Elaborato Reti di calcolatori I

Marco Di Fiandra matricola: N46004081 | Reti di calcolatori | Prof. Canonico Roberto |
05/07/2020

Indice

Introduzione e Traccia

Il progetto consiste nella realizzazione di una chat multiutente usando una architettura client-server utilizzando il linguaggio python e gli strumenti da quest'ultimo offerto per la programmazione delle socket.

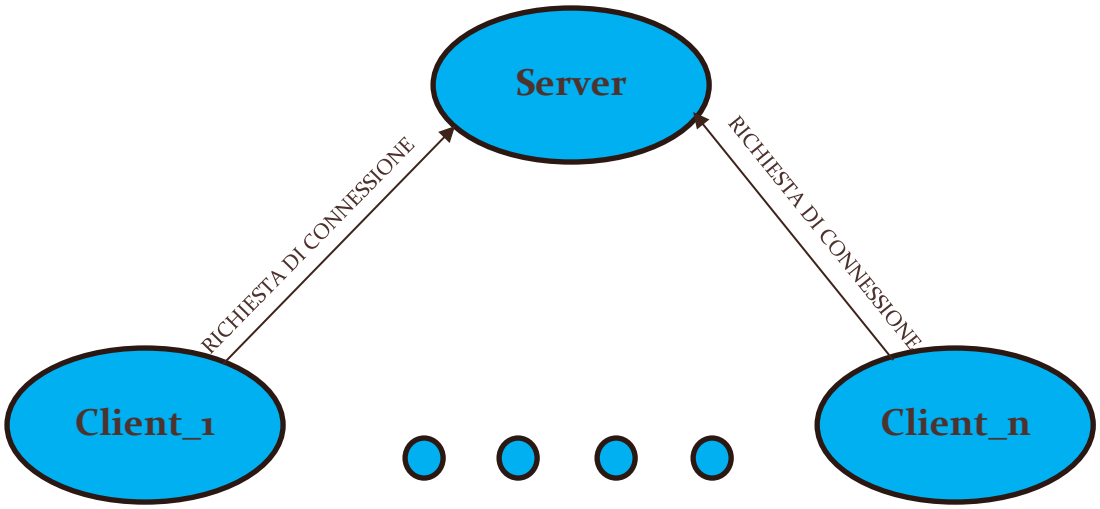
Si vuole anche realizzare un processo di analisi dati attraverso opportuni modelli di machine learning riguardanti il phrase labelling di modo da tracciare un istogramma delle preferenze dell'utente del quale si è interessati ricevere informazioni.

Per rendere l'applicazione più performante si fa uso anche del meccanismo di multithreading messo a disposizione di python.

Struttura di base

L'architettura di base del progetto fa uso del classico pattern client-server dove appunto un processo server accetta le connessioni richieste dai vari client, permettendo a questi ultimi di interagire attraverso la chat

Di seguito viene mostrata tale architettura a livello grafico



Il server dopo la richiesta di connessione manda una richiesta di nome che sta ad indicare a quest'ultimo con quale identificativo il client ha intensione di connettersi alla chat.

Una volta mandata tale richiesta il thread interessato alla comunicazione rimane in attesa del messaggio di risposta del client.

Una volta che il server riceve il messaggio di risposta, esso inoltra in broadcast il messaggio "id has joined the chat" dove id è il nome indicato dal client per connettersi alla chat.

Per lo sviluppo di tale progetto si è scelto l'uso delle socket TCP (che in python rientrano col nome di socket SOCK_STREAM).

Tale scelta è stata guidata dal fatto che TCP è un protocollo affidabile e connection oriented il quale offre benefici sul corretto invio dei pacchetti in rete verso il destinatario.

Data la scelta di utilizzare le socket TCP avremo bisogno di utilizzare una socket per l'ascolto di eventuali richieste di connessione e un'altra socket per gestire la comunicazione data la natura di TCP.

Sul server oltre ad essere presenti i vari thread per le comunicazioni, saranno presenti anche altri due thread di utilità.

Il primo thread sarà preposto all'analisi delle frasi scritte nella chat

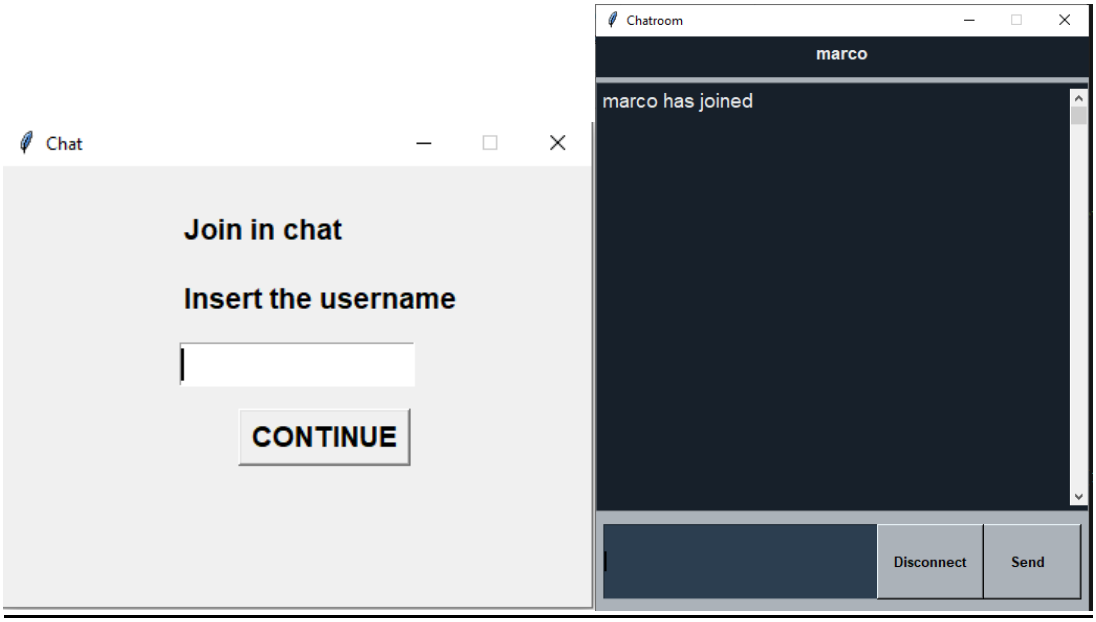
Il secondo thread sarà in ascolto sul canale di I/O in attesa del comando che permette di creare l'istogramma richiesto

Per quanto riguarda il client, quest'ultimo sarà provvisto di una GUI realizzata con la libreria tkinter di python.

Al client saranno già noti l'indirizzo del server al quale connettersi e la porta sulla quale indirizzarsi per contattare il server.

All'avvio dell'applicativo client verrà chiesto di inserire l'id col quale connettersi alla chat, dopo l'immissione di quest'ultimo si entrerà effettivamente nella chat

Di seguito alcuni screen dei passi appena descritti



Ogni messaggio inviato dal client sarà trasmesso in broadcast a tutti i client connessi alla chat.

Il client può anche decidere di abbandonare la chat attraverso l'operazione di disconnessione, la quale comporterà la chiusura della socket sia del client, sia la socket server relativa a tale connessione, attraverso l'invio di uno speciale messaggio; tale disconnessione sarà poi notificata a tutti i client ancora attivi.

File utilizzati

Server.py

Tale file contiene tutte le strutture e funzioni per la gestione del processo server.

Esso è composto dalle seguenti strutture:

classe Person_data: Struttura riportante quelli che sono gli attributi del dominio di pertinenza riguardante gli utenti della chat, in particolare riporta il loro id ed anche le occorrenze degli argomenti citati da quella persona

Enumerazione Prediction: Tale struttura enumera le varie categorie predicibili dal modello di ML utilizzato

Classe PhraseBuffer: Tale struttura è il buffer di comunicazione tra il thread che analizza le frasi ed i produttori di tali frasi, ovvero i thread hadler delle connessioni. Tale comunicazione è sviluppata attraverso il paradigma produttore-consumatore utilizzando una condition variable.

Le funzioni utilizzate in tale modulo sono:

connectionHandler(threadName,ConnectionSoket,addr):

Tale funzione rappresenta la logica applicativa per la gestione di una generica connessione, essa servirà come corpo ai vari thread di gestione

broadcast(message): Tale funzione permette di inoltrare in broadcast i messaggi ricevuti dal server

PredictionHandler(): Tale funzione rappresenta la logica applicativa per effettuare predizioni attraverso il modello ML

adottato, come detto in precedenza sarà il corpo del thread di analisi dei dati

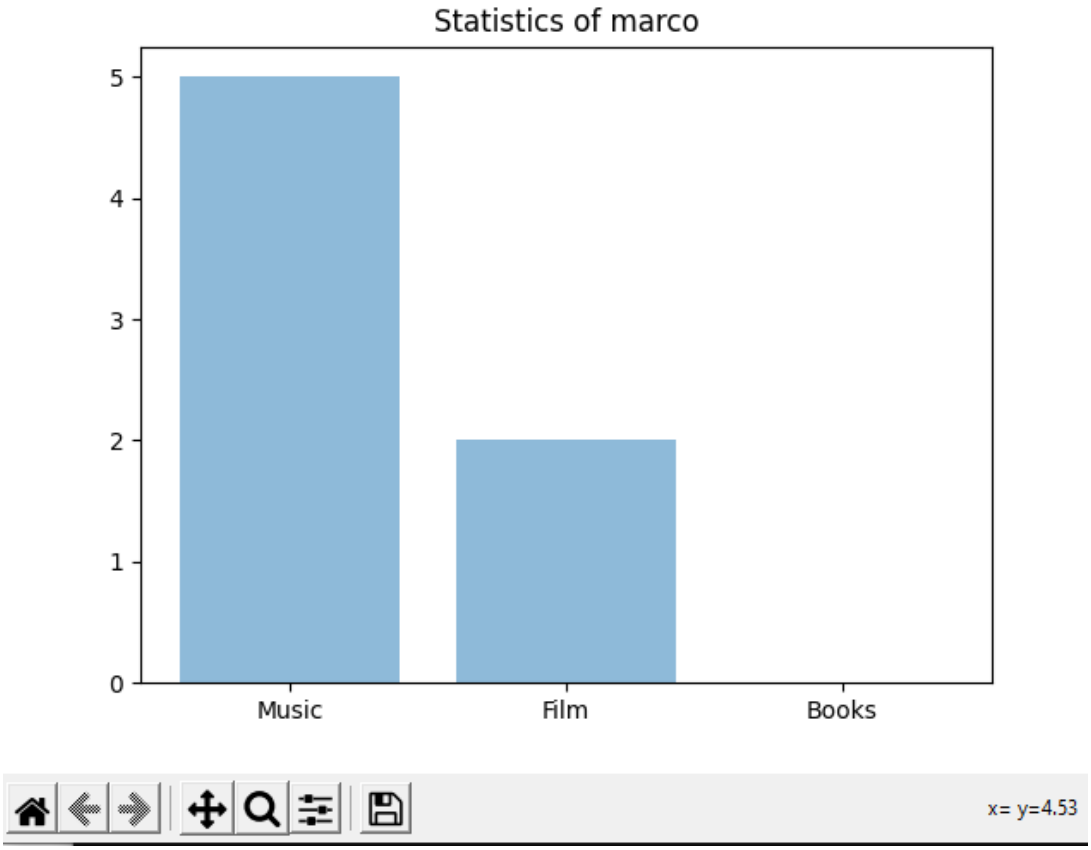
createStatisticGraph(): Tale funzione rappresenta la logica applicativa riguardante la creazione dei grafici di una specifica persona, in caso di omonimi verranno creati entrambe i grafici

Per creare un grafico relativo ad un client nella chat bisogna inserire il seguente comando:

makeHisto {id} #id va sostituito col l'id dell'utente nella chat

l'output sarà un istogramma degli argomenti trattati da questa persona nella chat

Di seguito viene mostrato un esempio di output.



makePrediction(name,message): Tale funzione è preposta alla produzione di predizioni riguardanti la frase in input

createStatistics(person): Tale funzione produce l'istogramma precedentemente mostrato attraverso l'uso della libreria matplotlib

searchByName(name): Tale funzione permette di ricercare una persona avente l'id in input all'interno di un array di persone connesse alla chat, tale funzione in caso di esito positivo ritorna l'insieme degli oggetti richiesti (multipli in caso di id ridondante), nel caso negativo ritorna None

Client.py

Tale file si occupa della gestione dell'interazione tra client e server.

Per la creazione dell'interfaccia grafica si è deciso di produrre una classe, la classe **GUI,** che avesse al suo interno tutte le routine di gestione dell'interfaccia grafica e della connessione

Per tale file ci soffermeremo su quelle che sono le routine di gestione della comunicazione, tralasciando le routine di gestione dell'interfaccia grafica.

sendMessage(self): Tale funzione permette di inviare tramite la socket di connessione col server il messaggio immesso nella texbox dell'interfaccia grafica, ad ogni invio ci sarà un thread che sarà creato per gestire l'evento di invio.

receiveMessage(self): Tale funzione permette di ricevere un messaggio dal server e di mostrarlo poi sull'interfaccia grafica. Tale funzione è il corpo di un thread attivo sulla macchina client sempre in ascolto sul canale di comunicazione TCP. Tale canale può prevedere 3 tipi di messaggi.

NAME: Messaggio di richiesta di id da parte del server

END: Messaggio di chiusura della connessione, tale messaggio è inviato dal server, dopo la deallocazione delle risorse relative a tale connessione, per chiudere definitivamente la comunicazione

disconnect(self): Tale funzione permette al client di abbandonare la chat in modo pulito, tale abbandono è notificato al server con uno speciale messaggio ("**$exit$**") che permette a quest'ultimo di deallocare le risorse.

Model.py

Tale file contiene tutte le istruzioni per caricare il modello di analisi dei dati costruito ed usarlo.

Tale modello è stato creato sempre attraverso l'uso di python ed il codice per la creazione e il train di tale modello è ritrovabile nel file **ReteNeurale.py**

Per la costruzione del modello è stato utilizzato un paper disponibile su kaggle nel quale vi erano spezzoni di articoli di giornale con il relativo topic a cui erano relativi.

Il modello dispone della definizione di sole 3 categorie: film,libri,musica, data questa limitazione l'analisi delle varie frasi verterà solamente su tali categorie.

Parlando di questo modello supporremo che il modello sia già stato compilato e salvato, tale modello va sotto il nome di "**Model_Trained_num_iter_3000**" caricabile tramite il modulo pickle di python.

In tale modulo troviamo la seguente funzione:

elaborazione(message): Tale funzione permette la processazione del messaggio ricevuto come input, si suppone che precedentemente si sia fatta un operazione di creazione di vocabolario attraverso la funzione di vettorizzazione.