possa essere cambiato con un altro tipo di classificatore solo modificando l'*import* sul server. Siccome, per funzionare, il classificatore ha bisogno delle recensioni, esse devono essere ottenute sempre tramite lo Scraping. I metodi di scraping delle recensioni devono essere definiti all'interno del modulo descritto in precedenza.

Grazie a questa architettura, modificando un numero limitato di linee di codice, è possibile modificare il sito di E-commerce (Amazon, AliExpress, Ebay,

...) su cui effettuare il filtraggio e il tipo di classificatore usato per stimare la positività o la negatività di una recensione.

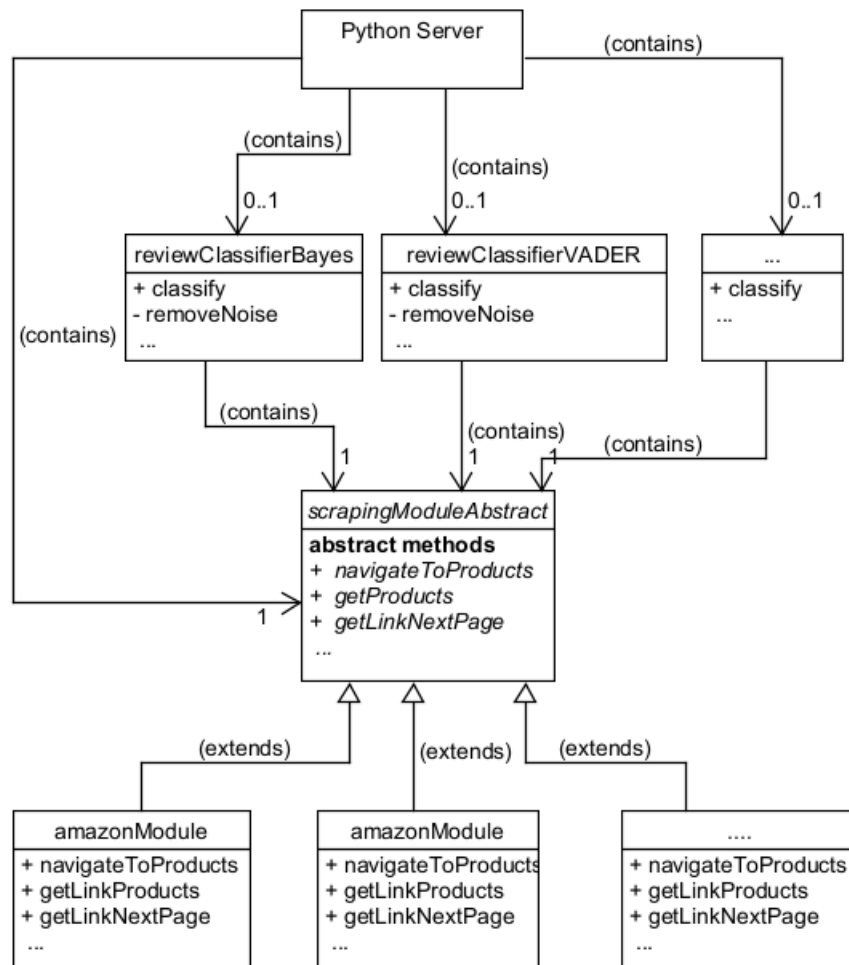


Figura 3.3: Architettura Server

Capitolo 4

Front-end

In questo capitolo verrà descritta l'interfaccia grafica dell'applicazione creata, chiamata **TIME FILTERING**.

4.1 Home - Saved searches

La prima schermata che l'utente vede quando avvia l'applicazione è la HOME. Nella HOME (Figura 4.1, Pagina 15) l'utente trova una casella di ricerca in cui può inserire il prodotto, e le caratteristiche di cui è interessato, che vuole cercare. L'applicazione si collegherà ad un Server che restituirà i risultati, filtrati secondo le specifiche (Figura 4.2, Pagina 15).

Accanto a ciascun risultato (Sezione 5.2.4) compare una piccola immagine che indica l'affidabilità del prodotto, ovvero quanto è pertinente rispetto alle specifiche inserite nella ricerca. L'immagine cambia in base alla pertinenza. L'affidabilità è decisa dal server in base a delle formule matematiche (Sezione 5.2.4).

Assieme al prodotto compare anche un pulsante CLASSIFY che permette di valutarne le recensioni. A valutazione completata comparirà una Dialog con indicate le percentuali di recensioni positive, negative e neutrali (Figura 4.4, Pagina 15). In questo modo l'utente, senza aver nemmeno controllato il prodotto, può avere una valutazione di esso facendogli risparmiare tempo, evitando così quei prodotti giudicati "scadenti" dagli altri utenti.

Utilizzando il menu, in alto a destra, l'utente può ordinare i risultati in base al prezzo, all'affidabilità e al rating basato sulle stelle inserite dagli utenti durante la scrittura della recensione.

Aperto il menu, in alto a sinistra, è possibile navigare sulla schermata SAVED SEARCHES. Nella SAVED SEARCHES (Figura 4.3, Pagina 15) l'utente troverà le ricerche salvate, tramite il tasto SAVE (Figura 4.2, Pagina 15) dopo aver fatto una ricerca nella HOME.

Utilizzando il tasto VIEW può mostrare tutti i prodotti che fanno parte di quella ricerca mentre tramite il tasto DELETE può eliminare la ricerca. In questo modo l'utente può facilmente, ed in poco tempo, accedere alle sue ricerche preferite.

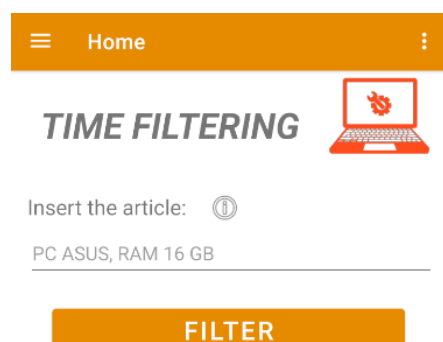


Figura 4.1: Home

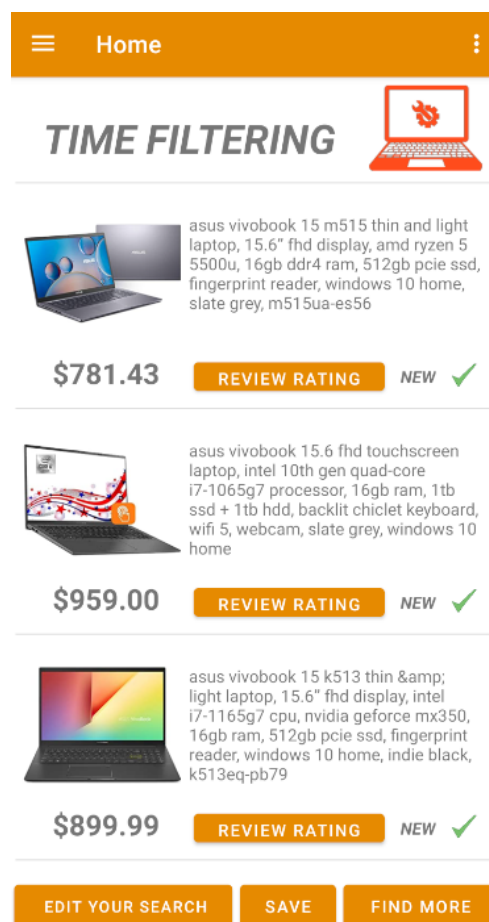


Figura 4.2: Lista dei prodotti

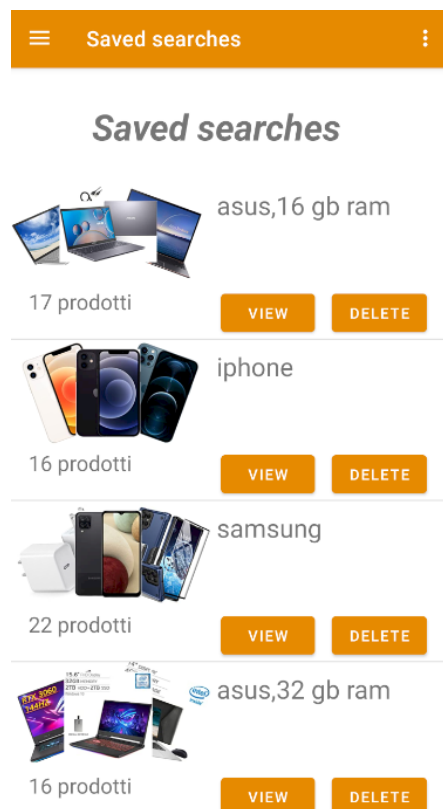


Figura 4.3: Ricerche salvate

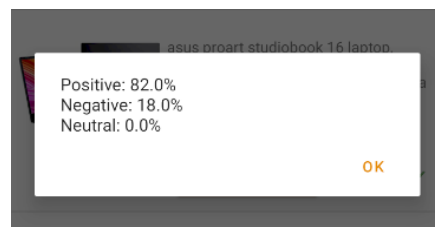


Figura 4.4: Classifica recensioni

4.2 Specifiche tecniche

L'applicazione è stata realizzata usando **JAVA** (Android nativo). E' organizzata mediante l'utilizzo del pattern **MVVM** spiegato in precedenza (Capitolo 3.1).

Il **ViewModel** è la parte fondamentale dell'applicazione. E' rappresentato con la classe **NetworkViewModel**. Ha vari metodi che vengono richiamati dall'interfaccia grafica per andare a reperire dei dati. Una volta che il Server ha risposto, i dati vengono manipolati per poter essere presentati nella View. Tramite i **LiveData** viene notificata alla View che l'operazione da lei richiesta è stata completata.

I **LiveData** vengono utilizzati per osservare le modifiche di un particolare view-Model e quindi aggiornare l'interfaccia grafica quando avviene la modifica. Essi sono sensibili al ciclo di vita dell'applicazione ovvero ogni volta che un **LiveData** viene modificato, gli aggiornamenti vengono inviati solo a quelle componenti dell'app che sono nello stato *Attivo*. Tutte quelle componenti che non sono attive verranno notificate non appena lo diventeranno. Qui sotto è riportata una parte di codice in cui vengono utilizzati i **LiveData**. La funzione, in background, contatta il server e ottiene tutte le ricerche che l'utente ha salvato.

```
public class findResearchDetails extends AsyncTask<JSONObject, Void,
    JSONArray> {
    @Override
    protected JSONArray doInBackground(JSONObject... jsonObjects) {
        return
sentPOSTRequest(urls.getServerUrlGetResearchDetail(),
jsonObjects[0]);
    }

    @Override
    protected void onPostExecute(JSONArray result) {
        searchResults.emptyList();

        JSONObject retValue = new JSONObject();

        try {
            if (result.getJSONObject(0).getBoolean("error")) {
                retValue.put("error", true);
                retValue.put("errorDescription",
result.getJSONObject(0).getString("errorDescription"));

researchDetailsLiveData.updateResearchDetails(retValue);
            } else {
                for (int i = 1; i < result.length(); i++) {
                    JSONObject item = result.getJSONObject(i);
                    SearchResult s = new
SearchResult(item.getString("productName"),
```

```

        item.getString("productLink"), item.getString("productImg"),
        item.getString("price"), item.getInt("reliability"), false,
        Float.parseFloat(item.getString("ratingReview"))));
        searchResults.addElement(s);
    }

    searchResults.getArrayList().sort(Comparator
        .comparingInt(SearchResult::getReliability));
    Collections.reverse(searchResults.getArrayList());

    adapterSearch.setData(searchResults.getArrayList());

    searchResults.setSavedSearch(true);

    retValue.put("error", false);

    researchDetailsLiveData.updateResearchDetails(retValue);
    }
    } catch (JSONException e) {
        e.printStackTrace();
    }
    }
}

```

Con l'istruzione

```
researchDetailsLiveData.updateResearchDetails(retValue);
```

notifichiamo alla View che le ricerche salvate sono arrivate o se ci sono stati degli errori nel download.

La View tramite

```
networkViewModel.getReadResearchLiveData()
    .observe(getViewLifecycleOwner(), result -> {...});
```

riceve la notifica del cambiamento di stato ed effettua le operazioni necessarie. In questo caso, se non sono avvenuti degli errori, nasconde la dialog del caricamento o altrimenti, in caso di errore, notifica all'utente il tipo di errore.

Capitolo 5

Back-end

5.1 Server

L'applicazione comunica con un Server scritto in **Python**. E' stato deciso quel linguaggio perchè, per andare ad analizzare i risultati di una ricerca sul Web è necessario utilizzare lo **Scraping** ed il modo più veloce per effettuarlo è tramite Python.

5.1.1 Python

E' un linguaggio di programmazione che ha le seguenti caratteristiche:

- **interpretato**: ovvero è processato a tempo di esecuzione tramite l'interprete, in questo modo, il programma, non deve essere compilato prima della sua esecuzione.
- **orientato agli oggetti**: cioè supporta lo stile di programmazione in cui il codice è incapsulato in oggetti.
- **interattivo**: è possibile dialogare direttamente con l'interprete.

Python è molto utile quando è necessaria una rapidità nello sviluppo delle applicazioni ma soprattutto quando è necessario connettere delle componenti tra di loro, infatti, proprio per questo motivo è stato utilizzato. Supporta moduli e package incoraggiando così la programmazione modulare ed il riutilizzo di codice.

5.1.2 Flask

Flask è un **web framework** che permette di sviluppare applicazioni web in modo facile e veloce. Con una numero esiguo di istruzioni consente di creare un server. Nello specifico, Flask, è un micro-framework ovvero ha poca dipendenza, talvolta anche senza dipendenze, dalle librerie esterne. In questo modo risulta veloce ed efficiente.

Qui di seguito un esempio di come è stata creata una **route** per rendere disponibile, "all'esterno", un servizio:

```
from flask import Flask
from flask import request
from flask import jsonify

@app.route("/deleteResearch", methods=['POST'])
def deleteResearch():
    content_type = request.headers.get('Content-Type')
    if content_type != 'application/json;charset=UTF-8':
        return {"error": "Content-Type not supported!"}, 422
    else:
        clientParameter = json.loads(request.get_data(), strict=False)
        idResearch = clientParameter["idResearch"]

        try:
            connection = mysql.connector.connect(user='root',
            password='', host='127.0.0.1', database='savedResearch')

            if connection.is_connected():
                cursor = connection.cursor()

                try:
                    #Elimino tutti i prodotti della ricerca
                    cursor.execute("DELETE FROM savedproducts WHERE
researchNum = " + str(idResearch))

                    connection.commit()

                    #Elimino la ricerca
                    cursor.execute("DELETE FROM search WHERE id = " +
str(idResearch))

                    connection.commit()

                    connection.close()

                    return jsonify([{"error":False}])
                except:
                    if connection.is_connected():
                        cursor.close()
                        connection.close()

                    return jsonify([{"error": True,
"errorDescription": "Error while deleting"
}])
```

```
except:
    return jsonify([
        "error": True,
        "errorDescription": "Connection error"
    ])

if __name__ == "__main__":
    app.run()
```

In questo modo, collegandoci alla URL `127.0.0.1/deleteResearch` viene richiamata la funzione `deleteResearch` che permette di servire la richiesta. In questo caso, viene eliminata la ricerca salvata sul Database.

5.1.3 Database

Per salvare le ricerche effettuate dall'utente è stato utilizzato un Database SQL-based, nello specifico **MySQL**. La connessione con il Database server avviene utilizzando un **connector**. Il Database connector, detto anche Database **driver**, è un software che permette ad un'applicazione di connettersi ed interagire con un Database. Quindi prima di aprire una connessione con il Database è necessario importare *MySQL Connector/Python*.

Una volta creata la connessione, tramite la creazione di un *cursor* è possibile inviare delle query al Database.

Una volta effettuate le modifiche deve essere eseguito il comando `connection.commit()` che permette di rendere permanenti le modifiche e terminare la transazione. Nel caso in cui sia accaduto un errore, tramite il comando `connection.rollback()` lo stato del database viene riportato a quello precedente alle modifiche effettuate nella transazione corrente, annullando così le modifiche.

5.2 Scraping

Per poter analizzare i prodotti forniti da un sito di E-commerce durante la ricerca, è stata utilizzata la tecnica dello **Web Scraping**.

Il Web Scraping è un processo che permette di estrarre contenuti e dati da una pagina web. E' possibile estrarre i contenuti in più formati, ma quello più utilizzato è l'HTML. Il processo può naturalmente essere fatto manualmente andando ad analizzare l'HTML della pagina ed estraendo i dati "a mano", ma lo scopo del Web Scraping è proprio quello di automatizzare il processo ed aumentare la capacità di dati che possono essere collezionati. Quando ci si avvicina allo Scraping bisogna fare attenzione a due aspetti molto importanti:

- **Varietà:** Ogni sito web è costruito in modo diverso, ha una specifica struttura HTML. Quindi, per ogni sito di cui vogliamo collezionare i dati, è necessario effettuare un lavoro "ad-hoc" ripetendo tutti i passaggi di scraping dall'inizio. Per ovviare a questo problema si è deciso di dividere la

parte di Web server dalla parte di Scraping in modo tale che il server sia riutilizzabile al modificarsi dello scraping (Capitolo 3.2).

- **Durabilità:** I siti web spesso vengono aggiornati, ovvero viene modificata la loro struttura e quindi il codice HTML che li compone. Questo porta quindi a dover aggiornare di frequente il tool di Scraping perchè un giorno può funzionare, ma il giorno dopo no. Quindi saranno necessarie integrazioni durante tutta la vita dell'applicazione creata.

Un'alternativa allo scraping sono le **API**. Esse svolgono la funzione di intermediari cioè permettono a due applicazioni di poter comunicare. La comunicazione avviene tramite lo scambio di **JSON** o di **XML** che sono dei formati per lo scambio di dati strutturati. In questo modo non è necessario convertire l'HTML in dati strutturati perchè questi formati lo sono già. Questa tecnica però non è sempre utilizzabile perchè le API molto spesso sono private.

5.2.1 Analisi struttura HTML

Prima di passare allo Scraping è necessario analizzare la pagina per capire com'è strutturata in modo da creare delle procedure automatizzate che riescano ad ottenere le informazioni. Per prima cosa è stata analizzata la pagina HTML che viene restituita dopo una ricerca su Amazon.

(Di seguito un esempio semplificato di come è formata)

```
<div data-component-type="s-search-result">
  <span class="a-size-base-plus a-color-base a-text-normal">
    <!-- Nome del prodotto -->
  </span>
  <span class="a-offscreen">
    <!-- Prezzo del prodotto -->
  </span>
  <a class="a-link-normal s-underline-text s-underline-link-text
s-link-style a-text-normal">
    <!-- Link all'immagine del prodotto -->
  </a>
</div>
```

Dopo di che è stata analizzata la pagina di un gruppo di prodotti, scelti in modo casuale, per capire quali fossero le regolarità dei tag utilizzati.

(Di seguito un esempio semplificato di come sono formati)

```
<div id="featurebullets_feature_div">
  <ul>
    <li>
      <span class="a-list-item">
        <!-- Caratteristica del prodotto -->
      </span>
    </li>
  </ul>
</div>
```

```
        ...
        ...
    </ul>
</div>
<div id="productDescription">
    <p>
        <!-- Descrizione del prodotto -->
    </p>
</div>
<div>
    <table id="productDetails_techSpec_section_1">
        <tbody>
            <tr>
                <th>
                    <!-- Titolo dettaglio -->
                </th>
                <td>
                    <!-- Dettaglio -->
                </td>
            </tr>
            ...
            ...
        </tbody>
    </table>
</div>
```

Una volta analizzata la pagina bisogna decidere che tipo di libreria utilizzare per trasformare i dati contenuti nell'HTML in modo che possano essere sfruttati. I due modi più utilizzati sono **Beautiful Soup** e **Selenium**.

5.2.2 Beautiful Soup

E' una libreria Python che permette di convertire i dati non strutturati contenuti nell'HTML in dati strutturati che possano essere utilizzati.

Tramite la seguente istruzione

```
soup = BeautifulSoup(page.content, "html.parser")
```

la libreria scarica il contenuto HTML nella variabile **soup** e rende possibile la navigazione di esso. Ad esempio tramite il comando `soup.find_all('p')` vengono restituiti tutti i tag *p*.

Il problema di questa tecnica è la lentezza, infatti, dopo una serie di test, è stato deciso di abbandonare questa libreria per adottarne un'altra molto più performante ovvero **Selenium**.

5.2.3 Selenium

E' una libreria Python che permette di emulare l'iterazione umana su un browser. Quindi l'iterazione con il browser non viene fatta da una persona ma da uno strumento automatico. Per usare Selenium è necessario scaricare un **Web Driver**. Serve per aprire in modo automatico il browser che decidiamo di utilizzare e di accedere al sito web che vogliamo. Bisogna scaricare il Web Driver specifico per il browser che abbiamo intenzione di utilizzare. Per l'applicazione creata è stato utilizzato Chrome in quanto, dopo una serie di test, si è rivelato quello più performante con questa libreria. Dopo aver fatto ciò è possibile utilizzare in modo automatico il browser. Per prima cosa bisogna collegarsi alla pagina principale di Amazon. Dopo di che si deve inserire nella barra di ricerca il nome del prodotto che ha cercato l'utente ed inviare il comando "ENTER". Infine, quando sono arrivati i risultati, si devono andare a leggere in modo da poterli analizzare. Tutto questo viene fatto con delle istruzioni specifiche.

(Di seguito una versione semplificata del codice utilizzato)

```
linkAmazon = 'https://www.amazon.com'
mainDriver.get(linkAmazon)

mainDriver.implicitly_wait(1)
search = mainDriver.find_element(By.ID, 'twotabsearchtextbox')
search.send_keys(keywordProductName.replace(',', ' '))
search.send_keys(Keys.ENTER)

mainDriver.implicitly_wait(0.5)

items = wait(mainDriver, 10).until(
    EC.presence_of_all_elements_located((By.XPATH,
        '//*[@data-component-type="s-search-result"]')))
```

Siccome la connessione non è veloce tanto quando lo scraping è necessario inserire delle `sleep(...)` o `driver.implicitly_wait(...)` in modo tale da lasciare il tempo alla connessione di scaricare la pagina.

Siccome, ai fini della ricerca, non è possibile solo analizzare il titolo del prodotto, perchè potrebbero non essere presenti dei dettagli importanti, bisogna navigare anche nella pagina del prodotto ed analizzare anche quella. Questo però porta al rallentamento delle prestazioni complessive perchè è una nuova pagina che deve essere scaricata. Si è stimato, che per analizzare 25 prodotti (quelli contenuti in una pagina Amazon) ci vogliono circa 3 minuti. Più il prodotto da cercare è specifico, più il tempo aumenta.

Una volta ottenuti i prodotti risultanti dalla ricerca su Amazon è necessario filtrarli in base alle caratteristiche che ha richiesto l'utente.

5.2.4 Filtraggio

Il filtraggio è stato il passaggio più critico nella realizzazione dell'applicazione.

Supponiamo che l'utente voglia cercare un Asus con 16GB di RAM. Cercando parola per parola all'interno dei prodotti di Amazon avrebbe un problema ovvero potrebbero essere restituiti come risultati pc con 16GB di SSD invece che di RAM, perchè cercando parola per parola, vengono restituiti tutti i prodotti che contengono la stringa 16GB. Il problema deriva dal fatto che le stringhe 16GB e RAM non sono cercate, nel prodotto, in modo correlato ma sono cercate in modo separato. Per ovviare a questo problema, l'utente viene obbligato ad inserire le caratteristiche del prodotto che vuole cercare separate da una virgola. Sempre facendo riferimento al esempio in precedenza, l'utente dovrebbe inserire *Asus, 16GB RAM, ...* nel campo di ricerca. In questo modo è possibile distinguere le singole caratteristiche. Verrà quindi cercato nel testo *Asus* e poi *16GB RAM*. Ciascuna caratteristica viene quindi cercata in blocco e non parola per parola.

Non basta però, perchè i prodotti di Amazon non hanno una rappresentazione comune delle caratteristiche, ma ognuno ha la propria. Ovvero è possibile che alcuni abbiano scritto *16GB RAM*, altri *GB 16 RAM* e così via. Si sono quindi resi necessari alcuni passaggi per manipolare le caratteristiche inserite dall'utente in modo da poter restituire il maggior numero di risultato possibili.

1. Nel testo del prodotto (quello che Amazon visualizza quando si ricerca un prodotto) vengono cercate tutte le **permutazioni** possibili della caratteristica. In riferimento all'esempio precedente vengono ricercate le seguenti parole 16 GB RAM, GB 16 RAM, ...

Dopo di che si provano tutte le **combinazioni** togliendo alcuni spazi tra le parole. Ad esempio verrà cercato *16 GBRAM, 16GB RAM, ...*

Se un prodotto contiene tutte le caratteristiche direttamente nel testo del prodotto allora l'affidabilità di esso è al 100%.

2. Le caratteristiche che non sono state trovate vengono ricercate nella pagina del prodotto ovvero andando a consultare anche i dettagli di esso sempre utilizzando il meccanismo spiegato al punto 1.

Se vengono trovate tutte le caratteristiche rimanenti allora, anche in questo caso, l'affidabilità del prodotto è al 100%

3. Per le caratteristiche che ancora non sono state trovate si fa ancora un ultimo tentativo. Si cercano le parole della caratteristica senza che siano collegate tra di loro, quindi non come un blocco unico. In riferimento all'esempio precedente, verrà cercato 16 poi GB ed infine RAM. Il prodotto per essere restituito all'utente deve comunque contenere tutte le caratteristiche che l'utente ha inserita ma naturalmente l'affidabilità sarà inferiore. In questo caso l'affidabilità viene calcolata con queste formule:

$$W = \frac{(90 \times Tw)}{Cw} \quad (5.1)$$

$$L = \frac{(90 \times Tl)}{Cl} \quad (5.2)$$

$$R = 100 - \frac{(W + L)}{2} \quad (5.3)$$

dove:

- Tw è il numero totale di parole nella ricerca dell'utente
- Cw è il numero di parole che non sono state trovate con la ricerca precedente
- Tl è il numero totale di lettere nella ricerca dell'utente
- Cl è il numero di lettere delle parole che non sono state trovate con la ricerca precedente
- W è la percentuale di parole che non sono state trovate con la ricerca precedente
- L è la percentuale di lettere che non sono state trovate con la ricerca precedente
- R è l'affidabilità che avrà il prodotto

Queste formule sono state generate dopo aver analizzato un numero molto grande di ricerche. E' stato quindi dedotto che più parole non sono state trovate con i metodi 1. e 2., più significa che il prodotto potrebbe non essere conforme ma soprattutto più le parole non trovate sono grandi, in lettere, più queste sono importanti e quindi l'affidabilità deve essere inferiore.

Una volta completati tutti questi passaggi per ciascun prodotto risultante dalla ricerca su Amazon, vengono spediti all'applicazione solo quelli conformi assieme alla percentuale di affidabilità calcolata.

Sull'applicazione, accanto a ciascun prodotto, è inoltre presente un bottone che consente di analizzare le recensioni del prodotto e quindi permette di valutarlo.

5.3 Sentiment Analysis

Sentiment analysis è una tecnica che permette di elaborare il linguaggio naturale e di estrarre, identificare e studiare stati affettivi e informazioni soggettive. In particolare, quello che serve nella mia applicazione è determinare se una recensione (quindi del testo) è positivo o negativo. Questa tecnica viene spesso utilizzata nell'ambito del *Market research* e *Social media monitoring*.

Sentiment analysis è una sottocategoria dell'**NLP** ovvero una tecnica che misura l'inclinazione dell'opinione delle persone (Positiva, Negativa, Neutrale). Gli approcci che possono essere applicati sono due:

- **Rule-based:** è una tecnica che permette di analizzare del testo senza *training* e senza modelli di *Machine learning*. Utilizza un insieme di regole chiamate **Lexicons**.
- **Machine Learning based**

Prima di procedere con le tecniche definite in precedenza è necessario trasformare il testo in modo che possa essere utilizzato dagli strumenti automatici. Per fare ciò è stata utilizzata la libreria **NLTK**

5.3.1 NLTK

"Natural Language ToolKit" è una libreria Python che contiene varie funzioni per l'analisi simbolica e statistica nel campo dell'elaborazione del linguaggio naturale. Per trasformare il testo è necessario seguire una serie di passaggi.

L'italiano non è un linguaggio supportato da questa libreria, quindi per gli esempi presentati si è utilizzato l'inglese.

5.3.1.1 Ripulitura

In questa fase vengono rimosse dalle recensioni i caratteri speciali ed i numeri perchè ininfluenti dal punto di vista della semantica. Ad esempio "Great CD: My lovely Pat has one of the GREAT voices of her generation." diventa "Great CD My lovely Pat has one of the GREAT voices of her generation". Questa operazione viene fatta con la semplicissima *Regular Expression* `[^A-Za-z]+`

Il testo però in questo modo non è ancora sfruttabile, quindi si passa alla fase di **Tokenization**.

5.3.1.2 Tokenization

E' il processo che permette di spezzare la recensione i pezzi più piccoli chiamati **Token**. Questo consente di lavorare in modo focalizzato, su ciascuna parte, che è comunque ricca di significato anche se considerata separatamente rispetto al testo. Esistono due tipi di Tokenization:

- **Tokenizing by word:** permette di identificare le parole che compaiono e la frequenza che hanno nel testo.
- **Tokenizing by sentence:** permette di capire come le parole sono relazionate le une alle altre.

La tecnica utilizzata è quella **by word**. Grazie alla libreria NLTK è possibile utilizzare la funzione `nltk.word_tokenize(...)` che restituisce la lista di Token. Ad esempio "Great CD My lovely Pat has one of the GREAT voices of her generation" diventa

```
['Great', 'CD', 'My', 'lovely', 'Pat', 'has', 'one', 'of',  
'the', ...]
```

A questo punto si passa alla fase di **Stopwords removal**

5.3.1.3 Eliminazione delle "stopwords"

Le Stopwords sono delle parole che portano con sé pochissima informazione rispetto al contenuto generale della frase, quindi, per semplicità possono essere rimosse. Per farlo si usa la funzione

```
("").join(e for e in tokens if e.lower() not in
stopwords.words('english'))
```

Il risultato applicato ai Token precedenti è

```
['Great', 'CD', 'lovely', 'Pat', 'one', ...]
```

Una volta fatto ciò è necessario effettuare la **Enrichment - POS tagging**.

5.3.1.4 Enrichment - POS tagging

Anche detta "*Parts Of Speech tagging*", è un processo che permette di trasformare i Token creati in precedenza in una lista avente la seguente forma (*parola*, *tag*). Assegna un'etichetta alla parola in base all'analisi grammaticale.

Con NLTK è possibile utilizzare la funzione `nlk.pos_tag(token)` che restituisce una lista con ad ogni parola il tag associato. Ad esempio

```
['Great', 'CD', 'My', 'lovely', 'Pat', 'has', 'one', 'of',
'the', ...]
```

diventa

```
[('Great', 'JJ'), ('CD', 'NN'), ('lovely', 'RB'), ...]
```

dove **JJ** indica un aggettivo, **NN** indica un nome e **RB** indica un avverbio. Naturalmente esistono anche altri Tag per le altre parti del testo.

5.3.1.5 Ottenimento delle "steam words"

Lo **steam** è una parte del testo che è responsabile del significato lessicale della frase. Ci sono due tecniche che possono essere utilizzate per estrapolare dalle parole le *steam words*:

- **Stemming**: è una tecnica meno accurata perchè si occupa solo di troncare il finale delle parole e spesso restituisce parole senza significato
- **Lemmatization**: è una tecnica che restituisce solo parole aventi significato. Necessita, però, che ciascuna parola abbia un **POS tag**.

E' stato deciso di utilizzare la tecnica della **Lemmatization** in quanto risulta più precisa e affidabile. Grazie alla funzione `wordnet_lemmatizer.lemmatize(word, pos=pos)` è possibile ottenere la *steam word* della parola passata come parametro. Una volta fatto questo le parole vengono ricomposte per formare una frase. Ad esempio

```
[('Great', 'NNP'), ('CD', 'NN'), ('lovely', 'RB'),...]
```

diventa "Great CD love ...".

Una volta estratte e rese utilizzabili attraverso i passaggi descritti in precedenza, le recensioni, possono passare alla fase di analisi per capire se ciascuna è positiva o negativa.

5.4 Rule-based

5.4.1 VADER

Valance Aware Dictionary and sEntiment Reasoner è un tool Rule-based che permette di effettuare il *Sentiment Analysis*. **VADER** usa un dizionario in cui ogni *lexicon* è associato ad un punteggio in base all'intensità emotiva e all'orientamento semantico. Questa tecnica non restituisce solo la positività o la negatività di una frase ma anche quanto una frase lo è.

Il punteggio di ciascuna recensione è ottenuto andando a sommare il punteggio di ciascuna parola presente nel testo.

Il vantaggio più importante che offre questa tecnica è che non richiede alcun *training data*, ma lo svantaggio più importante è che non è molto precisa. Quindi si è preferito passare ad una tecnica più avanzata.

5.5 Machine Learning based

E' una tecnica, alternativa alla precedente, che permette di assegnare a delle istanze non classificate, la loro classe di appartenenza. Grazie ad un insieme di dati detti **Learning set** e un **Algoritmo di apprendimento** tramite l'**induzione** viene creato un **modello**. Il modello sarà utilizzato per classificare le nuove istanze che non hanno una classe grazie alla **deduzione**. Il modello può avere varie forme, che dipendono dal tipo di algoritmo di apprendimento utilizzato, come ad esempio un **albero di decisione** o **regole di decisione**.

5.5.1 Classificatore Bayesiano

E' un algoritmo di classificazione **supervisionato** ovvero le classi non sono predette da esso ma vengono date tramite il **training set**. Questo tipo di classificatore si basa sull'algoritmo **Naive Bayes** che utilizza il teorema di **Bayes**. Il teorema di Bayes permette di calcolare la probabilità condizionata di un evento A rispetto ad un altro evento B sapendo che B si è verificato.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)} \quad (5.4)$$

Una caratteristica molto importante è che per l'algoritmo è importante la frequenza delle parole ma non la posizione che queste hanno all'interno del testo.

L'approccio utilizzato dall'algoritmo è questo:

1. Sappiamo di avere un insieme di possibili classi da assegnare $C = \{\text{Positiva}, \text{Negativa}\}$.
2. Deve essere calcolata la probabilità che una recensione appartenga ad una delle due classi.

3. Deve quindi essere eseguita un'iterazione su tutte le classi possibili e trovare quella che ha la probabilità condizionata più alta.

La recensione è vista come una **tupla** cioè un insieme di parole di in ognuna ha una delle caratteristiche.

I passi successivi mostrano com'è possibile derivare la formula utilizzata dall'algoritmo per calcolare la probabilità condizionata:

$$c^* = \operatorname{argmax} P(c|d) \quad c \in C \quad (5.5)$$

dove c^* conterrà la classe che ha la massima probabilità di appartenere alla tupla.

E' quindi possibile andare a sostituire $P(c|d)$ utilizzando il teorema di Bayes.

$$c^* = \operatorname{argmax} \frac{P(d|c) \times P(c)}{P(d)} \quad c \in C \quad (5.6)$$

dove $P(c | d)$ è la probabilità che nella classe si manifesti la tupla in questione, $P(c)$ è la probabilità a priori della classe e $P(d)$ è la probabilità che si manifesti la tupla in questione. $P(d)$ viene rimosso in quanto è un valore costante e quindi non cambia nel tempo.

$$c^* = \operatorname{argmax} P(d|c) \times P(c) \quad c \in C \quad (5.7)$$

A questo punto la recensione viene considerata come un insieme di parole w_i . w

$$c^* = \operatorname{argmax} P(w_1, w_2, \dots, w_n | c) P(c) \quad c \in C \quad (5.8)$$

Per continuare è necessario effettuare un'assunzione molto importante, ovvero dobbiamo supporre che la probabilità di ciascuna w (parola) sia **indipendente** dalle altre. Possiamo quindi ulteriormente semplificare la formula.

$$c^* = \operatorname{argmax} P(c) \times \prod P(w_i | c) \quad c \in C, \quad w_i = \text{parola } i - \text{esima} \quad (5.9)$$

Una volta derivata la formula è possibile applicarla. Nei passi successivi è mostrato un esempio.

1. Si calcola $P(c)$. Deve essere trovato il numero di recensioni che appartengono a ciascuna classe. Si suppone che il numero di recensioni di classe c siano Nc e il numero totale sia $Ntotal$.

$$P(c) = \frac{Nc}{Ntotal} \quad (5.10)$$

2. Si calcola $P(w_i | c)$. L'obiettivo è quello di calcolare il numero di volte che la parola w_i compare nel *training set* sapendo che la classe considerata è la c .

$$P(w_i | c) = \frac{\text{count}(w_i, c)}{\sum_{w_i \in V} \text{count}(w_i, c)} \quad (5.11)$$

dove V è l'insieme di tutte le parole presenti nel training set considerato. Esiste però un problema: in alcuni casi, il valore calcolato da questa formula può essere 0. Per evitare ciò si introduce un coefficiente chiamato **Laplace Smoothing Coefficient**(a), sia al numeratore che al denominatore.

$$P(w_i|c) = \frac{\text{count}(w_i, c) + a}{\sum_{w_i \in V} (\text{count}(w_i, c) + a)} \quad (5.12)$$

3. Si inseriscono i risultati calcolati precedentemente (nei punti 1. e 2.) nella formula 5.9. La probabilità più alta determinerà la classe di appartenenza della recensione.

Il modello creato dal classificatore sono le probabilità che vengono apprese e migliorate durante la fase di apprendimento.

5.5.2 Implementazione

Per poter adottare questo tipo di approccio è quindi necessario un **dataset** di partenza, ovvero una collezione di recensioni già classificate. Esso però deve essere manipolato per poter essere utilizzato, si procede quindi con i passaggi descritti in precedenza (5.4.1 - 5.4.5).

A questo punto si passa alla fase di **allenamento (training)**. E' un passo molto importante perchè permette al classificatore di interpretare correttamente i dati ed imparare in modo da eseguire le future classificazioni nel modo più corretto possibile. Il *dataset* di partenza, tramite la tecnica del **random subsampling** è stato diviso in varie partizioni. Alcune usate per la fase di apprendimento, altre per la fase di valutazione dell'accuratezza del classificatore. Siccome la fase di valutazione dipende molto dal *test set*, si ripete più volte il procedimento utilizzando partizioni diverse e alla fine viene fatta la media delle valutazioni per ottenere un risultato migliore è indipendente dalla composizione del *test set*.

Tramite l'istruzione

```
nltk.NaiveBayesClassifier.train(train_set)
```

il classificatore viene allenato.

Dopo di che, tramite l'istruzione

```
nltk.classify.accuracy(classifier, test_set)
```

è stato possibile valutare l'accuratezza del classificatore. Questa funzione confronta le classi predette dal classificatore rispetto alle classi preimpostate nel *test set*. L'accuratezza del classificatore creato è risultata al **71%**, che è un ottimo risultato. Naturalmente può essere migliorato ampliando il *dataset*.

Il modello deve poi essere salvato in modo tale che possa essere utilizzato.

Capitolo 6

Ottimizzazione

Dopo una serie di test di ricerca sull'applicazione, come anche spiegato in precedenza, è stato stimato che per analizzare 25 prodotti ci vogliono circa 3 minuti. Sono quindi state cercate delle soluzioni che potessero ottimizzare la ricerca in modo da restituire dei risultati all'utente nel minor tempo possibile.

6.0.1 Parallelizzazione ricerca

Per prima si è reso necessario parallelizzare la ricerca cioè cercare le caratteristiche richieste su più prodotti in modo contemporaneo. Quindi si è pensato di utilizzare un **Thread pool** di lunghezza variabile (in base alla disponibilità di risorse hardware). Ad ogni thread viene passata l'istanza del browser su cui aprire la pagina del prodotto ed il prodotto da analizzare. Il thread, se non trova tutte le caratteristiche richieste dall'utente nel nome del prodotto, analizza l'intera pagina del prodotto. Nello specifico, ciascun thread, è un **Future task** ovvero un oggetto che ritornerà un valore nel futuro, infatti ritornerà il prodotto se esso è conforme alle richieste oppure un JSON vuoto nel caso contrario. Per quanto riguarda invece l'utilizzo del Tread pool, si è deciso di utilizzare quello, invece che creare thread all'occorrenza, in modo tale da risparmiare ancora più tempo perchè i thread vengono creati una sola volta all'inizio e vengono riutilizzati "sprecando" così meno tempo. Applicando le ottimizzazioni descritte sopra si è riusciti a portare il tempo per analizzare 25 prodotti a circa 1/2 minuti. Era però ancora troppo.

6.0.2 Fast transmit

Ogni 5 prodotti conformi con le caratteristiche richieste dall'utente, essi vengo subito inviati come risultato all'applicazione. In questo modo l'utente, dopo circa 30 secondi, può già consultare dei prodotti. Nel frattempo, in background tramite altre richieste fatte dall'applicazione alla stessa risorsa server, vengono analizzati i prodotti rimanenti sempre utilizzando questo tipo di ottimizzazione. Quindi i prodotti man mano che passa il tempo aumentano.

Si è quindi passati dal restituire i prodotti in 3 minuti a restituire alcuni prodotti dopo solo 30 secondi. Naturalmente per le tempistiche della rete resta comunque

alto ma bisogna ricordarsi che l'utente non dovrà perdere ulteriore tempo perchè i risultati sono solo quelli conformi con quello che lui ha richiesto.

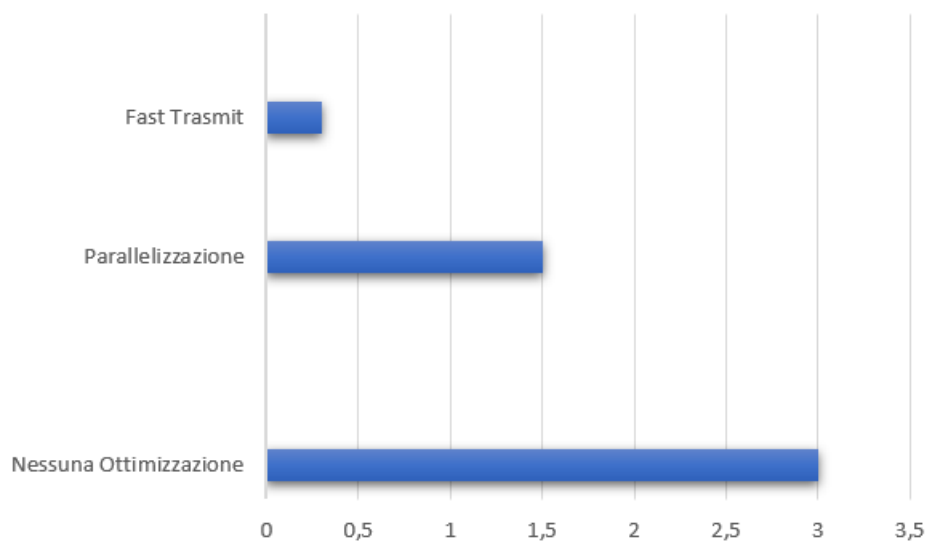


Figura 6.1: Grafico delle ottimizzazioni

Capitolo 7

Conclusioni

7.1 Difficoltà riscontrate

In questa sezione saranno esposte le maggiori problematiche riscontrate durante lo sviluppo dell'applicazione.

7.1.1 Problema del filtraggio

Il filtraggio è il punto cardine del progetto sviluppato. E' stato studiato nel dettaglio per garantirne la massima efficienza ma soprattutto per migliorare quella che è l'esperienza utente.

Durante la fase di studio, la difficoltà più importante che è stata riscontrata è quella di capire come cercare nei prodotti di un E-commerce le caratteristiche richieste dall'utente. Non è possibile cercare l'intera stringa inserita dall'utente nella barra di ricerca in quanto gli annunci hanno informazioni strutturate, mentre la stringa no. La soluzione a questo problema è possibile trovarla al paragrafo 5.2.4. La ricerca della suddetta ha richiesto un'analisi dettagliata riguardo alla composizione testuale degli annunci e alle possibili ricerche che un utente può fare.

7.2 Limitazioni

Una grande limitazione che è stata riscontrata deriva dall'utilizzo di **Selenium** seppur sia molto più veloce della tecnica utilizzata all'inizio ovvero **Beautiful Soup**.

Siccome, per poter navigare, usa delle finestre di un browser, la velocità complessiva dello scraping diminuisce molto. Infatti, parte del tempo viene persa solamente per aprire una nuova finestra del browser, mentre lo scraping del contenuto della pagina viene fatto in modo quasi istantaneo. Tutte le volte che bisogna aprire un link, cliccare un pulsante o simili, si perde molto tempo. Questa perdita di tempo deve essere sommata alla velocità della connessione perchè più questa è lenta più la velocità di risposta del browser diminuisce.

Quindi, la velocità dell'applicazione, dipende molto dalla potenza del calcolatore su cui è installato il server e la velocità della rete a cui è collegato.

7.3 Sviluppi futuri

7.3.1 Miglioramento filtraggio

Per prima cosa si potrebbe migliorare il modo in cui vengono cercate le caratteristiche del prodotto. Si dovrebbe, per ogni parola, trovare tutti i possibili sinonimi che i siti di E-commerce utilizzano e cercare anche quelli, oltre alle parole originali inserite dall'utente.

Infatti spesso, durante la prima fase di analisi delle ricerche, è stato riscontrato questo problema: cercando ad esempio "Asus, 16 GB RAM" gli E-commerce producono anche dei risultati in cui "RAM" è definita con il termine "DDR4" che è un tipo specifico di "RAM". Si potrebbero, quindi, inserire dei meccanismi di **Machine learning** che riescano ad imparare quali sono i sinonimi più utilizzati dagli annunci rispetto ad una parola cercata dall'utente.

7.3.2 Multiutenza

Un altro sviluppo sarebbe quello di inserire la possibilità di poter far usufruire l'applicazione da più utenti. Ora l'applicazione è pensata per un solo utente in quanto l'obiettivo del tirocinio era sullo scraping e sul filtraggio, non sulla multi-utenza già ampiamente analizzata durante tutto il corso di studi.

7.3.3 Miglioramento barra di ricerca

Infine, l'ultimo sviluppo potrebbe essere quello di evitare che l'utente vada ad inserire le caratteristiche del prodotto che vuole comprare separate da una virgola ma permettere, tramite una serie di menu a scelta multipla, di poter selezionare delle caratteristiche predefinite invece che doverle scrivere a mano. Le caratteristiche potrebbero essere mostrate all'utente in base anche alle sue ricerche passate quindi utilizzando l'apprendimento

7.4 Applicazioni simili

Purtroppo, o per fortuna, non esistono applicazioni simili che effettuano queste operazioni in quanto, viene dato per scontato, che i siti di E-commerce, permettano di effettuare il filtraggio dei risultati. In effetti è così, infatti è possibile scegliere, durante una ricerca, le caratteristiche che un prodotto dovrà avere. Però, quello che i siti di E-commerce fanno, è mescolare i risultati conformi alle caratteristiche con quelli che devono essere venduti secondo le politiche aziendali.

Questa applicazione è stata creata per mettere alla luce questi problemi e cercare di risolverli in modo efficiente.

7.5 Link GitHub

Durante tutto lo sviluppo dell'applicazione è stato utilizzato GitHub come supporto per il versioning del codice. E' possibile reperire il risultato della suddetta tesi al seguente link <https://github.com/fabriziosanino/bachelor-degree.git>

Bibliografia

- **Instant Developer**, *Filtri nelle app e nei siti: ecco perché creare quelli giusti è difficile, ma indispensabile*, <https://www.instantdeveloper.com/blog/ux-ui/filtri-nelle-app-e-nei-siti-ecco-perche-creare-quelli-giusti-e-difficile-ma-indispensabile/>
- **Real Python**, *Python and MySQL Database: A Practical Introduction*, <https://realpython.com/python-mysql/>
- **Real Python**, *Sentiment Analysis: First Steps With Python's NLTK Library*, <https://realpython.com/python-nltk-sentiment-analysis/>
- **Analytics Vidhya**, *Rule-Based Sentiment Analysis in Python*, <https://www.analyticsvidhya.com/blog/2021/06/rule-based-sentiment-analysis-in-python/>
- **Digital Ocean**, *How To Perform Sentiment Analysis in Python 3 Using the Natural Language Toolkit (NLTK)*, <https://www.digitalocean.com/community/tutorials/how-to-perform-sentiment-analysis-in-python-3-using-the-natural-language-toolkit-nltkstep-6-preparing-data-for-the-model>
- **Full Stack Python**, *Flask Web development*, <https://www.fullstackpython.com/flask.html>
- **Real Python**, *Beautiful Soup: Build a Web Scraper With Python*, <https://realpython.com/beautiful-soup-web-scraper-python/>
- **Analytics Vidhya**, *Performing Sentiment Analysis With Naive Bayes Classifier!*, <https://www.analyticsvidhya.com/blog/2021/07/performing-sentiment-analysis-with-naive-bayes-classifier/>
- **Geeksforgeeks**, *Abstract Classes in Python*, <https://www.geeksforgeeks.org/abstract-classes-in-python/>
- **Towards Data Science**, *SENTIMENTAL ANALYSIS USING VADER*, <https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664>

- **Towards Data Science**, *A Hitchhiker's Guide to Sentiment Analysis using Naive-Bayes Classifier*, <https://towardsdatascience.com/a-hitchhikers-guide-to-sentiment-analysis-using-naive-bayes-classifier-b921c0fb694>

Ringraziamenti

Un sentito grazie a tutte le persone che mi hanno permesso di arrivare fin qui e di portare a termine questa parte del mio corso di studi.

Grazie al mio relatore Segnan sempre presente, puntuale e disponibile.

Non posso non menzionare i miei genitori e mia sorella Federica che da sempre mi sostengono nella realizzazione dei miei progetti. Non finirò mai di ringraziarvi per avermi permesso di arrivare fin qui.

Ringrazio la mia fidanzata Cecilia per avermi trasmesso la sua immensa curiosità e il suo coraggio. Grazie per tutto il tempo che mi hai dedicato. Grazie perché ci sei sempre.