

Relazione Progetto Ingegneria dei Sistemi Distribuiti

Image Classification

Giovanni Imbesi

Matricola: 1000006253

Prof. Emiliano Tramontana
Prof. Andrea Fornaia

19 gennaio 2024

Indice

1	Introduzione	2
1.1	Problema	2
1.2	Motivazioni	2
2	Tecnologie Utilizzate	3
2.1	Satellite Image Downloader	3
2.1.1	Funzionamento	3
2.1.2	Utilizzo	4
2.2	Docker	4
2.3	RabbitMQ	5
3	Microservizi	6
3.1	Satellite Image Sender	6
3.1.1	Dettagli Implementativi	6
3.2	Satellite Image Consumer	7
3.2.1	Dettagli Implementativi	7
4	Considerazioni Finali	9
4.1	Demo	9
4.2	Sviluppi Futuri	10
	Bibliografia	11

Capitolo 1

Introduzione

In questo elaborato, verrà affrontato il problema dell’Image Classification mediante la progettazione e implementazione di due microservizi. L’obiettivo primario è quello di automatizzare il processo di acquisizione, suddivisione e classificazione di immagini satellitari, sfruttando una modalità di comunicazione asincrona.

1.1 Problema

L’idea alla base del progetto consiste nell’analisi di immagini satellitari per l’estrazione di informazioni rilevanti. In particolare, si propone la suddivisione di immagini di grandi dimensioni in sotto-immagini più gestibili, seguita dalla classificazione dei singoli pixel in base alla loro composizione cromatica.

1.2 Motivazioni

Le motivazioni che hanno ispirato la realizzazione di questo progetto sono molteplici. L’ampia disponibilità di immagini satellitari rappresenta un ricco e vasto bacino di dati, che, se adeguatamente analizzato, può fornire una gamma infinita di informazioni utili in diversi campi quali monitoraggio ambientale, previsioni meteorologiche, studi agricoli, urbanistica, e molto altro.

Capitolo 2

Tecnologie Utilizzate

Nel corso dello sviluppo del progetto, sono state adottate diverse tecnologie che hanno contribuito al raggiungimento degli obiettivi prefissati. Di seguito, vengono presentate nel dettaglio.

2.1 Satellite Image Downloader

Satellite Image Downloader[1] è un'applicazione Python[3] progettata per scaricare regioni cartografiche rettangolari basate su coordinate geografiche. Tale applicativo opera attraverso il download di tessere della mappa a un determinato livello di zoom, successivamente ne ritaglia il bordo per adattarle alla regione di interesse e le salva come immagine PNG. Grande flessibilità è data dal fatto che esso è in grado di gestire qualsiasi mappa raster che utilizzi Web Mercator, incluse Google Maps, Esri e OpenStreetMap.

2.1.1 Funzionamento

Una volta installate le dipendenze necessarie ed avviato il servizio, occorre inserire le proprietà della regione desiderata:

- Coordinate dei vertici superiore sinistro e inferiore destro di una regione rettangolare, espresse in gradi decimali (latitudine prima della longitudine). Ad esempio: 40.612123, -73.895381.
- Livello di zoom desiderato per ottenere l'immagine. Questo parametro influisce sulla scala e sulla risoluzione dell'immagine. Il suo range può andare da un minimo di 1 ad un massimo di 19.

Di seguito viene presentato un esempio di immagine ottenuta tramite il servizio.



Figura 2.1: esempio immagine satellitare

2.1.2 Utilizzo

L'applicativo presentato costituisce la componente centrale del progetto per ottenere le immagini satellitari. Il codice relativo a questa funzionalità è stato integrato all'interno del servizio *Satellite Image Sender*, consentendo agli utenti di inviare immagini satellitari in modo asincrono al servizio *Satellite Image Consumer* il quale è responsabile dell'analisi delle immagini ricevute e della generazione di statistiche dettagliate basate sui colori dei pixel.

2.2 Docker

Docker[2] rappresenta una componente chiave per la realizzazione e distribuzione del progetto, fornendo un ambiente di esecuzione isolato per ciascun microservizio coinvolto. La containerizzazione permette di creare, distribuire ed eseguire applicazioni in modo consistente e affidabile, garantendo al contempo la portabilità su diverse piattaforme.

È stata creata una immagine docker per ciascun microservizio incorporando le dipendenze necessarie, il runtime e le librerie richieste per il corretto funzionamento. Essi comunicano, con un container RabbitMQ il quale fornisce un meccanismo efficiente per la gestione della coda di messaggi asincroni tra i microservizi.

Tutti i container vengono avviati all'interno di una rete chiamata *satellite_network*, che consente una connettività agevole tra i vari componenti del sistema. Questo approccio favorisce la coesione dell'architettura distribuita del progetto, contribuendo a una maggiore scalabilità e una gestione più efficiente delle risorse.

2.3 RabbitMQ

RabbitMQ[4] è un middleware di messaggistica open source che implementa il protocollo Advanced Message Queuing Protocol (AMQP). Nel progetto, RabbitMQ è stato adottato per implementare la comunicazione asincrona tra i due microservizi. Satellite Image Sender invia le immagini alla coda di RabbitMQ, le quali vengono lette dal servizio Satellite Image Consumer per l'analisi.

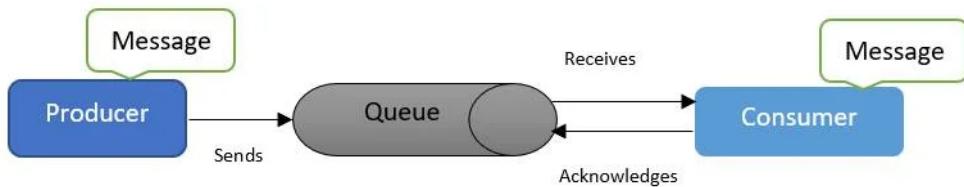


Figura 2.2: architettura producer-consumer[5]

Capitolo 3

Microservizi

In questo capitolo verranno descritti nel dettaglio i microservizi realizzati per l'elaborazione delle immagini satellitari.

3.1 Satellite Image Sender

Come anticipato, Satellite Image Sender si occupa di scaricare, suddividere ed inviare le immagini satellitari, attraverso l'applicativo *Satellite Image Downloader*. Di seguito verranno discusse le scelte progettuali ed i dettagli implementativi del servizio.

3.1.1 Dettagli Implementativi

Il Satellite Image Sender si basa su un file di configurazione denominato *preferences.json*, che gestisce le preferenze dell'utente, come l'URL del servizio di mappe, la dimensione delle tessere, il numero di canali e altri parametri. Il processo di invio delle immagini avviene attraverso i seguenti passaggi:

- **Input da utente:** L'utente fornisce le coordinate geografiche del vertice in alto a sinistra e in basso a destra della regione rettangolare, il livello di zoom desiderato e il numero di sotto-immagini in cui dividere l'immagine.
- **Download dell'immagine:** L'applicativo utilizza il modulo *image_downloading* di *Satellite Image Downloader* per scaricare l'immagine satellitare desiderata.
- **Divisione in sotto-immagini:** L'immagine viene suddivisa in un numero specificato di sotto-immagini. Ciascuna di esse rappresenta una porzione della regione originale.

- **Conversione in sequenza di byte:** Prima dell'invio, le sotto-immagini attraversano un processo di trasformazione. Utilizzando la libreria OpenCV, ciascuna sotto-immagine viene convertita nel formato PNG per garantire la trasmissibilità attraverso la coda di messaggi di RabbitMQ. Questa procedura impiega un processo di rappresentazione dell'immagine sotto forma di sequenza di byte. L'output di questa conversione è poi vincolato a un identificativo univoco, composto da un timestamp e un numero sequenziale. Questi parametri agiscono come chiave distintiva per il consumatore, agevolando l'identificazione delle sotto-immagini nel contesto del flusso asincrono.
- **Invio a RabbitMQ:** Ciascuna sotto-immagine viene inviata al servizio consumer attraverso la coda di messaggi chiamata *satellite-image-queue*.

3.2 Satellite Image Consumer

Satellite Image Consumer si occupa di ricevere sottoimmagini dalla coda RabbitMQ e, per ognuna di esse, classificare i pixel in base al colore. Di seguito verrà analizzato il funzionamento e le scelte implementative del servizio.

3.2.1 Dettagli Implementativi

Ciascun servizio consumer prevede i seguenti step:

- **Connessione a RabbitMQ:** Il servizio consumer stabilisce una connessione a RabbitMQ attraverso la coda di messaggi denominata *satellite-image-queue*, su cui il servizio consumer rimarrà in ascolto per la ricezione delle sotto-immagini.
- **Classificazione dei colori:** Ogni pixel della sottoimmagine viene confrontato con una lista predefinita di colori tra cui rosso, verde, blu, giallo, ciano, magenta, nero e bianco. Il confronto avviene mediante il calcolo della distanza euclidea tra la terna RGB del pixel e le terne RGB dei colori nella lista. Il colore più vicino è quindi assegnato al pixel. L'utilizzo della distanza euclidea nella classificazione dei colori fornisce una misura della similarità tra il colore del pixel e i colori predefiniti. Questo approccio permette di ottenere una classificazione accurata, specialmente in presenza di sfumature e tonalità leggermente diverse. La distanza euclidea è calcolata utilizzando la seguente formula:

$$\text{Distanza Euclidea} = \sqrt{(R1 - R2)^2 + (G1 - G2)^2 + (B1 - B2)^2}$$

Dove (R_1, G_1, B_1) e (R_2, G_2, B_2) rappresentano le terne RGB del pixel in esame e di uno dei colori predefiniti.

- **Analisi della luminosità:** Per distinguere se un pixel ha una tonalità chiara o scura, viene effettuata un’analisi della luminosità. Essa è calcolata utilizzando una combinazione lineare dei valori RGB ponderati. In particolare, la formula utilizzata è:

$$\text{Luminosity} = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Dove R , G , e B sono i valori delle componenti RGB del pixel. Il risultato della luminosità viene confrontato con una soglia (127.5) per determinare se il pixel è considerato chiaro o scuro.

- **Conferma di ricezione:** Dopo aver completato l’analisi di una sotto-immagine, il servizio consumer invia una conferma di ricezione a RabbitMQ. Ciò avviene attraverso il metodo *basic_ack*, che informa RabbitMQ che il messaggio è stato ricevuto correttamente e può essere rimosso dalla coda.

Il servizio consumer è progettato per essere scalabile, consentendo l’istanziazione di più container che consumano dalla stessa coda di messaggi asincroni, garantendo così un’analisi efficiente delle sotto-immagini inviate dal Satellite Image Sender.

Capitolo 4

Considerazioni Finali

4.1 Demo

Di seguito viene mostrato un esempio della scomposizione ed analisi cromatica di una immagine satellitare raffigurante un'area nel centro storico di Roma.

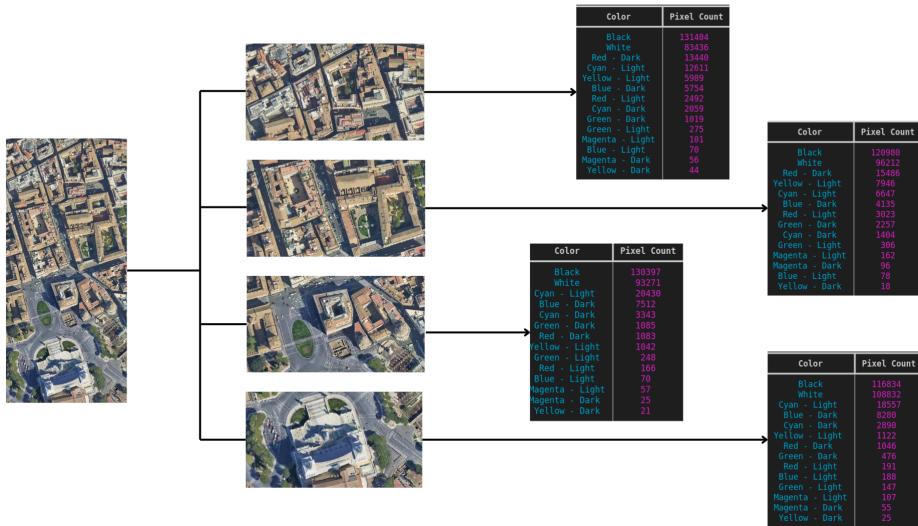


Figura 4.1: pipeline di esempio

È possibile osservare come in generale le statistiche rispecchino le tonalità cromatiche di ciascuna sottoimmagine.

4.2 Sviluppi Futuri

Il progetto può essere esteso introducendo un'interfaccia grafica che semplifichi l'interazione con l'applicazione. Essa potrebbe consentire agli utenti di selezionare agevolmente le regioni di interesse sulla mappa, visualizzare le sotto-immagini elaborate e personalizzare le preferenze di analisi.

Inoltre, l'integrazione di algoritmi di machine learning avanzati potrebbe migliorare la capacità del sistema di analizzare e comprendere le immagini. Ad esempio, l'implementazione di reti neurali convoluzionali (CNN) potrebbe consentire una classificazione più sofisticata degli oggetti presenti nelle sotto-immagini.

Queste implementazioni potrebbero offrire agli utenti una maggiore flessibilità e potenza analitica, rendendo l'applicazione più adattabile a una vasta gamma di scenari e esigenze specifiche.

Bibliografia

- [1] andolg. *Satellite Imagery Downloader*. URL: <https://github.com/andolg/satellite-imagery-downloader>.
- [2] *Docker Documentation*. Docker, Inc. URL: <https://docs.docker.com/>.
- [3] *Python Documentation*. Python Software Foundation. URL: <https://docs.python.org/3/>.
- [4] *RabbitMQ Documentation*. Pivotal Software, Inc. URL: <https://www.rabbitmq.com/documentation.html>.
- [5] *RabbitMQ Image Example*. Pivotal Software, Inc. URL: <https://roytuts.com/spring-boot-rabbitmq-producer-consumer-example/>.