UNIVERSITÀ DEGLI STUDI DI BARI "ALDO MORO"

DIPARTIMENTO DI INFORMATICA

Corso di Laurea in Informatica

Tesi di Laurea in Modelli e metodi per la sicurezza delle applicazioni

Web 5.0: Sistema di raccomandazione emozionale per la soddisfazione dell'utente

Relatore:
Prof. Donato Impedovo

Co-relatore:
Dott. Vincenzo Gattulli

Laureando:
Gianmarco Moresi

ANNO ACCADEMICO 2020/21

L'informatica non riguarda più i computer. Riguarda la vita.

Nicholas Negroponte

Abstract

Per aumentare l'accuratezza delle raccomandazioni o far nascere nuovi progetti con l'obiettivo di migliorare la vita dell'utente, è nato il web 5.0.

Il web 5.0, che attualmente è ancora in via di sviluppo, permetterà ai sistemi online e non, di poter rilevare, tramite determinate tecnologie e metodi, l'emozione dell'utente e poterla utilizzare per effettuare raccomandazioni o aiutare l'utente se si dovesse trovare in uno stato emotivo fragile. L'emozione potrebbe essere catturata in tanti modi diversi: scattando una fotografia al viso dell'utente rispettando la privacy e con consenso esplicitato dall'utente, oppure tramite fotografie scattate dal proprio smartphone, fotografie caricate online su un social network, messaggi e stati d'animo condivisi su un social network come Facebook, Instagram, Twitter ecc… Potrebbe essere una grande evoluzione nel mondo di internet perché potrebbe non solo a raccomandare prodotti come prima citato, ma anche ad aiutare persone in difficoltà emotiva, come ad esempio un lutto, bullismo online e non, una relazione che non è andata a buon fine e molte altre motivazioni.

Questo progetto nasce dall'obiettivo di poter utilizzare l'emozione rilevata dall'utente tramite una rete neurale profonda 50 layers, e consigliare dei titoli, come ad esempio dei film o serie tv, che potrebbero aiutare l'utente a star bene qualora il suo stato emotivo sia triste oppure continuare a mantenere lo stato emotivo attuale.

Per poter usufruire delle raccomandazioni utilizzando l'emozione, il sistema chiede esplicitamente all'utente il consenso di effettuare una fotografia al suo volto, dopo tale consenso, se il risultato è positivo, il sistema effettua una fotografia al volto dell'utente e viene memorizzata all'interno del server temporaneamente. Subito dopo lo scatto, la foto viene inviata in input ad una rete neurale conosciuta come ResNet50, la quale tramite diverse librerie e dopo aver eseguito un addestramento su diverse fotografie di altri utenti memorizzati in un dataset specifico, rileva la faccia dell'utente e di conseguenza l'emozione attuale dell'utente. Dopo aver rilevato l'emozione, la foto scattata inizialmente dal sistema principale viene eliminato dal server per contribuire alla privacy dell'utente, successivamente l'emozione viene inviata al sistema principale che la utilizzerà per calcolare le raccomandazioni dei titoli; a seconda dell'emozione rilevata verranno presi in considerazione solo titoli con determinati generi, ad esempio: comici, azione ecc…

I titoli raccolti non sono adatti immediatamente ad essere mostrati all'utente, quindi per poter essere mostrare all'utente, vengono richiamati dei moduli specifici scritti in JavaScript, dopo vengono richieste informazioni aggiuntive ai titoli raccolti per poter creare

delle schede dei film o serie tv composte da informazioni adatte ad essere mostrate in una interfaccia utente, come ad esempio la trama, il trailer ecc…

Indice

Sommario

Capitolo 1 – Introduzione

Introduzione

Il web ha sempre vissuto grandi cambiamenti da quando è stato lanciato il primo satellite artificiale *Sputnik1*, in pochissimi anni abbiamo visto come il web si sia evoluto dalla prima versione: la 0.0 ovvero lo sviluppo di internet, fino ad arrivare all'attuale 5.0.

Il web 1.0 chiamato anche *Read-only*, veniva utilizzato da chiunque avesse un calcolatore con connessione ad internet e un browser solo per la lettura di informazioni caricate online su un semplice sito web composto da sole pagine HTML, quest'ultimo è un linguaggio di markup ideato dal fisico e informatico *Tim Berners Lee.*

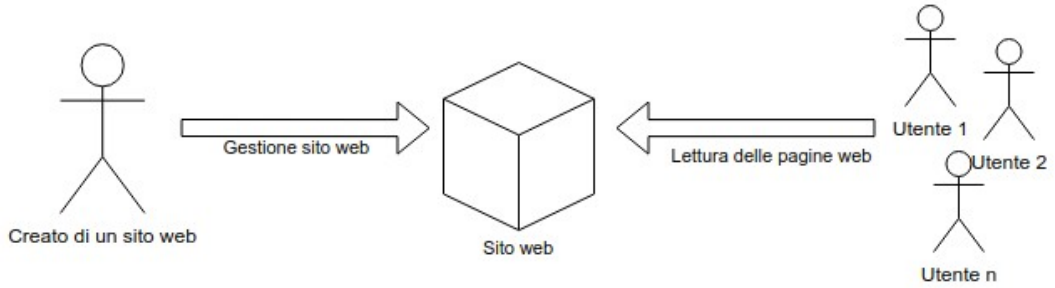

Figura 1: Schema del web 1.0

Per permettere agli utenti la comunicazione attraverso il web, pochi anni dopo nacque il web 2.0. Nacquero non solo i primi social network, ma ogni utente era libero di creare un forum su argomenti specifici e blog personali dove condividere storie, foto e altre informazioni. Un paio di anni dopo nasce il web 3.0 dove i siti web non sono composti solo da pagine HTML ma anche da database, intelligenza artificiale per individuare la necessità dell'utente e motori di ricerca per effettuare interrogazioni con un linguaggio naturale. Per superare la barriera dei siti web visitati solo su desktop, quindi sul computer, il web 4.0 porta con se la possibilità di sviluppare versioni di siti web anche per i dispositivi mobile per raggiungere più persone oppure per permettere alle persone di accedervi in qualsiasi momento e luogo.

Attualmente stiamo vivendo il web 5.0 chiamato anche web *simbiotico* o anche web sensoriale ed emotivo. Il web 5.0 è ancora in via di sviluppo, ma sappiamo che sarà in

grado di comunicare con gli utenti come se fosse una persona reale, ci aiuterà nelle decisioni di tutti i giorni, a riconoscere le nostre emozioni e di conseguenza comportarsi in modo differente, un vero e proprio assistente personale. Essendo ancora in via di sviluppo, non percepisce sentimenti ed emozioni degli utenti.

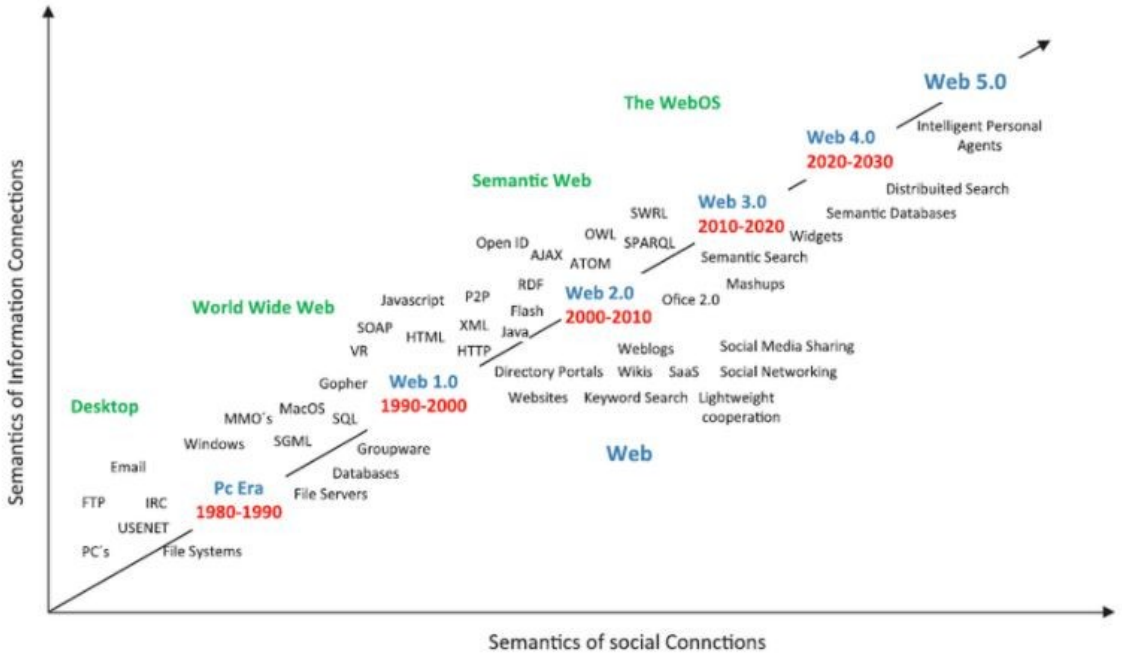

Figura 2: Evoluzione di internet fino al web 5.0

Il completo sviluppo del web 5.0 potrebbe portare grandi progressi ad Internet e a tutti i servizi offerti tramite esso. Ogni giorno le persone online condividono in piena liberà i proprio sentimenti tramite messaggi, foto e video. Rilevare l'emozione di un utente online, potrebbe aiutare tanti e-commerce online a capire meglio i gusti degli utenti, apportare modifiche ai prodotti o cambiando i prodotti venduti [26].

L'analisi automatica dell'emozione non deve essere utilizzata solo nell'ambito commerciale, per cambiare i prodotti venduti o cambiare la strategia di vendita, ma può essere utilizzata e sfruttata anche in ambito medico con l'aiuto dei social network. Dall'anno 2020 ad oggi la popolazione mondiale ha subito gravi crolli emotivo a causa della pandemia per il virus Covid-19, per aiutare i pazienti da problemi mentali come ad esempio la depressione perché isolati in casa, oppure a causa della perdita di una persona vicina al paziente, gli operatori sanitari possono rilevare l'emozione del paziente tramite l'analisi automatica dell'emozione sfruttando internet e le informazioni condivise online dai pazienti sui social network [27].

Dato l'incompleto sviluppo del web 5.0, in questa tesi verranno discusse varie soluzioni per permettere l'avanzamento dello sviluppo del web sensoriale ed emotivo. Il sistema

che verrà spiegato in questa tesi porterà alla creazione di un sistema online in grado di rilevare l'emozione dell'utente tramite una fotografia scattata con la webcam del computer dell'utente, e dopo aver rilevate l'emozione dalla foto, il sistema raccomanderà dei titoli di film o serie tv da guardare all'utente per migliorare o continuare ad avere l'emozione rilevata dal sistema. Se ad esempio l'utente è felice, verranno consigliati dei titoli comici per mantenere l'utente felice o migliorare la sua felicità.

Capitolo 2– Stato dell'arte

Stato dell'arte

Il rilevamento automatico delle emozioni possono salvare la vita a migliaia di persone e aiutarle a stare bene. Possono garantire la sicurezza a persone che magari durante la guida si sentono male, si addormentano, si spaventano o si distraggono; possono aiutare ragazzi, adulti a migliorare il proprio stato d'animo, individuando pazienti depressi, o situazioni anomale. Ma non solo è possibile applicare il riconoscimento automatico delle emozioni nel campo medico ma anche nel campo del commercio. Riconoscere le emozioni di utenti o clienti possono aiutare a modificare e creare nuove strategie di marketing e aumentare le vendite dei prodotti.

Non sempre riconoscere l'emozione di un utente è semplice, infatti il web 5.0, ovvero il web che si occupa di rilevare i sentimenti dell'utente e le emozioni è ancora in via di sviluppo. La prime difficoltà nel creare un sistema che rileva l'emozione dell'utente è l'utilizzo di un dataset composto da immagini rilevanti che raffigurano l'utente. Il dataset pubblico utilizzato in questo progetto è molto grande, ma non significa che le immagini al suo interno sono realmente rilevanti per rilevare l'emozione dell'utente, anche perché una volta addestrata la rete neurale, le immagini che verranno poi date in input una volta caricato il sistema online dovranno avere la stessa qualità delle immagini utilizzate nel dataset, e non è sempre semplice perché ogni utente ha un dispositivo differente con webcam diverse, e di conseguenza la qualità della foto scattata è diversa cambiando il risultato della predizione.

Per la costruzione di un sistema in grado di raccomandare elementi sulla base delle emozioni e delle preferenze degli utenti, è stata effettuata una ricerca approfondita online su diverse metodologie utilizzate per costruire recommender system *content based*, *collaborative filtering* e sulla base delle emozioni catturate tramite social network o tramite espressione facciale dell'utente.

Gli algoritmi di raccomandazione *content based* sono delle tecnologie per suggerire i contenuti più adatti ai propri utenti e corrispondere alle preferenze dell'utente [3]. Invece gli algoritmi di raccomandazione *collaborative filtering* si basano sul fare previsione sulle preferenze un utente specifico in base alle preferenze di un'insieme di utenti considerati simili all'utente specifico, ovvero che hanno preferenze in comune tra di loro [1]. Per avere dei miglior risultati sulle varie predizioni, unisco gli sforzi dei due tipi di algoritmi [2], *content based* e *collaborative filtering*, andando a consigliare più elementi all'utente oppure dare importanza agli elementi raccomandati sia dall'algoritmo *content based* che da quello *collaborative filtering.* J. Bobadilla et all. [1] hanno lavorato ad un

algoritmo di raccomandazione *collaborative filtering* per un sistema di l'e-learning assegnando un peso (valore) ad ogni utente in base alla sua conoscenza, dando pesi maggiori ad utenti che hanno più conoscenze rispetto ad altri (tramite i risultati positivi ricevuti durante i vari test). I suggerimenti dati dal sistema si baseranno sulla somiglianza tra l'utente attivo e utenti simili, e gli elementi votati da utenti con un maggior peso saranno più significativi rispetto ad elementi votati da utenti con un peso minore [1]. Al giorno d'oggi, la quantità dei film è aumentata in modo esponenziale e per l'utente finale è diventato difficile cercare dei film che rispecchiano i suoi gusti. M. J. Awan et all. [4] hanno proposto un algoritmo per la raccomandazione di film migliori agli utenti tramite un approccio *collaborative filtering* basato sul modello ALS (*alternating least squared), un algoritmo* che permette la fattorizzazione di una matrice per il calcolo distribuito e parallelo, utilizzata per raccomandare film più votati all'utente. Il sistema raccomanda i film all'utente in base alla categoria del film ricercato per ultimo, e tramite questi dati, il motore di raccomandazione viene istruito per suggerimenti futuri. Con l'utilizzo del modello ALS e una tecnica di filtraggio collaborativo, sono stato risolti problemi di *cold start*, di dispersione e scalabilità [4].

I sistemi di raccomandazione possono essere applicati in qualsiasi campo, anche nei servizi di trasporto multimodale, adattandosi alle esigenze dell'utente durante il viaggio attraverso i dati, statistiche e tendenze estratte dagli utilizzi passati del sistema. A. Rivas et all. [12] hanno ideato un sistema formato da tecniche che permettono di migliorare l'esperienza multimodale dell'utente analizzando le varie preferenze, e per fare ciò il sistema è dotato di una tecnica per l'estrazione di informazioni degli utenti da social network come Twitter. Con i vari dati estratti il sistema elabora un profilo con le preferenze divise per categorie di un utente specifico. Il sistema raccoglie due tipologie di dati: (1) un insieme di tweet anonimi per ogni categorie di preferenze scelte, utilizzati per addestrare il classificatore dei dati; (2) informazioni sull'utente per l'analisi delle preferenze. I dati estratti vengono elaborati per la classificazione, andando ad applicare tecniche di pulizia, pre-elaborazione ecc., come ad esempio la rimozione di contenuti che non forniscono informazioni per l'elaborazione del linguaggio eliminando URL, menzioni ad utenti, punteggiatura ecc.. Successivamente viene eseguito il *Topic Modelling,* uno strumento per l'estrazione del testo e utilizzato in modo frequente per scoprire strutture semantiche nascose nel testo estratto [11]. Dopo l'addestramento, è possibile capire gli argomenti chiave per un determinato profilo di un utente sulla base dei tweet scritti o condivisi [12].

I sistemi di raccomandazione possono effettuare raccomandazioni andando ad effettuare l'analisi semantica del testo estratto da pagine web, cosi da migliorare l'accuratezza delle raccomandazioni. I. T. Afolabi et all. [14] hanno presentato un sistema di raccomandazione di articoli in vendita su un negozio online utilizzando l'analisi semantica del testo estratto da pagine web tramite un crawler. La metodologia utilizzata si basa in due

fasi: (1) la prima fase è il pre-*processing* di dati testuali estratti utilizzando la combinazione di ontologie, sviluppate appositamente per il sistema e esistenti. I set di dati testuali vengono estratti dalle pagine web tramite un web crawler, definito come software o anche *script* che naviga tra le pagine web seguendo i vari link presenti in una pagina web e scarica documenti in modo metodico e automatizzato. I web crawler vengono chiamati anche robot, spider o worm [13]. I documenti recuperati online seguono un processo di pre-processing semantico utilizzando ontologie di dominio. (2) La seconda fase si concentra nell'utilizzo dell'algoritmo *Naïve Bayes* per le raccomandazioni. Il classificatore *Naïve Bayes* e' un semplice classificatore probabilistico basato sull'applicazione del teorema di Bayes con una forte assunzione di indipendenza. Il classificatore predice se un dato elemento appartiene ad una determinata classe [14]. Per aumentare le forme di istruzione e offrire agli studenti un apprendimento di alta qualità J. Jordán et all. [2] hanno proposto un algoritmo di raccomandazione ibrido utilizzando algoritmi di raccomandazione *content-based* e *collaborative filtering* per la raccomandazione di video istruttivi per docenti e studenti, permettendo non solo di raccomandare video di qualità allo studente iscritto, ma anche a utenti non iscritti al sistema, consigliando video simili a quello che sta guardando. Andando ad unire i due tipi di algoritmi, è possibile risolvere il problema del *cold start* e la mancanza *serendipità,* ovvero elementi consigliati rilevanti e nuovi per l'utente, ma anche differenti dagli elementi che l'utente ha votato. La scelta di un sistema ibrido può migliorare l'accuratezza in raccomandazioni. I due algoritmi sono composti da differenti moduli utilizzati per gestire differenti campi all'interno del sistema: l'algoritmo *content-based* tiene conto dell'attività dell'utente, quindi un modulo basato sul profilo, e un altro componente tiene conto della tipologia di video che guarda in un determinato istante di tempo; invece l'algoritmo *collaborative filtering* offre suggerimenti che si basano sulle preferenze di utenti simili all'utente corrente [2].

Cosi come Amazon, Alibaba e molti altri colossi di e-commerce, è possibile costruire un sistema di raccomandazione ibrido per aiutare i potenziali acquirenti ad effettuare le loro decisioni di acquisto. T. Tran e R. Cohen [5] hanno proposto un sistema di raccomandazione ibrido per il commercio elettronico, l'architettura da loro proposta si divide in più componenti: (1) Agente di interfaccia interattivo che fa da intermediario tra l'utente e il sistema di raccomandazione, sceglie il miglior sistema di raccomandazione da utilizzare e coordinate le operazioni; (2) *knowledge-based engine*, permette di generare raccomandazioni da inoltrare all'utente tramite il primo componente; (3) *knowledge base* del prodotto di dominio, ovvero tutti i vari prodotti presenti all'interno del sistema; (4) *Collaborative filtering engine*, il quale crea un profilo di interesse per l'utente; (5) Database dei voti dell'utente per gli articoli, è possibile votare l'articolo tramite degli aggettivi {*eccellente, buono, sopra la media, nella media, sotto la media*} che poi il sistema convertirà in punteggio. Tale sistema avrà tutti i vantaggi di un sistema di racco-

mandazione collaborative filtering e di un sistema content based, e risolverà tutti i principali svantaggi dei due sistemi di raccomandazione. [5].

Nel mondo della chimica, le varie entità/composti chimici stanno aumentano sia in numero che in complessità, generando grandi insieme di dati troppo profondi per essere analizzati dai vari ricercati scientifici. Il machine learning arriva in supporto anche ai ricercatori scientifici aiutandoli a scegliere i migliori composti da studiare per le loro ricerche. M. Barros et all. [6] hanno sviluppato un sistema di raccomandazione ibrido per identificare componenti chimici di interesse per i ricercati scientifici. Il modello da loro proposto è composto da algoritmi *collaborative filtering* (*Alternating Least Squares e Bayesian Personalized Ranking)* per feedback impliciti, i quali sono ampiamente utilizzati all'interno dei metodi di *collaborative filtering* per raccomandare elementi. [7] I feedback impliciti sono delle azioni che l'utente compie all'interno del sistema, come ad esempio: *cliccare su un articolo*, *leggere la descrizione di un articolo ecc. , i* quali possono avere dei significati di preferenza dell'utente verso un elemento. Invece per il modello content based viene utilizzato un algoritmo che permette di effettuare la similarità semantica tra componenti chimici utilizzando l'ontologia di ChEBI [6]. La maggior parte degli attuali recommender system musicali non valutano fattori importanti come la selezione congiunta degli articoli e le caratteristiche degli articoli, dovrebbero essere prese in considerazione anche la percezione e esperienza dell'utente. A. Gatzioura et all. [10] hanno proposto un sistema di raccomandazione di canzoni ibrido che permette di costruire playlist per l'utente. Le raccomandazioni del sistema sono concentrate ad uno specifico concetto semantico dedotto dalla playlist avviata. In questo esperimento viene utilizzato la *Latent Dirichlet Allocation* (LDA), ovvero un modello Bayesiano gerarchico a tre livelli utilizzato per la raccolta di dati discreti [8], viene applicato prima all'intero set di playlist utilizzate in precedenza e poi per ogni nuova playlist, dopo viene applicato il *Case-Based Reasoning (CBR),* un tipo di ragionamento analogico che lavora con l'esperienza precedente dove è possibile ricavare soluzioni a un nuovo problema ecc. [9], su un set raffinato di playlist. Quindi il sistema presentato da A. Gatzioura et all. segue l'idea del CBR, ovvero che i problemi simili hanno soluzioni simili. Data una nuova playlist, vengono trovate playlist precedenti più simili e utilizza le loro caratteristiche per costruire una nuova playlist con brani più appropriati [10]. In un recommender system è possibile generare predizioni di articoli utilizzano anche più risorse web , infatti S. Bostandjiev et all. [15] hanno presentato un sistema di raccomandazione ibrido interattivo che genera previsione di elementi lavorando con preferenze estratte da più social network e risorse del web semantico, come ad esempio Facebook, Twitter e Wikipedia. Il sistema lavora con algoritmi di raccomandazione *content based, collaborative filtering* e tramite un interfaccia interattiva utilizzata per spiegare all'utente le varie raccomandazioni suggerite. Il sistema si divide in 3 layers: (1) *Profile layer*, un livello che contiene le preferenze dell'utente che possono variare da dominio a dominio, nel sistema presentato da S. Bostandjiev et all. vengono indicati gruppi musicali o contanti. In

questo livello è possibile regolare il peso per ogni preferenza; (2) *Context Layer*, contiene elementi proveniente da differenti sorgenti, come ad esempio i vari social utilizzati, che possono essere usati per produrre le raccomandazioni, anche qui sono presenti regolatori per i modificare il livello di preferenza per ogni elemento; (3) *Recommendation Layer*, contiene le varie raccomandazioni per ogni sorgente utilizzata nel layer precedente, ordinati per rilevanza [15].

Al giorno d'oggi gli strumenti dominanti nel mondo sono i social network dove le persone comunicano, condividono notizie, immagini e video. Le immagini e video che vengono condivise contengono grandi quantità di dati di alta qualità per capire gli interessi degli utenti rispetto a social network basati solo sul testo. K. Kim et all. [16] hanno proposto un framework per scoprire gli interessi degli utenti iscritti ad un social network in base alle immagini caricate su di esso. Il framework si basa su *Latent Dirichlet Allocation (LDA)* e si divide in 3 moduli: (1) modulo per la generazione di grafi di argomenti generali, che estrae informazioni rilevanti dalle immagini tramite l'utilizzo di MS Azure e gli hashtag inseriti in ogni foto; (2) modulo per l'estrazione di grafi di argomenti individuali e (3) modulo per la raccomandazione basato sulla somiglianza del grafo, che effettua il calcolo della somiglia del grafo tra utente target e gli altri utenti; sulla base della simiglianza del grafo, ogni utente o argomento viene valutato nella prospettiva dell'utente target e avviene la raccomandazione [16]. S. K. Addagarla e A. Amalanathan [17] hanno presentato un sistema di raccomandazione che permette di suggerire all'utente articoli con un immagine di presentazione simile ad un altro. Viene effettuata l'estrazione e la generazione delle caratteristiche dell'immagine utilizzando tecniche di deep learning ovvero l'architettura SqeezeNet basata sulla rete neurale convoluzionale (CNN) per il processo di ingegneria della funzionalità e costruendo un albero dell'indice utilizzando l'algoritmo ANNOY (*Approximate nearest neighbors Oh yeah)* e tramite la misura della distanza, vengono recuperati i top-N elementi dal risultato. Questo approccio ha una precisione del 96,22% rispetto ad altri approcci [17]. Per rendere un AI trasparente e affidabile, è essenziale spiegare le raccomandazioni fornite all'utente. A. Ghazimatin et all. [18] hanno presentato un framework chiamato Elixir dove il feedback degli utenti sulle spiegazioni fornite dal sistema sui suggerimenti viene utilizzato per apprendere le preferenze dell'utente. Il framework effettua la raccolta dei feedback, ovvero viene chiesto all'utente attivo se gli piace o non gli piace la somiglianza tra due elementi suggeriti, e il feedback viene codificato con un numero di gradimento {+1,-1, 0}, 0 se non ha espresso giudizio. Viene costruita una matrice con i vari feedback dell'utente e inviata al recommender, l'obiettivo è imparare i vettori di preferenza per ogni utente che possono essere poi combinati con le varie rappresentazioni di articoli esistenti nel sistema e produrre suggerimenti appropriati per l'utente [18]. Utilizzando i sistemi di raccomandazione possiamo suggerire elementi all'utente in base alle loro esigenze e interessi in modo efficace, ma c'è un problema molto importante per i sistemi di raccomandazione: il *cold start,* ovvero un problema che si verifica quando il sistema non è in

grado di fare alcuna inferenza per utenti o articoli perché non ha ancora abbastanza informazioni a riguardo. Q. Y. Shambour et all. [19] hanno proposto un approccio chiamato *Trust-Semantic* potenziato con *Multi-Criteria CF (TSeMCCF),* che permette di sfruttare le relazioni di fiducia e valutazioni multi-criterio degli utenti e le relazioni semantiche degli elementi presenti all'interno del sistema per ottenere risultati efficaci quando non abbiamo molte informazioni [19]. Per migliorare l'accuratezza nei recommender system, è comune utilizzare le informazioni collaterali. L. Chen et all. [20] hanno proposto un sistema *collaborative filtering memory-based,* ovvero, dato un utente attivo, il sistema trova gli utenti che hanno preferenze simili all'utente attivo e aggrega le valutazioni degli utenti trovato sugli elemento che non sono valutati dall'utente attivo [21]. Nel sistema proposto, vengono effettuate le raccomandazioni di elementi tramite l'utilizzo di categorie di oggetti, in questo caso generi cinematografici. Inizialmente vengono ricavate le preferenze dell'utente per ogni elemento utilizzando le valutazioni esplicite, con lo stesso metodo, vengono ricavate le valutazioni complessive per ogni articolo, poi viene identificato quanto ogni elemento appartiene ad un certo genere. I due vettori ricavati vengono combinati per ottenere un'ulteriore matrice utente-elemento-peso, utilizziamo il peso della matrice utente-elemento per effettuare spiegazioni all'utente del perché è stato consigliato quel determinato elemento. Questo sistema utilizza la matrice utente-elemento-peso, citata prima, anche per risolvere il problema della scarsità associato alla somiglianza basata sulla correlazione. Il modello proposto ha prestazioni significative rispetto ai metodi di base, e ha una migliore complessità computazione per le previsioni rispetto ad un sistema kNN (k-nearest neighbor) [20].

Il web è in rapida evoluzione ogni giorno, arrivando ad analizzare anche i sentimenti degli utenti per creare un Web sensoriale emotivo chiamato Web 5.0 [22]. È possibile sfruttare tali emozioni inserendoli in un sistemi di raccomandazione ed effettuare delle predizioni di elementi che coincidono con le emozioni dell'utente. Ad esempio, in un sito per la condivisione di foto, se l'utente è felice il sistema di raccomandazione potrebbe suggerire delle immagini divertenti, oppure in un sito di film streaming, è possibile suggerire dei film o serie TV in base allo stato d'animo dell'utente. I social network sono delle piattaforme che permettono alle persone di condividere i propri pensieri, video, immagini e notizie con gli amici, parenti, ecc. Tramite i media e i testi condivisi sui social network è possibile rilevare e analizzare i sentimenti espressi dalle persone e utilizzarli per generare raccomandazioni all'utente attivo. K. Sailunaz e R. Alhajj [23] hanno presentato un sistema in grado di rilevare i sentimenti delle persone tramite i post condivisi all'interno del social network Twitter. Per la raccolta dei dati, ovvero i post condivisi, le informazioni dell'utente etc. sono stati utilizzate le API di Twitter e anche metodi per il web scraping all'interno del quale venivano presi in considerazione solo post originali dell'utente e non condivisi da un'altra pagina sul profilo dell'utente, ovvero i *retweet*. I post raccolti seguivano una fase di pre-processing, ovvero di pulizia dove venivano eliminate le parole inutili o la punteggiatura al post, come menzioni degli

utenti, tutte le faccine 'digitali', ecc. Successivamente per preparare il dataset, ogni commento è stato associato ad un sentimento come: rabbia, paura, gioia ecc. Per la classificazione dei post alle rispettive classi di sentimento ed emozioni, è stato utilizzato un classificatore conosciuto come "Naïve Bayes" dimostrando risultati nettamente migliori rispetto ad altri approcci di apprendimento automatico con l'aggiunta del *k-fold cross validation* [23]. Nella vita di tutti i giorni le emozioni hanno un ruolo importante non solo tra la comunicazione e interazioni tra due essere umani, ma anche tra l'interazione uomo e computer. Con l'aiuto della computer vision e la psicologia è possibile individuare tramite video o immagini i sentimenti e l'emozioni degli utenti che utilizzano un sistema, permettendo di migliorare il suo stato di salute, la sua emozione e controllarlo qualora ci fossero dei cambiamenti d'umore. L. Do et all. [28] hanno lavorato ad un'insieme di modelli basati sulle reti neurali che permettono il riconoscimento dell'emozione tramite un video diviso in fotogrammi. Le espressioni che vengono rilevate dal modello sono: rabbia, disgusto, paura, felicità, tristezza, sorpreso e neutro. Nel progetto vengono utilizzate diverse reti neurali: la prima rete neurale adottata è una CNN multi livello detta anche MLCNN che estrae le caratteristiche del volto ad ogni fotogramma, la seconda rete è la 3DCNN che permette l'analisi delle informazioni temporali, queste reti neurali compongono il primo modello dei 3 utilizzati. Il secondo modello è una combinazioni di 2 moduli: VGG-Face e della memoria a breve termine LSTM. Il terzo modello invece utilizza la rete Xception per codificare le caratteristiche con la statistica. L'esperimento ha portati grandi risultati con differenti video, raggiungendo un'accuratezza del 90% [28]. Per identificare l'emozione di un utente vengono coinvolti molti componenti differenti quali il corpo, la postura, le espressioni del viso e la voce. L'analisi della voce chiamata *speech analytics* è una sfida recente aperta in tutte le applicazioni e campi, come ad esempio nel campo della salute, videogiochi e nel campo dei call center, infatti comprendere meglio i bisogni di un cliente al telefono significa per l'azienda gestire meglio il cliente e di conseguenza maggiori benefici per il cliente e per l'azienda. A. Agrima et all. [29] hanno presentato un metodo per l'estrazione delle caratteristiche della voce basato sull'approccio pseudo-fonico. L'obiettivo del lavoro è quello di andare ad estrarre le caratteristiche in base a diversi segmenti come le unità sillabiche per rimediare al vinco della variazione linguistica. Vengono effettuate le varie estrazioni di indizi utilizzando vari metodi di elaborazione del segnale. Due descrittori uno a basso livello chiamato LLD e uno ad alto livello chiamato HLD sono stati ottenuti da un segnale vocale che è stato etichettato da quattro emozioni: rabbia, tristezza, gioia e neutrale. Ogni sequenza audio per essere scelta deve soddisfare il criterio di udibilità e passare attraverso un processo diviso in varie fasi:

- modulare il segnale secondo un gruppo di variabili contestuali, culturali e linguistiche che ha l'obiettivo di comunicare emotivamente o meno;
- catturare il segnale prodotto dal parlante;

- annotare il segnale;
- segmentare il segnale in forma sillabica;
- estrarre le varie caratteristiche acustiche utilizzando tecniche per l'elaborazione del segnale;
- classificare il suono utilizzando un algoritmo K-NN.

La raccolta delle informazioni sono state acquisite da discorsi recitati e non da conversazioni reali o spontanei. Le varie situazioni analizzate comprendono vari contesti come scene interne,esterne, monologhi e dialoghi. I vari file audio raccolti sono stati convertiti in formato .wav e tagliata ala struttura sillabica, poi è stato elaborato nuovamente tramite una classificazione di tipo sillabica e memorizzata nella stessa cartella. Vengono estratte caratteristiche come formanti, altezza ed energie in sei bande dal segnale vocale. I vari risultati per questo esperimento oscillavano tra l'86,21% al 78,95% per le emozioni positive, quindi neutro e gioia, e emozioni negative come rabbia e tristezza. L'algoritmo di classificazione utilizzato è un algoritmo K-NN [29].

Capitolo 3 – Progettazione

Progettazione

3.1 La progettazione del recommender system

L'obiettivo principale di questo progetto è permettere di migliorare l'emozione dell'utente tramite un sistema in grado di fotografare il voto dell'utente utilizzando la webcam del computer, e rilevando tramite una rete neurale l'emozione. Tramite l'emozione rilevata, il sistema permetterà tramite l'ausilio di informazioni aggiuntive di consigliare all'utente dei titoli di film e serie TV in grado di migliorare l'emozione o continuare a mantenere l'emozione attuale. Il sistema non sarà solo in grado di rilevare l'emozione dell'utente e consigliare dei titoli, ma sarà in grado di raccomandare altre titoli sulla base delle preferenze dell'utente, sia tramite feedback espliciti, come ad esempio un mi piace su un titolo mostrare dal sistema oppure tramite feedback impliciti, come il passaggio del mouse sulla locandina, lettura della trama ecc…

Gli elementi che verranno suggeriti dal sistema sono film e serie TV, ed è stato scelto il dataset pubblico *Netflix movies and TV shows*, contenente tutti i titoli presenti sulla piattaforma Netflix con diverse proprietà dei vari titoli come: *show_id, type, title, director, cast ecc...* Grosso problema per che può portare l'utente al disinteresse del sistema è il dataset non aggiornato di film e serie TV, gli ultimi aggiornamenti risalgono al 2020, e Netflix rilascio ogni giorno film e serie TV nuove, quindi anche una settimana di ritardo nell'aggiornamento del dataset è molto rilevante, in questo caso è 1 anno.

Il sistema però può essere riadattato su dataset differenti come ad esempio un dataset di prodotti di vario genere (elettronica, abbigliamento, ecc…). Il problema è adattare le varie query applicate al dataset attuale con il nuovo dataset, si potrebbe cercare un dataset con le stesse tipologie di colonne al dataset attuale, ad esempio: descrizione, genere, anno di rilascio, titolo ecc…

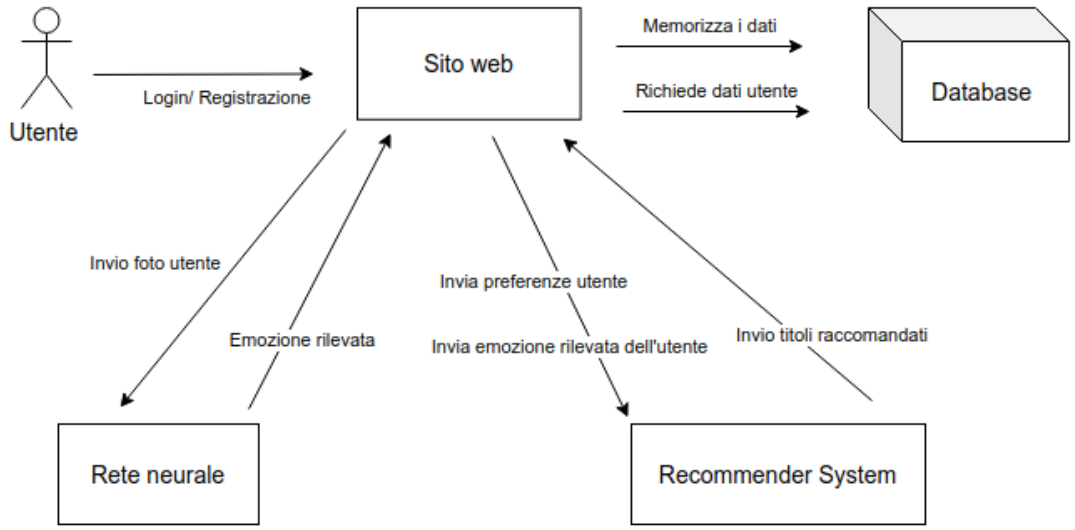

Figura 3: Schema principale del sistema

Prima di effettuare la raccomandazione in base all'emozione dell'utente, è bene progettare il sistema di raccomandazione basato sul contenuto dell'elemento che servirà a consigliare all'utente dei titoli simili a titoli dove ha espresso un gradimento. In questo caso il recommender system content based composto da una matrice TF-IDF che ci servirà per costruire un grafo. La matrice TF-IDF è il prodotto della frequenza dei termini, in questo caso TF e l'inverso della frequenza dei documenti IDF.

$$tf-idf(t,d)=tf(t,d)*idf(t,d)$$

tf(t,d) è la frequenza del termine *t* nel documento *d. L'*inverso della frequenza dei documenti viene calcolato come segue:

$$idf(t,d)=\log\frac{n_d}{1+df(d,t)}$$

dove n_d è il numero totale di documenti e df(d, t) è il numero di documenti d che contiene il termine t. Dopo aver calcolato la matrice tf-idf, è possibile effettuare la ricerca di elementi simili all'elemento target in base alla loro descrizione utilizzando la similarità del coseno.

Con i dati dei vari titoli che compongono il dataset, e la matrice tf-idf, è possibile costruire un grafo indiretto; ogni nodo può essere: titolo del film, attori o registi, categorie, paese, descrizione e similarità ovvero film simili in base alla descrizione del film target. Invece gli archi vengono etichettati in base al collegamento, ovvero se un attore ha reci-

tato in un film, l'arco verrà etichettato con 'acted_in', se un titolo ha un determinato genere verrà collegato il film o serie TV con il genere e etichettato con 'cat_in'; per il regista l'arco si chiamerà 'directed' e cosi viene fatto per tutte le relazioni.
Infine, per effettuare la vera e propria raccomandazione, viene esplorato il grafo in 3 fasi:

- vengono esplorati i vicini del film target, in questo caso la lista di attori, registi, paese e la categoria;
- vengono esplorati i vicini di ogni vicino, scoprendo i film che vengono condividono i nodi con il film/serie tv di partenza;
- poi viene effettuato il calcolo della misura di *Adamic Adar*

La misura di Adamic Adar viene utilizzata per effettuare il calcolo della vicinanza tra nodi in base ai loro vicini in comune. Più la misura calcolata è alta, più i due nodi sono vicini.

$$adamicAdar(x,y) = \sum_{u \in N \cap N(y)} \frac{1}{\log(N(u))}$$

Dove:

- x e y sono due film o serie tv (nodi);
- N(u) è la funzione che dato un nodo in input restituisce tutti i nodi adiacenti

Se i nodi *x* e *y* condividono un nodo *u* che ha molti nodi adiacenti, il nodo *u* non è rilevante perché $\frac{1}{\log(N(u))}$ non sarà alto. Se invece i nodi *x* e *y* condividono un nodo *u* che non ha molti nodi adiacenti, allora la frazione sarà alta e il nodo *u* sarà rilevante.

Per comunicare i film o serie TV dell'utente al recommender system vengono utilizzati differenti componenti: l'interfaccia utente del sito web che permette tramite moduli javascript di salvare e prelevare le preferenze dell'utente, dopo aver preso i vari titoli preferiti dell'utente, vengono inviati tramite API al recommender system che tramite gli algoritmi citati sopra effettua i calcoli e creare un array di titoli raccomandati. I titoli raccomandati prima di essere inviati nuovamente all'interfaccia utente vengono utilizzati per costruire le schede dei film e serie TV tramite l'ausilio di un sito web chiamato TheMovieDb.org. Per poter comunicare con il sito web in questione viene creato un account e poi viene creata una chiave API per poter inviare le richieste API dal server utilizzato.
Una volta generata la chiave, vengono effettuate delle chiamate API al sito themoviedb.org con i titoli dei film raccomandati, i dati in JSON restituiti vengono poi utilizzati per costruire oggetti di film e serie TV. Dopo aver creato i vari oggetti, vengono in-

viati all'interfaccia utente che li legge e li inserisce in componenti per mostrarli graficamente.

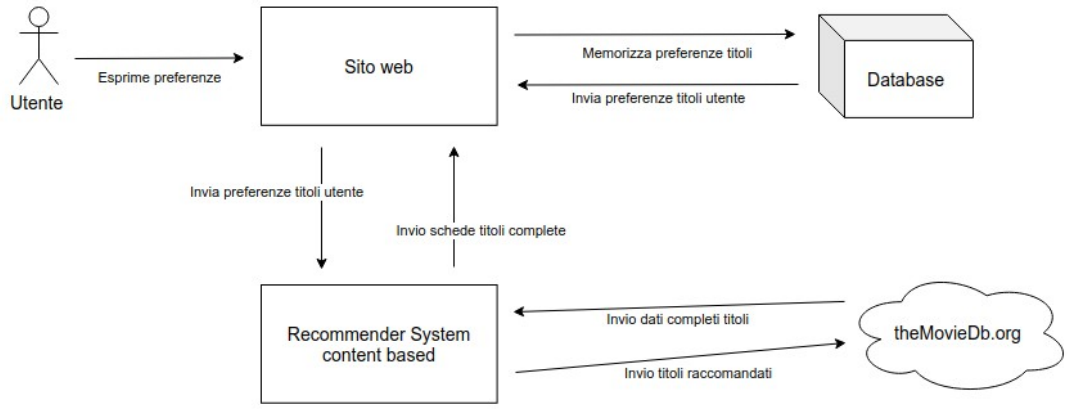

Figura 4: Schema per raccomandare i titoli all'utente

3.2 La rete neurale per il riconoscimento delle emozioni

Come viene scritto nel capitolo 1, il web 5.0 si concentra sul rilevare l'emozione dell'utente che utilizza il sistema e interagire con esso. Per avvicinarci al web 5.0, in questo esperimento viene utilizzata una rete neurale in grado di rilevare l'emozione attraverso il viso dell'utente.

Per seguire l'esperimento sul riconoscimento delle emozioni tramite video frame condotto da L. Do et all. [28], in questa tesi viene presa in considerazione una CNN (rete neurale convoluzionale), utilizzata per il riconoscimento di immagini rappresentando i dati in matrici multidimensionali. La CNN estrae ogni porzione dell'immagini data in input, conosciuta come *campo recettivo*, e per ogni neurone assegna dei pesi in base al ruolo del campo recettivo [24].

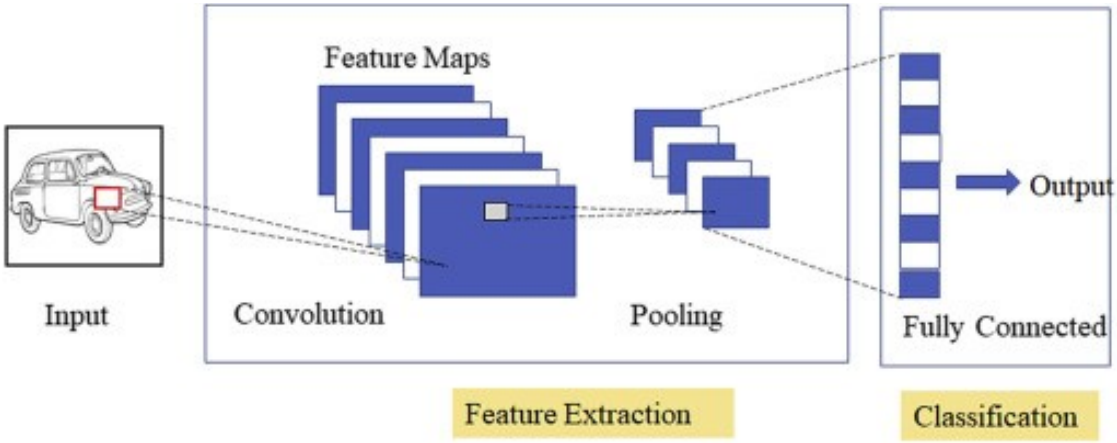

Figura 5: Architettura di una CNN.

Il dataset utilizzato per addestrare e verificare la rete neurale è FER2013 contenente immagini etichettate in base all'emozione della persone presente sull'immagine. Ogni immagine ha come etichetta: happy, sad, angry, afraid, surprise, disgust e neutral, ma per questo esperimento vengono presi in considerazioni solo 3 emozioni: happy, sad e neutral.

La rete neurale utilizzata per questo esperimento si chiama ResNet50 conosciuta anche come Residual Networks con 50 strati di profondità. Lo scopo principale del modello è aumentare la precisioni man mano che il modello diventa più profondo.

Per la costruzione del modello dobbiamo prendere in considerazioni diversi parametri:

- *pooling layers* per ridurre il numero di parametri quando le immagini sono troppo grandi;
- *max pooling* prende l'elemento più grande della mappa delle features e lo sottocampiona;
- *dropout* in un layer nascosto è tra il 0.5 e 0.8;
- *dense* è l'unico strato di rete nel modello, alimenta tutte le uscite dal layer precedente a tutti i neuroni e ogni neurone fornisce un output al layer successivo, viene impostato con 1024 neuroni.

Il numero di epoche utilizzate in questo modello sono 50, quali giocano un ruolo importante nel decidere l'accuratezza del modello, in questo caso ha raggiunto l'86%. Per aumentare l'accuratezza è possibile espandere il set di dati per il training.

Per rilevare l'emozione dell'utente viene scattata la foto tramite webcam dai moduli scritti in javascript che lavoro con il server scritto in express.js. Per comunicare alla rete

neurale che è stata scattata al foto, viene inviato tramite API il nome del file, ovvero il nome della foto scattata dal modulo javascript, la rete neurale successivamente muove la foto dalla cartella del server in express.js alla cartella del server in python per lavorare con essa.

Durante l'elaborazione della foto, la rete neurale effettua la creazione di una nuova immagine modificata dove è raffigurata solo la faccia dell'utente eliminando il resto della foto principale che potrebbe contenere materiale inutile; oltre a ritagliare la faccia dell'utente viene anche cambiato il colore della foto, cosi da ridurre la dimensione in termini di kb, una foto a colori non aiuta il rilevamento dell'emozione perché anche una foto in bianco e nero raffigura in modo corretto l'espressione dell'utente.

Dopo aver lavorato con la foto dell'utente, viene inviata alla vera e propria rete neurale che dopo essere stata addestrata con un dataset adeguato, rileva l'emozione dell'utente. Subito dopo aver rilevata l'emozione, il sistema elimina le due foto dell'utente, quella originale e quella modificata per garantire la privacy dell'utente. L'emozione rilevata viene inviata al server in express.js per continuare il processo di raccomandazione dei titoli.

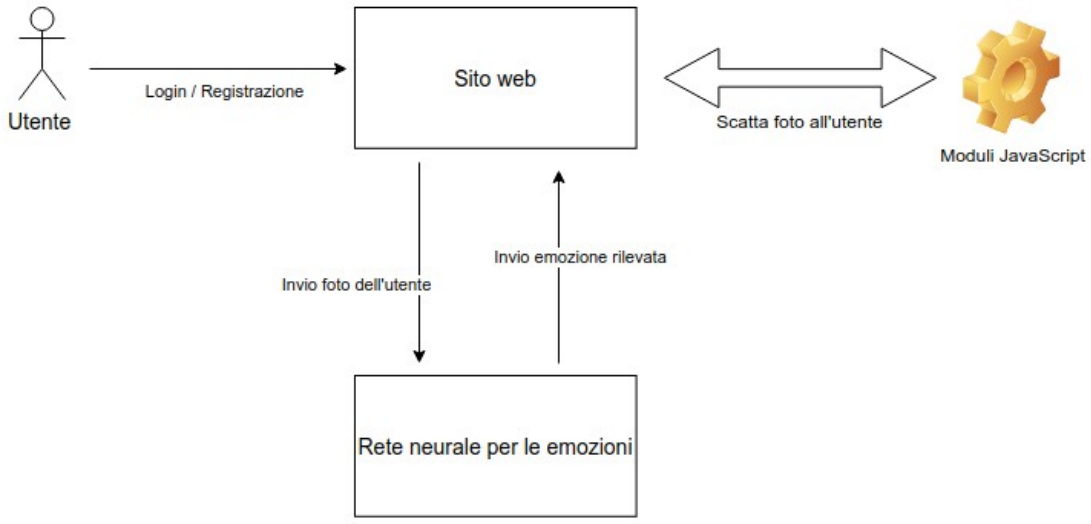

Figura 6: Schema per il rilevamento dell'emozione

3.3 Database e server

Il recommender system e la rete neutrale inizialmente potevano lavorare singolarmente eseguendo il tutto tramite console, bastava richiamare il comando dal terminale

“python <nome_file.py>” e il codice veniva eseguito, ora i due sistemi vengono utilizzati tramite un sito web composto da un server scritto in JavaScript, in particolare con il framework Express.JS e un framework per il lato front-end chiamato React costruito solo per interfacce utente.

Il server principale del sito web permette la comunicazione sia con l’interfaccia utente, in questo caso il front-end, dove vengono mostrati i vari titoli raccomandati all’utente, le nuove uscite, le impostazioni dell’utente in una pagina dedicata e sia con un altro server che permette di lavorare alle raccomandazioni, effettuare query al dataset dei film e serie tv utilizzato, manipolazione del dataset per poter eseguire in modo efficiente le query e al rilevamento dell’emozione dell’utente tramite il suo volto.

La comunicazione tra il front-end e il server che gestisce sia il recommender system e la rete neurale avviene tramite chiamate API effettuate su diversi route. Dopo aver ricevuto l’interazione dell’utente che potrebbe essere esprimere una preferenza su un titolo, eliminare la preferenza, il login, la registrazione; l’azione viene inviata tramite chiama API al server principale il quale a sua volta effettuerà le azioni dovute per quell’interazione, ad esempio se l’utente effettua un login o registrazione, il server eseguirà delle query al database, o inserirà dei dati all’interno del database, se invece l’utente esprime la preferenza su un titolo, prima viene inserita la preferenza nel database e poi viene effettuata una seconda chiamata API al server secondario che andrà a calcolare la raccomandazione per quel titolo preferito dell’utente. Dopo aver eseguito i vari metodi, viene effettuata un’altra richiesta API al sito themoviedb.org dove viene costruita la scheda per ogni film o serie tv raccomandato, questo per arricchire il titolo all’interno dell’interfaccia utente.

Dopo aver costruito le varie schede titoli, la risposta, ovvero le schede costruite, viene inviata al server primario e poi inviata all’interfaccia utente per poi essere mostrata all’utente e attendere nuovamente un’interazione dell’utente.

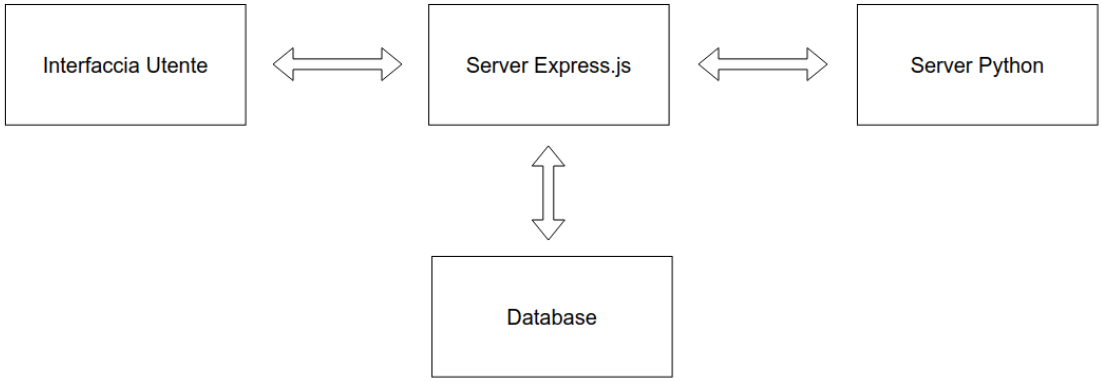

Figura 7: Schema progetto

Per memorizzare le varie informazioni quali i dati dell'utente, le preferenze espresse dall'utente oppure il tempo passato a leggere la trama di un titolo mostrato sul sito, il server comunica con un database adatto a questo progetto composto dalle seguenti tabelle:

- una tabella dedicata alla memorizzazione degli utenti del sito;
- una tabella per memorizzare le preferenze espresse dell'utente verso determinati titoli;
- una tabella per memorizzare il tempo spese dall'utente per leggere la trama di un titoli, questa informazione potrebbe essere utilizzata come feedback implicito per lavori futuri.

La tabella dedicata ai dati dell'utente dove vengono memorizzati i dati anagrafici dell'utente come: nome, cognome, email, password, viene utilizzata quando l'utente effettua il login o la registrazione, sempre tramite l'utilizzo di moduli javascript vengono effettuati dei controlli per individuare se l'utente è registrato o meno, questo a seconda dell'azione dell'utente: se l'utente è già registrato e sta effettuando il login, viene prima eseguita una query per ritrovare l'email dell'utente, se esiste, e successivamente la password salvata precedentemente viene decriptata e viene effettuato un controllo se coincide con quella inserita dall'utente esterno, se coincide allora l'utente potrà essere reindirizzato alla dasboard visualizzando i titoli; se invece l'utente sta eseguendo la registrazione sul sito per la prima volta, una volta inseriti i dati vengono controllati per evitare che l'utente abbia messo un'email già esistente, quindi viene effettuata la prima query al database di ricerca dell'email, se la ricerca va a buon fine e non esistono email simili allora vengono inseriti i dati dell'utente all'interno del database e l'utente è abilitato ad accedere alla dashboard per visualizzare i film e serie tv.

La tabella dedicata alle preferenze dell'utente composta da: email utente, per identificare durante una query le preferenze di un determinato utente, il titolo dove è stata espressa la preferenza dall'utente, la data dell'inserimento della preferenza dell'utente.

La tabella viene presa in considerazione ogni qualvolta l'utente esprime un feedback esplicito su un titolo mostrato sulla dashboard. Dopo aver effettuato il login o la registrazione, verranno mostrate all'utente le varie schede dei film e serie tv, ogni film o serie tv ha in dotazione un pulsante a forma di cuore che permette all'utente di esprimere la preferenze per quel titolo. Una volta che l'utente preme il pulsante, i moduli javascript acquisiscono il titolo del film o serie tv in questione, essendo che l'utente può anche eliminare una preferenza con la stessa metodologia, viene eseguita prima una query sulla tabella dedicata alle preferenze per cercare il titolo interessato:

- se il titolo esiste allora viene eliminato dalla tabella delle preferenze;
- se invece il titolo non esiste, tramite un modulo javascript viene inserito il film o serie tv all'interno della tabella per poi essere ritrovato al momento della costruzione della dashboard e inserito in una sezione dedicata.

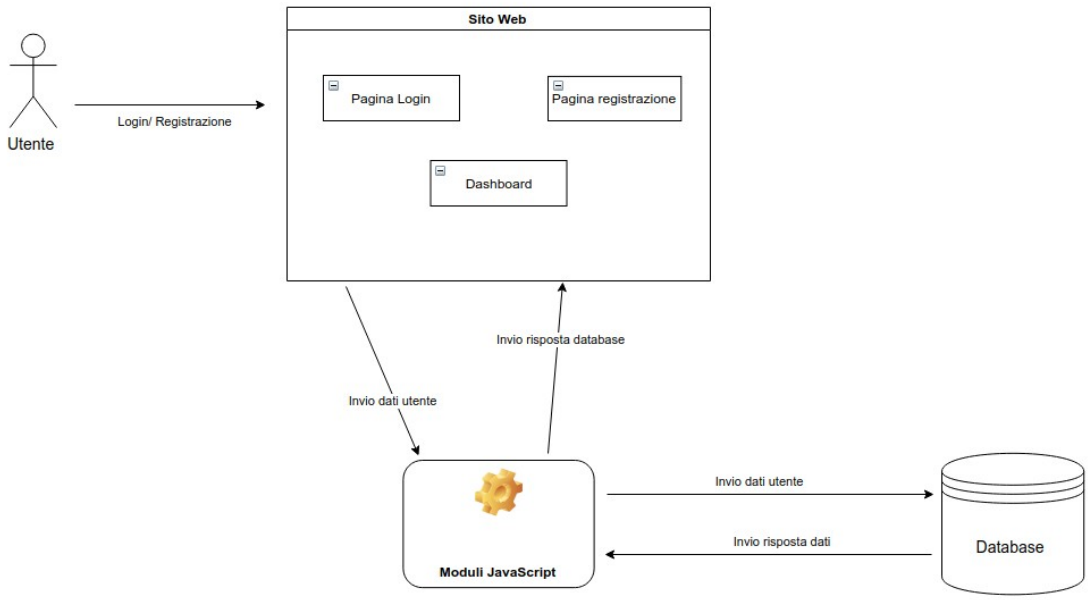

Figura 8: Schema interazione Server - Database

La tabella dedicata al tempo di lettura della trama composta da: email utente, titolo interessato e tempo passato a leggere la trama, viene aggiornata ogni qualvolta l'utente legge la trama di un titolo mostrato sulla dashboard. Per leggere la trama, ogni titolo ha in

dotazione un pulsante ‘trama’ dove una volta cliccato mostrerà uno snippet contenente la trama del film o serie tv in questo.

La lettura della trama potrebbe essere sfruttata come un feedback implicito, un utente potrebbe essere interessato al titolo oppure no. Ogni volta che l’utente premere il pulsante ‘trama’ presente su ogni scheda, viene attivato un timer da 0 e viene salvato il film o serie tv in questione; una volta premuto il tasto della trama, essa scompare e il timer prima avviato si ferma, tramite una chiama API vengono inviati questi dati ai moduli javascript i quali inseriscono questi dati all’interno della tabella dedicata alla trama letta dall’utente.

Possono esistere due casi :

- il primo caso è che la trama del film o serie tv non è mai stata letta, in questo caso viene inserito il dato all’interno della tabella;
- se invece la trama del film o serie tv è stata già letta in precedenza, il dato riguardante quel film o serie tv viene aggiornato con il dato aggiornato.

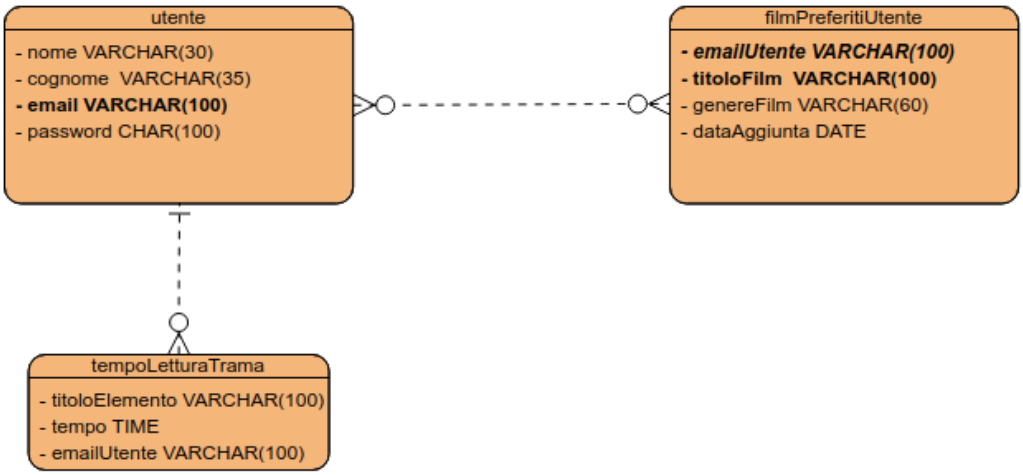

Figura 9: Schema del database del progetto

3.4 Questionario sperimentazione

L’obiettivo da raggiungere in questo progetto è permettere all’utente finale di poter navigare all’interno di un sistema il quale permette, tramite l’emozione dell’utente, di poter effettuare determinate raccomandazioni al fine di migliorare lo stato emotivo dell’utente.

Per poter giungere ad un risultato finale gradito dall'utente, bisogna effettuare una sperimentazione con un numero di utenti, in questo caso 9 persone, che testeranno il sistema creato andando ad eseguire determinati task come ad esempio il login, la registrazione, la lettura della trama di differenti film e serie tv, visualizzare il trailer, cambiare impostazioni utente ecc…

Dopo aver terminato i vari task sottoposti all'utente, verrà richiesto di compilare un questionario SUS, anche in forma anonima per capire lo stato del sistema, se presenta dei problemi, se è possibile apportare delle modifiche e offrire una qualità maggiore all'utente. Il questionario SUS è composto da 10 domande inerenti all'usabilità di un sito web, per mettere in risalto quindi aspetti funzionati e/o critici.

Oltre alle domande previste per il questionario SUS, sono state inserite domande inerenti al sistema per il rilevamento dell'emozione e alle raccomandazioni effettuate da esso. Questo per capire se il sistema, essendo in una fase di test e anche in via di sviluppa, riesce comunque ad avere un corretto funzionamento, se le raccomandazioni sono inerenti allo stato d'animo dell'utente che utilizza il sistema, oppure sono sbagliate, non aiutano l'utente oppure sono irrilevanti per l'utente.

Non sono state effettuate delle domande per il recommender system content based perché l'attenzione deve focalizzarsi sul recommender system basato sulle emozioni, il nostro obiettivo è quello di dare forma al web 5.0 e cercare il più possibile di aiutare l'utente a migliorare il suo umore o continuare a mantenere il proprio stato emotivo.

Capitolo 4 – Implementazione

Implementazione

4.1 Implementazione recommender system

Per raggiungere l'obiettivo di creare un sistema ibrido in grado di consigliare determinati elementi sia in base all'emozione e sia in base alle preferenze dell'utente, dobbiamo costruire un recommender system content based, una rete neurale in grado di rilevare la faccia dell'utente e l'emozione, un sito web online composta da server principale, front-end ovvero la parte grafia del sistema, un database per memorizzare le varie preferenze e dati dell'utente.

Il recommender system utilizzato nel sistema è stato implementato tramite il linguaggio di programmazione Python perché è un linguaggio adatto a questo scopo e a questi tipi di progetti, in grado di facilitare lo sviluppo del sistema tramite l'aiuto di librerie open source, quindi librerie che possono essere modifiche da una community online per migliorare le varie funzionalità che offre, migliorare i bug, ovvero gli errori molto più velocemente e il codice è visibile da chiunque online.

La prima libreria utilizzata è *networkX,* utilizzata per la creazione, manipolazione e lo studio di strutture dinamiche e complesse. È possibile creare differenti tipologie di grado: non orientato, orientato, multigrafo non orientato, multigrafo orientato. Utilizzando questa libreria possiamo confermare la potenza e perché è stato scelto Python come linguaggio di programmazione per costruire questo sistema. Tramite questa libreria scritta in Python è possibile inserire un nodo alla struttura tramite una singola istruzione, senza dover creare altre variabili, altre strutture o istruzioni che appesantiscono il codice.

In questo caso è stata selezionata per permettere la creare del grafo composto da tutti i dettagli di un titolo presente all'interno del dataset utilizzato per questo progetto. Ogni nodo è una proprietà, ad esempio un nodo per il titolo della serie tv o film, un nodo dedicato ad ogni componente del cast, un nodo dedicato al genere e al tipo di titolo, se è un film o serie tv.

```
import networkx as nx

G = nx.Graph(label="MOVIE")

for element in rowi['actors']:
        G.add_node(element,label="PERSON")
        G.add_edge(rowi['title'], element, label="ACTED_IN")
```

Figura 10: Creazione grado e aggiunta di nodi attori e archi

Lavorando con il dataset composto da colonne in base alle proprietà e tuple per inserire i valori, il formato è .csv, la scelta più opportuna da fare è quella di utilizzare la libreria *pandas*, per manipolare velocemente il dataset ed esplorarlo facilmente; alcune operazioni banali sono: la lettura di dataset, l'eliminazione di valori nulli, eliminazione di colonne o query. Anche in questo caso la libreria permette di eseguire operazioni in modo efficace tramite una singola istruzione, come ad esempio la lettura e la sottomissione di una query al dataset memorizzando il risultato, ovvero una nuova tabella composta da colonne e indice per ogni tupla.

```
import pandas as pd

# Lettura del dataset presente nella cartella 'dataset'
df = pd.read_csv('dataset/netflix_titles.csv')

# Query sottoposta al dataset
risultato = pd.Series(data=np.array(weight),index=movies)
```

Figura 11: Lettura del dataset tramite Pandas e query

Per la visualizzazione dei risultati dopo aver manipolato il dataset, è stato utilizzata la libreria *Matplotlib* che facilità la visualizzazione dei risultati per renderli comprensibili all'utente che effettua l'analisi. Le visualizzazioni mostrate dalla libreria possono essere statiche, animate e anche interattive. All'interno del recommender system è stata utilizzata per visualizzare il grafo con i nodi vicini dopo aver dato in input il titolo target.

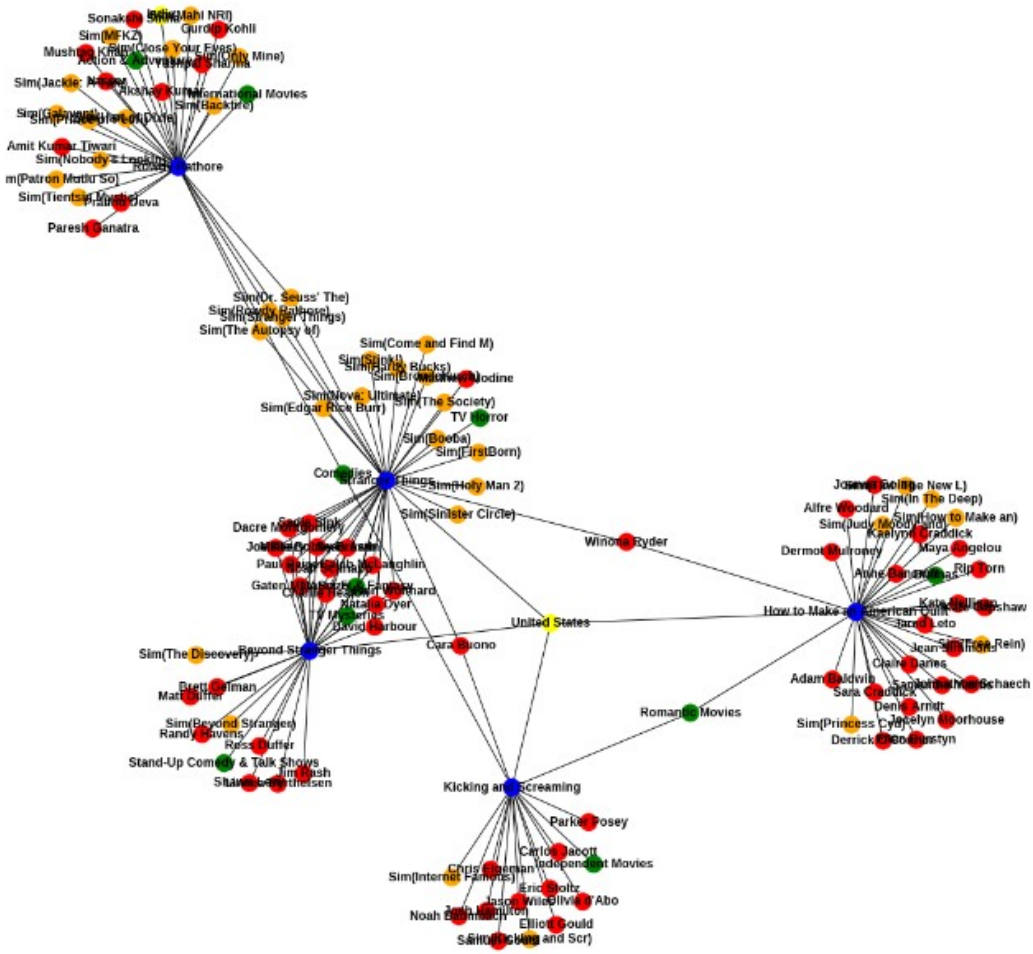

Figura 12: Grafo delle raccomandazioni di un titolo (Stranger Things)

4.2 Implementazione della rete neurale per le emozioni

Per implementare una rete neurale in grado di riconoscere il viso dell'utente e la sua emozione possiamo utilizzare sempre delle librerie programmate in Python per sfruttare tutta l'efficacia del linguaggio ed eseguire operazioni con poche righe di codice.

Per implementare il sistema di riconoscimento facciale abbiamo bisogno di un classificatore in grado di rilevare il viso umano all'interno di una foto, e per fare ciò possiamo utilizzare *Haar Cascade Classifier,* si tratta di un approccio basato sull'apprendimento automatico in cui una funzione a cascata (dal termine cascade) viene addestrata da molte immagini positive e negative, dopo aver addestrato il modello viene poi utilizzato per rilevare oggetti in altre immagini, in questo caso utilizzeremo il modello addestrato per rilevare i volti nell'immagine data in input. Lavorando con i volti delle persone, l'algoritmo ha bisogno di essere addestrato con tanto immagini positive, ovvero con i volti, e immagini senza volti dette immagini negative. Per addestrare il modello vengono utiliz-

zate delle caratteristiche Haar simili al kernel convoluzionale, ogni caratteristica è un singolo valore ottenuto sottraendo la somma dei pixel sotto il rettangolo bianco dalla somma dei pixel sotto il rettangolo nero.

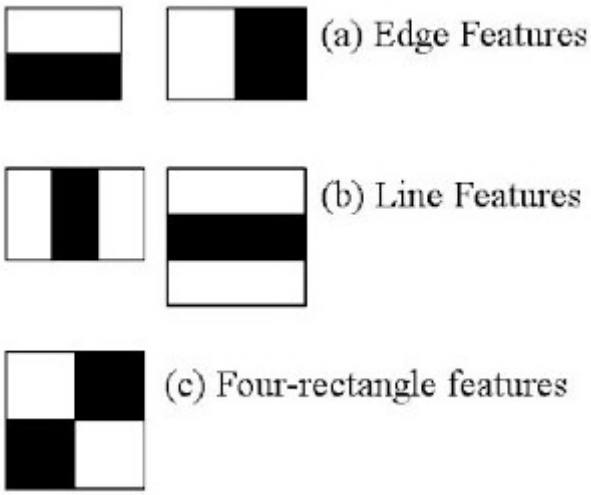

Figura 13: Caratteristiche Haar utilizzate per l'addestramento

Per ridurre il numero di calcoli da effettuare viene introdotta l'immagine integrale, e per ridurre la ricerca di caratteristiche, vengo eliminare le parti non rilevanti all'interno della foto dove il viso dell'utente non è presente, viene ritagliata quindi l'immagine e viene messo in evidenza solo il viso per effettuare successivamente il calcolo della caratteristiche e velocizzare il processo di ricerca, viene introdotto il concetto di cascata di classificatori, ovvero le caratteristiche sono raggruppate in diverse fasi di classificatori e applicate una per una, se una fallisce il primo stadio si scarta, se invece non fallisce continuiamo con i vari stadi. Se vengono superati tutti gli stadi allora la regione è una faccia.

La prima libreria importata è *numpy* che fornisce al programmatore differenti funzioni matematiche, funzioni per algebra lineare e permette la creazione e gestione di array multidimensionali.

La seconda libreria viene utilizza molto spesso nell'ambito della computer vision, ovvero l'interpretazione e la comprensione del mondo in base a delle immagini o video diviso per frame [25]. *OpenCV* è una libreria open source supportata in molti linguaggi di programmazione come Python, C++, Java e molti altri, permette l'elaborazione di immagini e video per l'identificazione di persone, oggetti oppure la firma di una persona. In questo caso viene utilizzato per il rilevamento del volto tramite il classificatore a cascata, viene utilizzato per leggere l'immagine dell'utente, ridimensionare l'immagine dell'utente e creare una nuova immagine che contiene solo il volto dell'utente per velocizzare il calcolo delle caratteristiche.

```
import cv2

facedata = 'dataset/haarcascade_frontalface_alt.xml'
cascade = cv2.CascadeClassifier(facedata)
img = cv2.imread(image)

# Ridimensiona l'immagine iniziale
miniframe = cv2.resize(img, minisize)
# Identificazione del volto dell'utente
faces = cascade.detectMultiScale(miniframe)

# Creazione della nuovo immagine contenente solo il volto
cv2.imwrite(image+'Modificata.jpg', sub_face
```

Figura 14: Cascade Classifier per identificare il volto dell'utente

La rete neurale ha bisogna di essere addestrata con il dataset dato in input, e per fare ciò possiamo ricorrere alla libreria open source *Keras*, utilizzata per l'addestramento automatico e reti neurale, e in questo progetto viene utilizzata insieme alla libreria *TensorFlow*. Inizialmente vengono importati i vari moduli per i: layers, modelli, moduli per l'ottimizzazione e per il pre-processing dell'immagine.

L'ImageDataGenerator di Keras permette l'aumento delle immagini in tempo reale, ovvero che ogni trasformazione può essere applicata alle immagini che vengono date in input al modello mentre esso è in fase di addestramento.

```
train_dir = 'train'
val_dir = 'test'
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(48,48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
        val_dir,
        target_size=(48,48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')
```

Figura 15: ImageDataGenerator di Keras

La rete neurale utilizzata all'interno di questo progetto è conosciuta come ResNet50 ovvero Residual network con 50 layers di profondità. Il nostro modello verrà addestrato con questa rete neurale già pre-addestrata. La rete neurale ResNet50 salta le connessioni e passa il residuo al layers successivo, in questo modo la rete continua ad allenarsi. Ovviamente dobbiamo impostare vari parametri per il corretto funzionamento della rete:

- vengono utilizzate differenti funzioni di attivazione di diversa tipologia, ovvero Lineare e Non lineare. Le funzioni di attivazioni sono: *Rectified Linear Unit (ReLU) e Softmax,* vengono utilizzate per ottenere i valori dati come risultato nell'intervallo da 0 a 1 o da -1 a 1 in base alla funzione;

- *pooling layers* è un parametro utilizzato per andare a ridurre il numero di parametri quando le immagini in input sono troppo grandi;

- *downsampling* riduce la dimensione di ogni mappa ma conserva le caratteristiche più importanti;

- *max pooling* ricampiona l'elemento più grande dalla mappa delle caratteristiche;

- dropout è una tecnica di regolarizzazione per le reti neurale e modelli di apprendimento profondo. Per un layer nascosto è possibile impostare valori che vanno da 0.5 a 0.8;

- dense unico layer effettivo nel modello che alimenta tutte le uscite dal layer precedente.

```
emotion_model = Sequential()
emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))
```

Figura 16: Parametri per il modello

Successivamente viene addestrato il modello utilizzando l'ottimizzatore di Adam prima importato tramite la libreria Keras, il parametro *categorical_crossentropy* calcola la perdita di crossentropia tra etichette e previsioni, la funzione di perdita viene usata quando ci sono due o più classi di etichette. Per addestrare il modello sono state impostate 50 epoche .

```
emotion_model.compile(loss='categorical_crossentropy',
                      optimizer=Adam(lr=0.0001,
                                     decay=1e-6),
                      metrics=['accuracy'])

emotion_model_info = emotion_model.fit_generator(
        train_generator,
        steps_per_epoch=28709 // 64,
        epochs=50,
        validation_data=validation_generator,
        validation_steps=7178 // 64)
```

Figura 17: Impostazioni per l'addestramento della rete

4.3 Implementazione del server in Python

Dopo aver costruito il recommender system e la rete neurale addestrata per poter rilevare l'emozione dell'utente, abbiamo bisogno di creare un server in Python per permettere la comunicazione con i moduli che compongono il recommender system e la rete neurale, ricevere le varie informazioni, come ad esempio i film da suggerire, vari query sui film e l'emozione dell'utente rilevata tramite rete neurale.

Per fare ciò abbiamo bisogno di creare il primo server in Python composto da endpoint API per inviare a richiedenti i quali possono essere client o altri server, informazioni riguardanti i titoli tramite il recommender system o emozioni dalla rete neurale. Nel nostro progetto, il server scritto in Python verrà utilizzato da un server scritto in javascript per ricevere le informazioni. Per creare il server in Python abbiamo bisogno di *flask*, un micro-framework web che permette di costruire la parte back-end delle applicazioni web.

In questo caso, tramite *Flask* andremo a costruire un server in python che resterà in ascolto su una determinata porta per ricevere le varie chiamate da un determinato client qualsiasi o server.

```
import flask
from flask import request, jsonify

app = flask.Flask(__name__)
app.config["DEBUG"] = True

app.run()
```

Figura 18: Creazione server con Flask

Dopo aver creato il server che resterà in ascolto sulla porta 5000, andiamo a costruire dei route specifici per le varie risorse richieste dal client: una route per i film o serie tv recenti, una route con parametro che permette di memorizzare il valore del parametro, in questo caso il titolo preferito dell'utente e tramite il recommender system (prima sviluppato) richiamato tramite l'import di un modulo, possiamo restituire i vari titoli raccomandati, ed infine una route che prende in input l'emozione e restituisce titoli adatti per l'emozione.

```
@app.route('/film/nuovi', methods=['GET'])
def api_nuovi():
    return jsonify({'film_nuovi': qCSV.film_nuovi()})

@app.route('/film', methods=['GET'])
def api_id():
    if 'titolo' not in request.args:
        return "Errore: nessun titolo trovato nell'URL."
    else:
        titolo = request.args['titolo']

    result = raccomandazione(titolo)
    result = qCSV.costruisci_film(result.index.values.tolist())
    return jsonify({'film_raccomandati': result[:10]})

@app.route('/filmemo', methods=['GET'])
def emozione_film():
    if 'emozione' not in request.args:
        return "Errore: nessun titolo trovato nell'URL."
    else:
        emozione = request.args['emozione']
    return jsonify({'film_emozione': qCSV.film_base_emozione(emozione)})
```

Figura 19: Implementazione dei route per le richieste API

Per i titoli recenti e le varie query da applicare al dataset è stato creato un nuovo file python contente dei moduli che restituiscono i vari titoli in base ai criteri da rispettare. I vari moduli possono essere richiamati in file python esterni, in questo, come si può notare nella figura precedente, è stata utilizzata una libreria esterna chiamata *qCSV* che permette di restituire, dato il modulo chiamato, dei risultati, come ad esempio il modulo per ricevere i film o serie tv nuove prelevate dal dataset. Prima di restituire i vari titoli al server python, vengono richiamati dei moduli interni alla libreria per modellare i film cercati e restituire solo le informazioni necessarie come il titolo, il genere e il tipo.

```
def costruisci_film(elencoFilm):
    df_noedit = pd.read_csv(netflix_titles.csv)
    df_noedit = df_noedit.drop('show_id', axis=1)
    df_noedit = df_noedit.drop('director', axis=1)
    df_noedit = df_noedit.drop('country', axis=1)
    df_noedit = df_noedit.drop('cast', axis=1)
    df_noedit = df_noedit.drop('date_added', axis=1)
    df_noedit = df_noedit.drop('release_year', axis=1)
    df_noedit = df_noedit.drop('rating', axis=1)
    df_noedit = df_noedit.drop('duration', axis=1)
    df_noedit = df_noedit.drop('description', axis=1)

    film_cercati = list()
    for i in range(10):
        for k in range(len(df_noedit)):
            if(df_noedit.values[k][1] == elencoFilm[i]):
                film_cercati.append({df_noedit.values[k][1]:
                                     {df_noedit.values[k][0] :
                                      categoria_elemento(df_noedit.values[k][2])}})
    return film_cercati

def categoria_elemento(stringa_categorie):
    array_parole = stringa_categorie.split(",")
    return array_parole[0]

def film_nuovi():
    df = netflix['release_year'].sort_values(ascending=False)[:10].to_frame()
    film_nuovi = list()

    for i in df.index:
        film_nuovi.append(netflix['title'].iloc[i])

    film_nuovi = costruisci_film(film_nuovi)
    return film_nuovi
```

Figura 20: Moduli per interrogare il dataset

4.4 Implementazione del server in JavaScript

Dopo aver implementato il server in Python abbiamo bisogno di richiedere le informazioni tramite un server scritto in JavaScript tramite la librerie Express.js. L'utilità di utilizzare un server in javascript è la facilità di gestione della sessione dell'utente tramite i cookie, poi è possibile importare le varie librerie tramite uno gestore dei pacchetti chiamato npm.

Per prima cosa abbiamo bisogno di implementare il server che resterà in ascolto su una determinata porta, in questo caso 3001. Per implementare il server abbiamo bisogno di differenti pacchetti quali: *express, cookieParser e session* per la gestione della sessione dell'utente dopo aver effettuato il login o la registrazione.

```
import express, { json } from "express";
import cookieParser from "cookie-parser";
import session from "express-session";
const app = express();
app.use(cookieParser());
app.use(bodyParser.urlencoded({
    extended: true
}));

app.use(session({
    key: "userCookie",
    secret: process.env.SECRET_COOKIE,
    resave: true,
    saveUninitialized: false,
    cookie: {
        expires: 1000 * 60 * 15,
    }
}));

app.listen(3001, () => {
    console.log('Server avviato.');
});
```

Figura 21: Implementazione server in JavaScript

Cosi come il server in python, anche su questo server abbiamo bisogno dei vari route che possiamo utilizzare per effettuare le chiamate API dal client, ricevere i vari film, serie tv e mostrare i risultati tramite interfaccia utente. Ma per fare ciò, abbiamo bisogno di far comunicare il server javascript con il server scritto in python. Il server in javascript riceverà i dati dal server python e andrà ad effettuare la costruzione delle scheda complete dei titoli mandati in output dal server python. Ogni scheda che verrà costruita sarà composta da una trama, trailer, genere e altre informazioni rilevanti, tutto ciò per ogni film ricevuto dal server. Per eseguire questa costruzione, abbiamo bisogno di effettuare richieste API verso un sito online chiamato *themoviedb.org*, sito web contente tutte le informazioni su film e serie tv.

```
app.get('/film/nuovi', (req,res) =>{
    //  richista server python
    film_nuovi()
        .then((response) => {
            //  richiesta API al sito online di film per la costruzione di oggetti
            dati_film(response.film_nuovi)
                .then(dati=>{
                    //  Invio al client
                    res.send(dati);
                });
        });
    //  fine film_nuovi()
});

// FILE api.js PER I METODI SPECIFICI
export const film_nuovi = async () => {
    let response;

    //  richiesta api
    try {
        response = await axios.get('http://127.0.0.1:5000/film/nuovi');
        //  Se la richiesta è andata a buon fine
        if(response.status === 200)
            return response.data;

        return response.status;
    } catch (e) {
        // errore
        throw new Error(e.message)
    }
}
```

Figura 22: Chiamata API al server Python con JavaScript

Per effettuare le chiamate a themoviedb.org abbiamo bisogno di una chiave che ci permetterà di effettuare le chiamate; senza questa chiave, che possiamo richiedere sul sito dopo aver effettuare la registrazione di un account, non avremo l'autorizzazione necessaria ad effettuare chiamate API e reperire le varie informazioni.

Quando riceveremo i dati dal server python, possiamo inviare una richiesta API a *themoviedb.org* con differenti parametri in input: la chiave del nostro account per essere autorizzati ad effettuare la chiamata, il tipo di titoli che stiamo richiedendo, ovvero se è un film o una serie tv, e il titolo che stiamo cercando. Il risultato sarà un array di oggetti in JSON con le varie informazioni: titoli, lingua, trama, key per il trailer, key per l'immagine ecc… Tutte queste informazioni vanno organizzate in oggetti in JSON e inviati al client tramite il server scritto in javascript.

```
export const dati_film = async (lista_film) =>{
    let response, responseTrailer;
    const films = [];

    let i = 0;
    for (const film of lista_film){
        let valore = Object.values(film);

        if(Object.keys(valore[0]) == 'TV Show'){
          response = await axios.get('https://api.themoviedb.org/3/search/tv?api_key='
                                     +process.env.API_KEY_FILM+'&query='
                                     + Object.keys(film));

          if(response.data.total_results != 0 && response.data.results[0].poster_path != null){
              responseTrailer = await axios.get('https://api.themoviedb.org/3/tv/'
                                                +response.data.results[0].id+'/videos?api_key='
                                                +process.env.API_KEY_FILM+'&language=en-US');

              if(responseTrailer.data.results.length != 0){
                  let filmObj = {
                      locandina: "https://image.tmdb.org/t/p/w500"+response.data.results[0].poster_path,
                      titolo : response.data.results[0].name,
                      trama : response.data.results[0].overview,
                      genere : Object.values(valore[0]),
                      tipo: Object.keys(valore[0]),
                      trailer: 'https://www.youtube.com/embed/'+responseTrailer.data.results[0].key
                  }
                  i++;
                  films.push(filmObj);
              }
          }
```

Figura 23: Richiesta dati al sito themoviedb.org

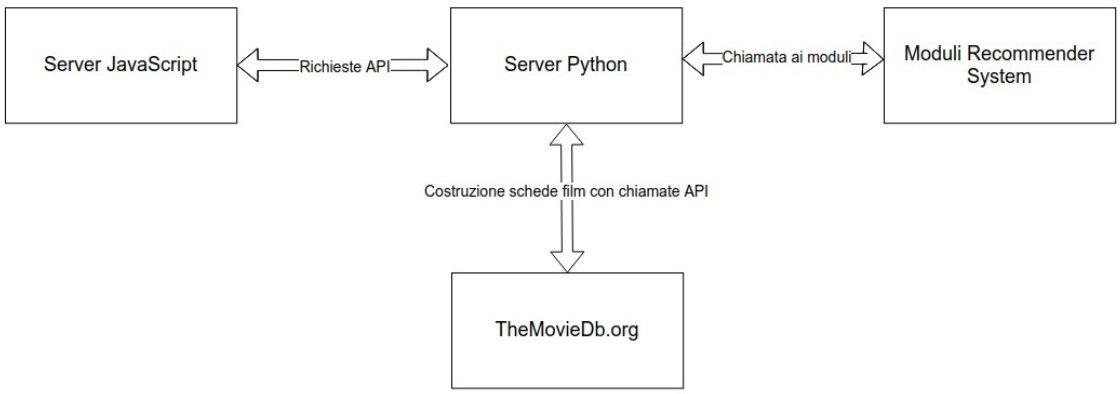

Figura 24: Comunicazione con chiamate API tra server

4.5 Implementazione del client

Per avere un'interfaccia utente dinamica e reattiva è possibile utilizzare il framework sviluppato da Facebook chiamato *React*.

Tramite React è possibile creare dei componenti da utilizzare per creare interfacce utente senza dover riscrivere il codice, ad esempio per questo progetto è stato creato il componente *film* che permette di mostrare tutti i dettagli del film come: titolo, locandina, genere, pulsante per la trama e tipo di titolo.

```
<div>
  <Row>
  {films.map((film) => (
      <Col className="filmItem" xs lg="2" key={film.titolo}>
          <Card style={{ width: '18rem' }}>
              <Card.Img variant="top" src={film.locandina} />
              <Card.Body>
                  <Card.Title>{film.titolo} </Card.Title>
                  {['top'].map((placement) => (
                      <OverlayTrigger
                      trigger="click"
                      key={placement}
                      placement={placement}
                      overlay={
                          <Popover id={`popover-positioned-${placement}`}>
                          <Popover.Header as="h3">Trama</Popover.Header>
                          <Popover.Body>
                              {film.trama}
                          </Popover.Body>
                          </Popover>
                      }
                      >
                      <Button variant="outline-warning"
                          onClick={() => prendiTempo(film.titolo)}>Info</Button>
                      </OverlayTrigger>
                  ))}
                  <Trailer film={film} />
                  <Button variant="outline-danger" className='btn-like' value={film.titolo}
                        onClick={() => saveFilm(film.titolo, film.genere[0], film.tipo[0])}>
                      <i className="far fa-heart"></i>
                  </Button>{' '}
                  <br></br><br></br>
                  <Badge pill bg="warning" text="dark">
                      {film.genere}
                  </Badge>{' '}
                  <Badge bg="secondary">{film.tipo}</Badge>{' '}
              </Card.Body>
          </Card>
      </Col>
  ))}
  </Row>
</div>
```

Figura 25: Componente React per mostrare tutti i film e serie tv

Per poter mostrare tutti i dati riguardanti i titoli, dobbiamo effettuare tramite la libreria *axios* delle chiamate API al server scritto in javascript. Il server manderà in output un array, indipendentemente dalla tipologia di titoli richiesti, contente oggetti in JSON con i vari dati dei titoli (titolo, tipo, locandina, genere ecc…).

```
const fetchFilm = async () => {
  //  richiesta per mostrare i film piaciuti
  const filmPreferiti = await axios('http://localhost:3001/film/preferiti');
  setFilmPreferiti(filmPreferiti.data);
  filmPreferitiLet = filmPreferiti.data;


  //  Richiesta per ricevere i nuovi film
  const rispostaFilmNuovi = await axios('http://localhost:3001/film/nuovi');
  setFilms(rispostaFilmNuovi.data);

  setUltimoCaricamento(true);
}
```

Figura 26: Richieste API per i film e serie tv

Dopo aver ricevuto questi dati sotto-forma di array, possiamo richiamare la funzione *.map()* sull'array ricevuto dal server e utilizzare il componente film per mostrare i dati dei titoli nell'interfaccia utente. La pagine web dove vengono mostrati tutti i titoli all'utente cambia colore in base all'umore dell'utente. Se l'umore è triste, il colore tenderà ad avere una sfumatura verde e nera, se invece è felice il colore tenderà ad una sfumatura celeste e nera, se l'emozione è neutra, il colore non cambierà e resterà quello originale.

Le varie chiamate API vengono utilizzate anche per mantenere le pagine web al sicuro da accessi non autorizzati, ogni qualvolta un utente prova ad accedere ad una pagina, viene effettuato un controllo con i cookie per capire se l'utente è registrato o ha effettuato il login; se il controllo va a buon fine, allora potrà continuare a navigare all'interno del sito, se il controllo non va a buon fine, verrà renderizzato alla home dove potrà scegliere se effettuare i login o registrarsi.

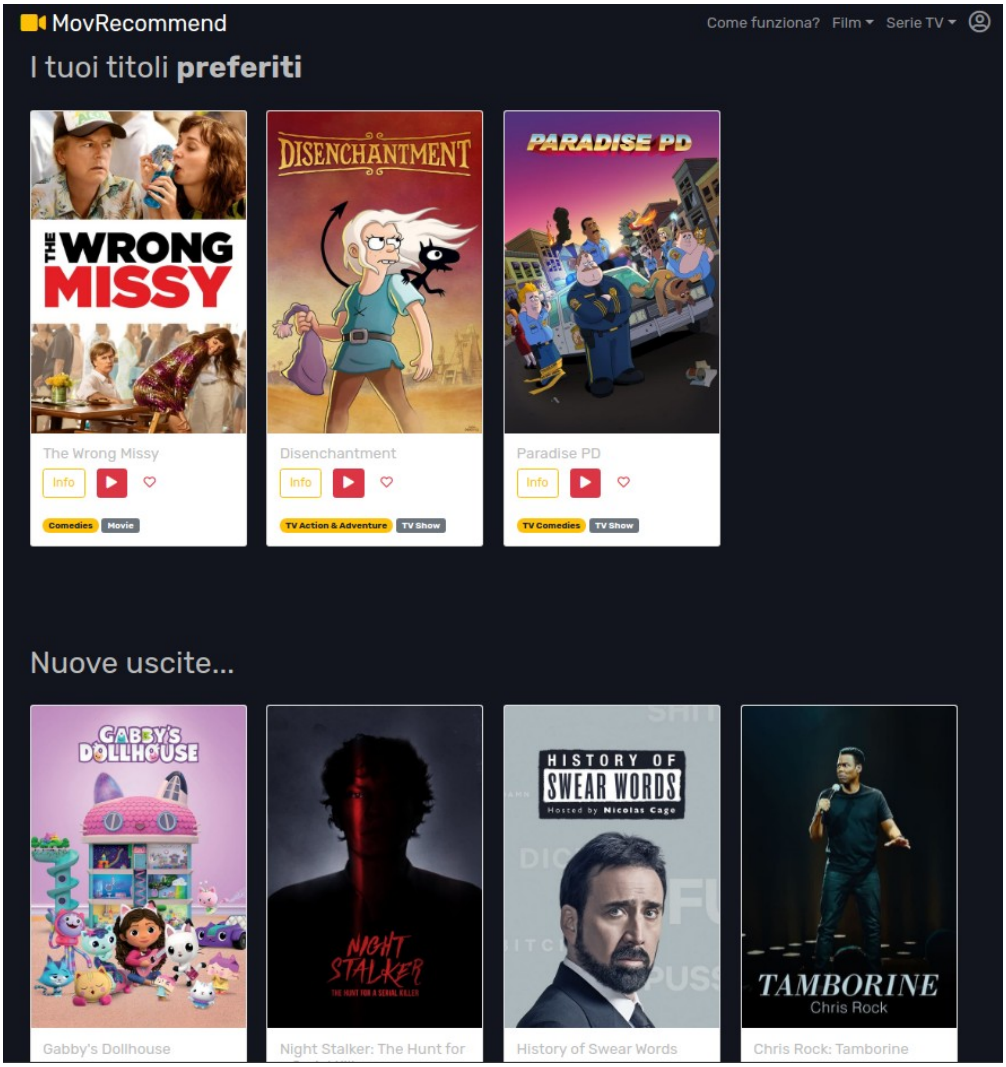

Figura 27: Schermata Home dopo il login o registrazione

Ovviamente è possibile utilizzare i componenti anche per altri elementi dell'interfaccia utente come: la navbar, il footer, logo, impostazioni utente ecc…

Infatti all'inizio è stata costruita una pagina iniziale che comprende un logo iniziale, una navbar che inizialmente è vuote perché mostrerà dei pulsanti solo quando l'utente effettuerà il login o la registrazione, e sono stati inseriti due pulsanti per permettere all'utente di effettuare il login o la registrazione.

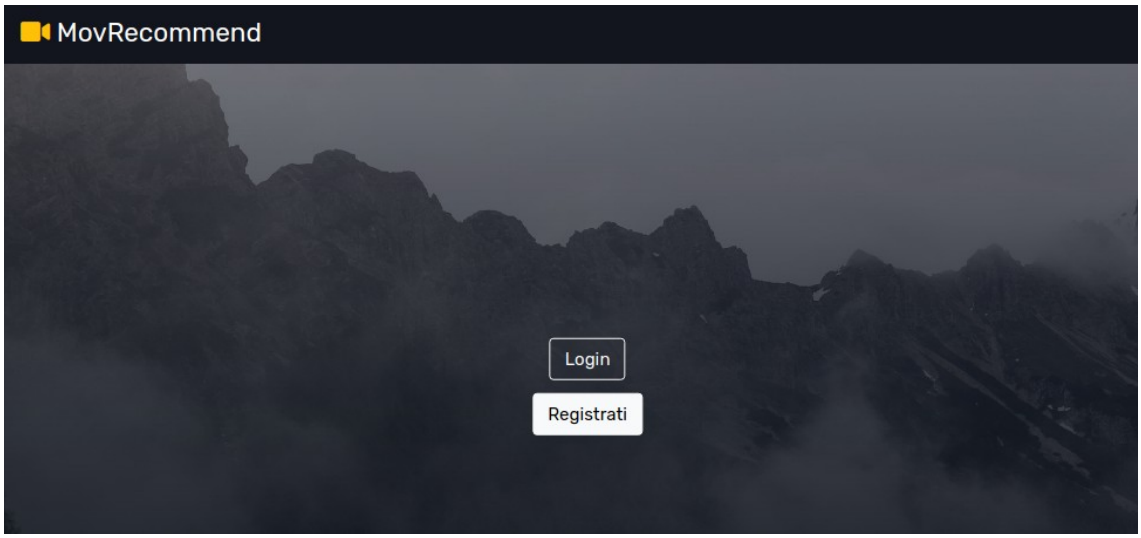

Figura 28: HomePage del sito web

Ogni singola pagina del sito è stata costruita interamente utilizzando i componenti. Le due pagine di login e registrazione comprendono varie textbox dove è possibile inserire i dati dell'utente per effettuare la creazione di un nuovo account o accedere con un account già esistente.

La pagina di Login comprende un textbox dove è possibile inserire l'email dell'utente e un textbox dove è possibile inserire la password. Quando l'utente inserisce email e password, viene effettuata un controllo da parte di moduli javascript che permettono di controllare la password inserita all'interno del textbox per evitare di essere ben visibile all'interno del database, poi vengono inviati al database i dati il quale effettua un controllo per vedere se l'utente è già registrato. Se esiste l'email, viene effettuato un controllo con la password inserita e quella già presente all'interno del database. Se le password corrispondono, allora l'utente è abilitato a navigare sul sito, sennò verrà inviato un messaggio di errore.

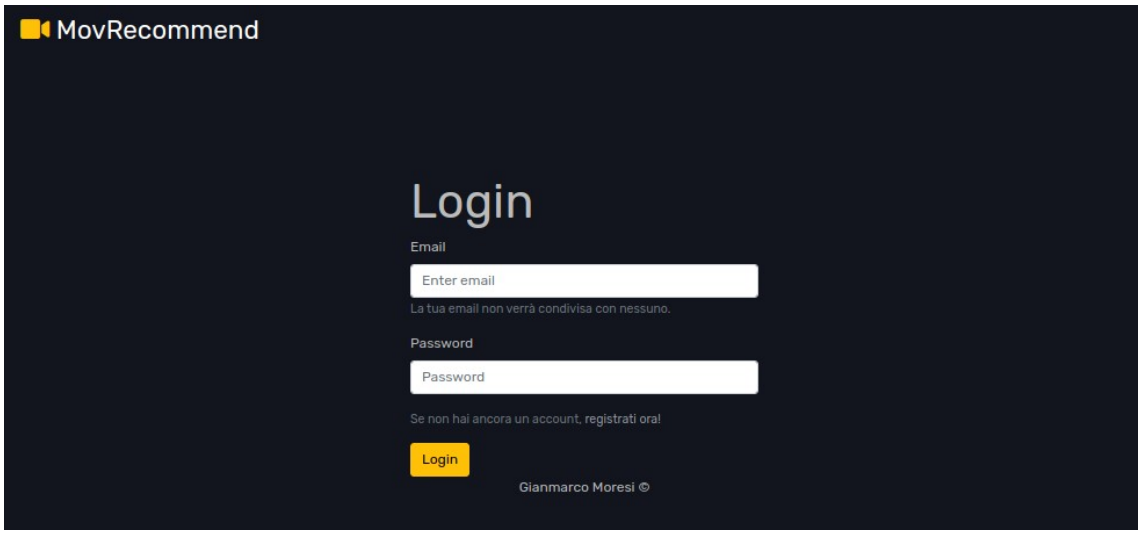

Figura 29: Pagina di login del sito

Questo progetto lavora con il volto dell'utente, il che significa che il sistema deve essere autorizzato a scattare tramite la webcam una foto all'utente e processarla tramite la rete neurale per identificare l'emozione. Per avere il permesso dell'utente, quando l'utente effettua il login o la registrazione, il sistema invia un messaggio all'utente avvisandolo che verrà scattata una foto per capire l'emozione. Se l'utente accetta proseguirà con la navigazione, sennò resterà nella pagina di login o registrazione.

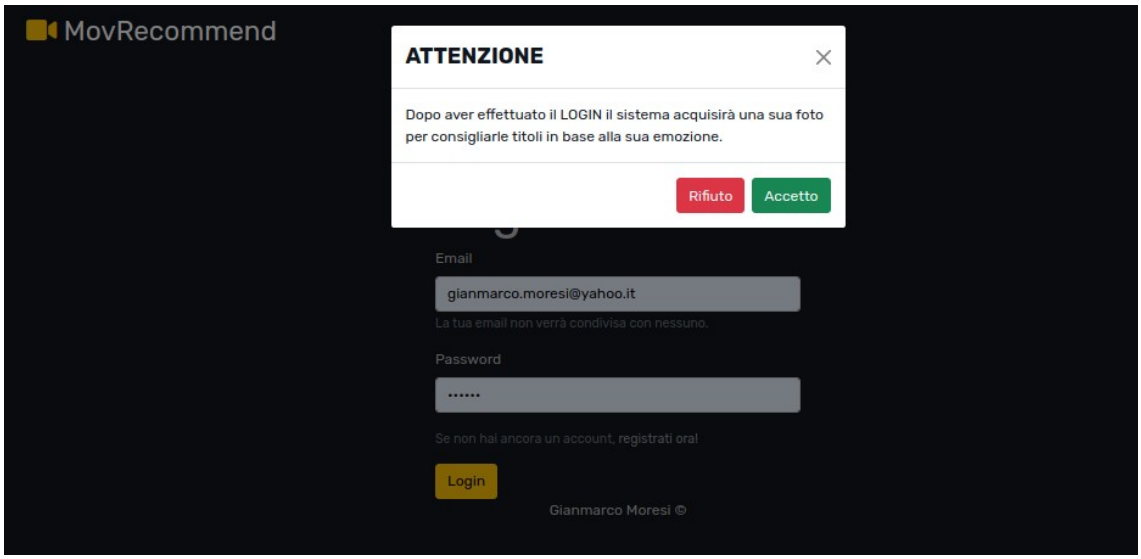

Figura 30: Messaggio di richiesta autorizzazione dal sistema

Per la pagina di registrazione, vengono chiesti diversi dati anagrafici all'utente, compresa la conferma della password. Una volta inseriti tutti i dati, viene effettuato un controllo preliminare sull'email inserita dall'utente, perché può esiste un account con la stessa email inserita, e quindi il server restituirà un errore perché l'email è già presente nel database. Quindi l'utente dovrà inserire una nuova email, ma i vecchi dati anagrafici come nome, cognome ecc… verranno mantenuti nei rispetti textbox, cosi da evitare di inserirli nuovamente e annoiare l'utente.

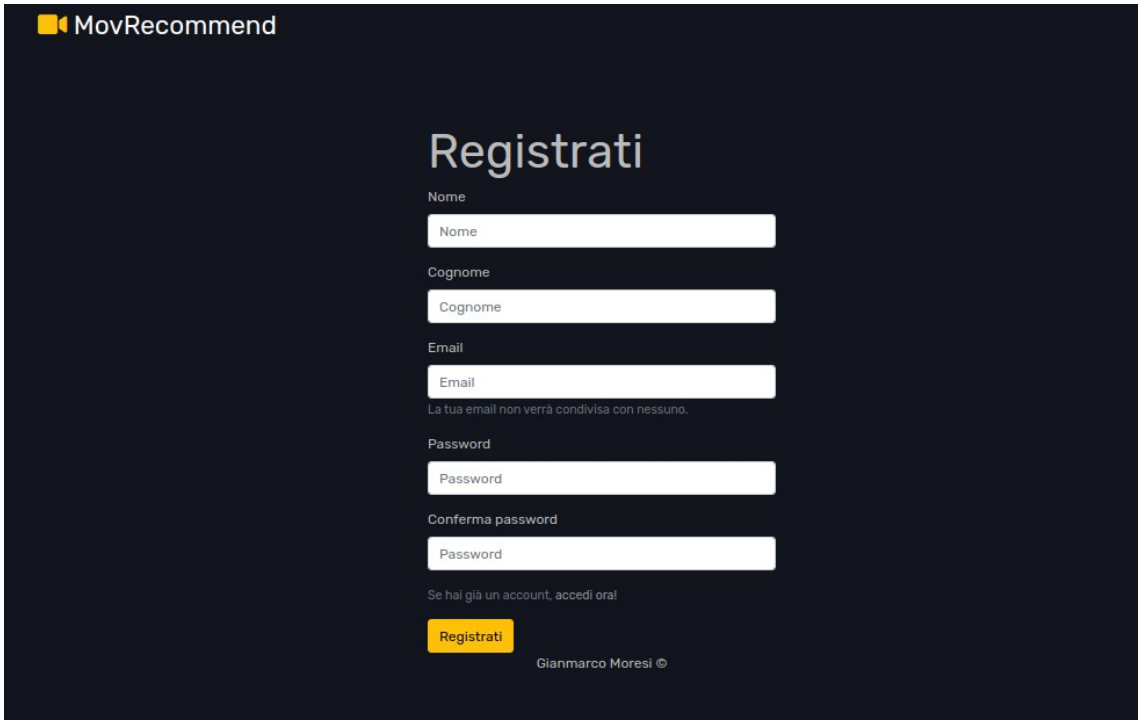

Figura 31: Pagina di registrazione del sito

Dopo aver eseguito il login o la registrazione, l'utente visualizzerà la dashboard dove sono presenti tutti i film e serie tv. È presente una pagina dedicata alle impostazioni dell'utente, ma ha il solo scopo di mostrare i dati dell'utente e poter eliminare diverse preferenze dell'utente sui vari titoli mostrati. Tramite l'icon del cestino presente affianco ad ogni titolo mostrato nella pagina, è possibile inviare una query al database per eliminare il titolo dalla tabella dellle preferenze.

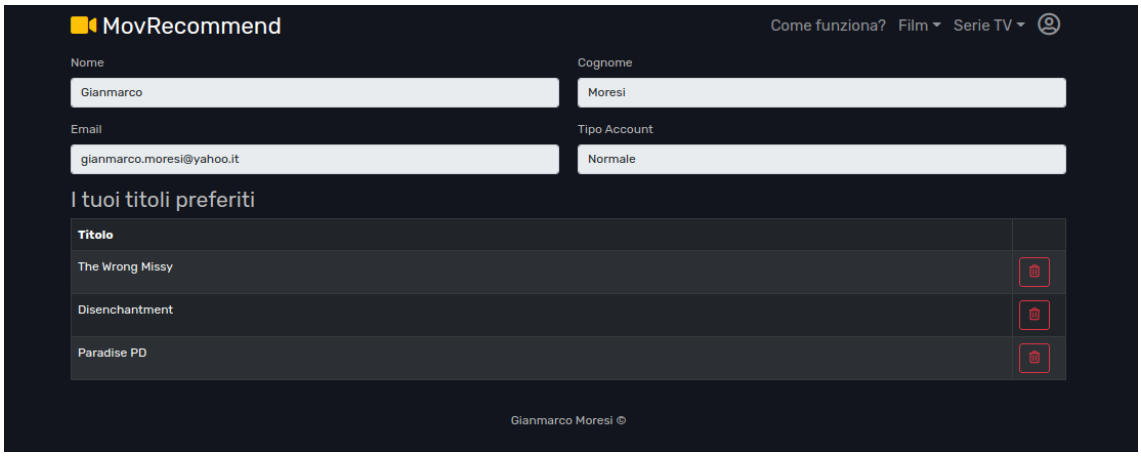

Figura 32: Pagina dei settings dell'utente con le preferenze

All'interno dei vari componenti vengono effettuate delle chiamate API al server scritto in JavaScript per richiedere i vari titoli da mostrare all'interno dell'interfaccia utente. La libreria utilizzata per le varie chiamate si chiama *axios,* dopo aver effettuato la chiamata si attende la risposta da parte del server ed è possibile effettuare una nuova chiamata la server.

Questo sistema è stato utilizzato nella pagina dashboard del sito per poter mostrare all'utente i titoli recenti, le raccomandazioni e i titoli in base all'emozione.

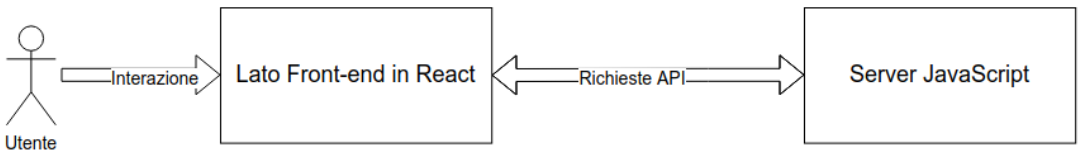

Figura 33: Interazione Client - Server

Capitolo 5 – Sperimentazione

Sperimentazione

Per misurare l'efficacia delle raccomandazioni effettuate in base all'emozione dell'utente rilevata mediante l'utilizzo di una rete neurale e un dataset di immagini di utente, ho eseguito dei test con utenti reali accompagnati da un questionario finale. Lo scopo del progetto è permettere al sistema di soddisfare un bisogno dell'utente mediante lo studio e analisi dell'emozione.

Il test è stato svolto da 9 utenti reali, di sesso differente: in questo caso il 55,6% maschile ovvero 5/9 utenti, il restante 44,4% femminile ovvero 4/9 e un utente ha deciso di non esprimere il proprio sesso. Tutti gli utenti coinvolti durante i test hanno un età differente tra di loro per diversificare le abilità e conoscenze.

Differenti utenti navigano ogni giorno in internet, e non tutti hanno la stessa conoscenza, la stessa velocità nel compiere le azioni o intuizioni, ecco perché è stato deciso di scegliere differenti tipi di partecipanti.

Le varie domande sono state studiate da un team di esperti e caricate online cosi da essere accessibili a tutti per misurare l'usabilità di un sistema. Il test di usabilità può essere effettuato dagli utenti in qualsiasi fase di sviluppo del servizio, per studiare il comportamento dell'utente oppure migliorare il servizio risolvendo anomalie o migliorando determinate parti del sistema. Le varie domande sono state sottoposte agli utenti dopo aver completato differenti tasks sul sito web. Per capire e studiare bene l'efficacia del sistema di rilevamento automatico dell'emozione sono state scritte due nuove domande , una dove è possibile rispondere con una scala numerica che va da 1 a 5 e una tramite una risposta multipla. La prima domanda nuova dove è possibile rispondere tramite scala multivalore è: "Le raccomandazioni emotive erano conformi ai tuoi gusti" e "Il sistema ha rilevato correttamente la sua emozione".

La prima domanda ha ricevuto molti esiti positivi:

- 4 persone su 9 hanno selezionato 5, ovvero il massimo della scala, questo può significare che i titoli proposti erano adatti per migliorare l'emozione o che rispecchiano l'emozione;
- 2 persone su 9 hanno selezionato 4, sono soddisfatti dei vari titoli proposti;
- 3 persone su 9 hanno selezionato 3, sono mediamente soddisfatti dei titoli raccomandati. Questo numero potrebbe essere causato dal dataset non aggiornato dei titoli perché le ultime uscite si fermano all'anno 2020, e Netflix carica ogni gior-

no titoli differenti, quindi anche un mese di ritardo dall'aggiornamento del dataset potrebbe essere rilevante, oppure perché si aspettano titoli differenti.

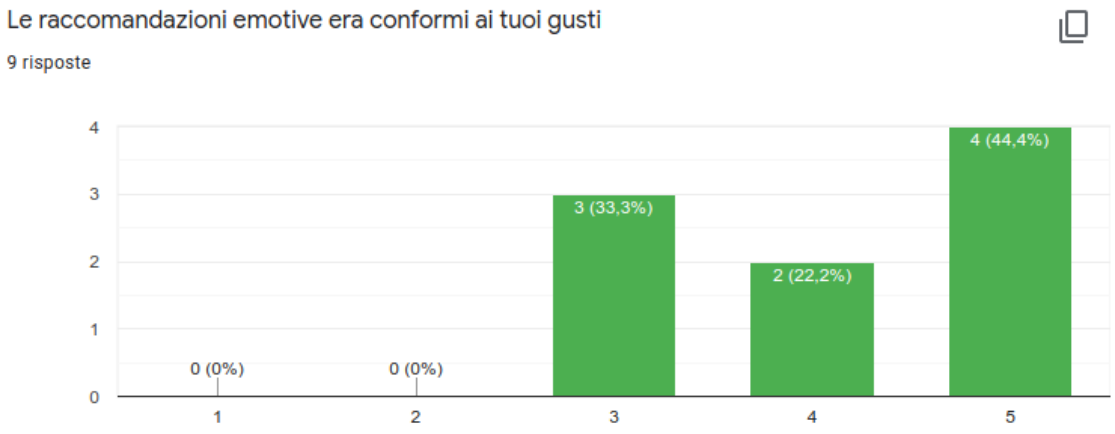

Figura 34: Domanda sulle corrette raccomandazioni in base all'emozione

La seconda domanda "Il sistema ha rilevato correttamente la sua emozione" dove era possibile rispondere con "Si" e "No" permette di valutare il sistema di rilevamento dell'emozione e non di raccomandazione.

- Il 66,7 % ovvero 6 utenti su 9 ha votato con "Si", ciò significa che il sistema ha rilevato correttamente l'emozione e non hanno dovuto ricarica la pagina per poter essere rilevati nuovamente;
- Il restante 33,3% ovvero 3 utenti su 9 ha votato "No". Questo potrebbe significar che il sistema non ha riconosciuto il volto dell'utente e quindi non ha rilevato l'emozione dell'utente, oppure l'emozione rilevata dal sistema non era giusta rispetto all'emozione reale dell'utente.

Figura 35: Domanda sulla correttezza del rilevamento dell'emozione

I due problemi sopra citati potrebbero essere dovuti al dataset utilizzato per addestrare la rete neurale perché le foto presenti nel dataset sono di qualità differente rispetto a foto scattate con la webcam del computer. Le foto scattate con il computer sono sgranate e la rete neurale non riconosce bene il volto; Oppure il luogo dove l'utente utilizza il sistema non è abbastanza illuminato per poter rendere ben visibile il volto dell'utente.

Capitolo 6 – Sviluppi futuri e conclusioni

Sviluppi futuri e conclusioni

Dati i risultati nel capitolo precedente e analizzata l'accuratezza della rete neurale al momento del training al capitolo 3, possiamo definire soddisfacente la prima versione del sistema che permette di rilevare l'emozione dell'utente tramite rete neurale, e con l'emozione analizzata permettere le varie raccomandazioni su film e serie tv. Non solo vengono effettuate le varie raccomandazioni sulla base delle emozioni ma vengo effettuate anche raccomandazioni sulla base delle preferenze espresse dall'utente.

Un ostacolo molto importante che potrebbe essere risolto in una prossima versione è il dataset utilizzato per il training, si potrebbero sostituire le foto attuali con altre foto scattate esclusivamente con la webcam, ma richiederebbe un gran numero di foto dato che il dataset attuale è molto grande; Andando a riaddestrare la rete neurale creando un nuovo modello potrebbe migliorare l'accuratezza della rilevazione automatico perché le caratteristiche apprese sono differenti rispetto ad una foto in alta qualità.

Una possibile alternativa è quella di diminuire di una percentuale X le foto del dataset e sostituirle con le foto scattate tramite la webcam cosi da differenziare le tipologie di foto e qualità, oppure di aggiungere le foto al dataset senza cancellare le foto attuali, ovviamente ogni foto deve essere etichettata con l'emozione dell'utente, e addestrare nuovamente la rete neurale con il nuovo dataset e ricontrollare i risultati e l'accuratezza.

Un altro aggiornamento è quello di utilizzare i secondi misurati durante la lettura della trama da parte dell'utente per raccomandare altri titoli all'utente. Attualmente all'interno del database è stata creata una tabella dedicata per salvare il timer per ogni film e serie tv interessata e nel server scritto in javascript è stata implementata la funzione per misurare i secondi passati a leggere la trama. La funzione viene attivata ogni qualvolta l'utente preme il pulsante per leggere la trama, quando l'utente smette di leggere la trama, la funzione ferma il timer e invia i secondi, il titolo del film o serie tv e l'email dell'utente al database il quale inserisce i dati se sono nuovi, o vengono aggiornato se invece sono già presenti in esso. Tutto quello che è possibile fare è ordinare i dati presenti nel database in ordine decrescente, prendere X film o serie tv con un tempo di lettura alto e inviarlo al server di python per effettuare il calcolo delle raccomandazioni mediante l'uso dei moduli. Dopo aver effettuato la raccomandazioni, costruire le varie schede per ogni film con i dati mancanti quali trailer, locandina ecc… e inviare le schede al client dove verranno mostrate mediante interfaccia utente.

Una nuova feature che è possibile implementare in una prossima versione è registrare il tempo di permanenza del cursore del mouse sulla locandina di un film o di una serie tv.

Questo potrebbe essere facilmente implementato in javascript, più nello specifico in un componente di React, dove già risiede la funzione per misurare il tempo di lettura della trama tramite gli eventi, in questo caso l'evento onMouseOver. La logica potrebbe essere la stessa per la lettura della trama, viene avviato il timer, quando il mouse si sposta dalla locandina viene inviato il titolo, tempo e email dell'utente al database per memorizzarlo o aggiornarlo. Per effettuare le raccomandazioni con questi nuovi dati possiamo inviare i dati al server di python dopo averli ordinati in ordine decrescente sul tempo di visualizzazione, e effettuare le raccomandazioni con i moduli. Questo metodo potrebbe fornire al sistema un possibile feedback implicito dell'utente, ma non è attendibile perché molte volte l'utente potrebbe muove il mouse in modo casuale focalizzandosi altrove.

Una possibile nuova aggiunta è quella di una chat gestita da un agente automatico che aiuta l'utente ad effettuare nuove visioni di titoli, oppure chiedere tramite determinate domande all'utente delle informazioni per capire lo stato d'animo, con chi si trova l'utente, cosa sta facendo, cosa ha fatto in precedenza e consigliare la visione di film o serie tv.

Riferimenti

1: J. Bobadilla, F. Serradilla, A. Hernando, MovieLens, << Collaborative filtering adapted to recommender systems of e-learning >>, Knowledge-Based Systems, pp. 261- 265,2009

2: J. Jordán, S. Valero, C. Turró, V. Botti, << Using a Hybrid Recommending System for Learning Videos in Flipped Classrooms and MOOCs >>, Electronics, pp. 1 – 10,2021

3: N.N. Qomariyah, D. Kazakov, << A genetic-based pairwise trip planner recommender system >>, J Big Data, pp. 1 - 17,2021

4: M. J. Awan, R. A. Khan, H. Nobanee, A. Yasin, S. M. Anwar, U. Naseem, V. P. Singh, << A Recommendation Engine for Predicting Movie Ratings Using a Big Data Approach >>, Electronics, pp. 1 - 17,2021

5: T. Tran, R. Cohen, << Hybrid Recommender Systems for Electronic Commerce >>, AAAI Workshop, pp. 1-7,2000

6: M. Barros, A. Mitinho, F. M. Couto, << Hybrid semantic recommender system for chemical compounds in large-scale datasets >>, Journal of Cheminformatics, pp. 1-18,2021

7: M. Wang, M. Gong, X. Zheng, K. Zhang, << Modeling Dynamic Missingness of Implicit Feedback for Recommendation >>, Adv Neural Inf Process Syst, pp. 1-18, 2018

8: D. M. Blei, A. Y. Ng, M. I. Jordan, << Latent Dirichlet Allocation >> , Journal of Machine Learning Research, pp. 1-30,2003

9: J. L. Kolodner, << Instructional Design: Case-Based Reasoning >> ,College of Computing, pp. 1-13,2002

10: A. Gatzioura, J. Vinagre, A. M. Jorge, M. Sànchez-Marrè, << A Hybrid Recommender System for Improving
Automatic Playlist Continuation >>, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, pp. 1-12, 2019

11: H. W. Cho, << Topic Modellng >>, Osong Public Health and Research Perspectives, pp. 115-116, 2019

12: A. Rivas, A. González-Briones, J. J. Cea-Morán, A. Prat-Pérez, J. M. Corchado, << My-Trac: System for Recommendation of Points of Interest on the Basis of Twitter Profiles >>, Electronics, pp. 1-19, 2021

13: Md. Abu Kausar, V.S. Dhaka, S. K. Singh, << Web Crawler: A Review >>, International Journal of Computer Applications, pp. 1-7,2013

14: I.T. Afolabi, O.S. Makinde, O.O. Oladipupo, << Semantic Web mining for Content-Based
Online Shopping Recommender Systems >>, International Journal of Intelligent Information Technologies , pp. 41 - 56, 2019

15: S. Bostandjiev, J. O'Donovan, T. Höllerer, << TasteWeights:
A Visual Interactive Hybrid Recommender System >>, RecSys '12: Sixth ACM Conference on Recommender Systems, pp. 35-41, 2012

16: K. Kim, J. Kim, M. Kim, M. Sohn, << User interest-based recommender system
for image-sharing social media >>, Springer Science+Business Media, pp. 1-23, 2020

17: S. K. Addagarla, A. Amalanathan, << e-SimNet: a visual similar product recommender system for E-commerce>>, Indonesian Journal of Electrical Engineering and Computer Science, pp. 563-570, 2021

18: A. Ghazimatin, R. S. Roy, S. Pramanik, G. Weikum, << ELIXIR: Learning from User Feedback on Explanations
to Improve Recommender Models >>, Proceedings of the Web Conference 2021, pp. 1-11,2021

19: Q. Y. Shambour, N. M. Turab, O. Y. Adwan, << An Effective e-Commerce Recommender System Based on Trust and Semantic Information >>,CYBERNETICS AND INFORMATION TECHNOLOGIES, pp. 1-16, 2021

20: L. Chen, Y. Yuan, J. Yang, A. Zahir, << Improving the Prediction Quality in Memory-Based Collaborative Filtering Using Categorical Features >>, Electronics, pp. 1-17,2021

21: B. Jeong, J. Lee, H. Cho, << Improving memory-based collaborative filtering via similarity updatingand prediction modulation >>, Information Sciences, pp. 602-612,2009

22: D. B. Osario, M. P. Ortiz, C. R. Armengot, A. Colino, << Web 5.0: the future of emotional competences in highereducation >>,International Network of Business and Management, pp. 274-287, 2013

23: K. Sailunaz, R. Alhajj, << Emotion and sentiment analysis from Twitter text >>, Journal of Computational Science,, 2019

24: S. S. Nisha e M. N. Meeral, << Applications of deep learning in biomedical engineering >>,, 245 - 270, 2021

25: Qiang Ji, << Computer vision applications >>, Probabilistic Graphical Models for Computer Vision, 191-297,2020

26: P. Nandwani e R. Verma, << A review on sentiment analysis and emotion detection from text >>, Social Network Analysis and Mining,1 - 19, 2021

27: M. Singh, A. K. Jakhar, S. Pandey, << Sentiment analysis on the impact of coronavirus in social life using >>, Social Network Analysis and Mining volume, 1 - 11, 2021

28: L. Do, H. Yang, H. Nguyen, S. Kim, G. Lee, I. Na, << Deep neural network-based fusion model for emotion >>,The Journal of Supercomputing, 1 - 18, 2021

29: A. Agrima, I. Mounir, A. Farchi, L. Elmaazouzi, B. Mounir, << Emotion recognition from syllabic units using k-nearestneighbor classification and energy distribution >>,International Journal of Electrical and Computer Engineering, 5438 -5449,2021