Documentazione Programmi Python: Analisi di Similarità Multimodale

Analisi CLAP (Audio-Testo) e CLIP (Immagine-Testo)

22 maggio 2025

Sommario

Questo documento fornisce una documentazione dettagliata per due programmi Python distinti. Il primo programma utilizza il modello **CLAP (Contrastive Language-Audio Pretraining)** per analizzare la somiglianza semantica tra brani audio e le loro descrizioni di genere. Il secondo programma impiega il modello **CLIP (Contrastive Language-Image Pretraining)** per valutare la somiglianza tra immagini e i loro titoli. Entrambi i programmi estraggono embedding, calcolano la somiglianza coseno, visualizzano le distribuzioni e riassumono i risultati statisticamente, evidenziando esempi di corrispondenze migliori e peggiori.

1 Introduzione

Questa documentazione copre due script Python che esplorano la somiglianza semantica tra diverse modalità (audio-testo e immagine-testo) utilizzando modelli di deep learning pre-allenati: CLAP per l'audio e CLIP per le immagini. Entrambi i programmi seguono un flusso di lavoro simile: caricamento del modello, preparazione dei dati, estrazione degli embedding, calcolo della somiglianza e analisi dei risultati.

2 Librerie Comuni Necessarie

Prima di eseguire i programmi, assicurati di aver installato le seguenti librerie Python, comuni a entrambi gli script. Puoi installarle tramite `pip` dalla tua console o terminale:

- **Pytorch**: Un framework di deep learning. Entrambi i modelli (CLAP e CLIP) sono basati su PyTorch.

```
pip install torch torchvision torchaudio --index-url https://
    download.pytorch.org/whl/cu118

```

(Nota: la parte `--index-url` è per PyTorch abilitato per CUDA; regola se stai usando solo la CPU o una diversa versione di CUDA.)

- **pandas**: Per la manipolazione dei dati, specialmente per gestire i file di metadati.

```
pip install pandas

```

- **numpy**: Per le operazioni numeriche e l'analisi statistica.

```
pip install numpy

```

- **matplotlib**: Per tracciare gli istogrammi delle distribuzioni di somiglianza e visualizzare esempi.

```
pip install matplotlib

```

- **tqdm**: Per visualizzare le barre di avanzamento durante l'elaborazione dei dati, utile per dataset di grandi dimensioni.

```
pip install tqdm

```

3 Programma 1: Analisi di Similarità Audio-Testo con CLAP

Questo programma analizza la somiglianza tra brani audio e le loro descrizioni di genere utilizzando il modello CLAP.

3.1 Librerie Specifiche per CLAP

- **librosa**: Sebbene `msclap` gestisca il caricamento dell'audio, `librosa` è una libreria comune per l'elaborazione audio e potrebbe essere utile per altre attività audio.

```
pip install librosa

```

- **msclap**: La libreria principale per interagire con il modello CLAP.

```
pip install msclap

```

3.2 Descrizione Dettagliata del Programma CLAP

1. **Configurazione Iniziale**:
    - **Rilevamento Dispositivo**: Determina se utilizzare la GPU (CUDA) o la CPU.

- **Percorsi Critici**: Definizione dei percorsi per i metadati (`tracks.csv`), la cartella audio (`fma_small`) e i pesi del modello CLAP (`CLAP_weights_2023.pth`). ***Devi aggiornare questi percorsi affinchè corrispondano alla tua configurazione.***

2. **Caricamento del Modello CLAP**:
   - Carica il modello CLAP pre-allenato. Viene gestita l'eccezione in caso di errore nel caricamento del modello.
3. **Funzione per Ottenere il Percorso Audio**:
   - La funzione `get_audio_path(track_id)` costruisce il percorso completo di un file audio basandosi sull'ID del brano e sulla struttura del dataset FMA.
4. **Caricamento e Filtro dei Metadati**:
   - Carica il file `tracks.csv`.
   - Filtra i metadati per includere solo i brani del sottoinsieme `'small'` del dataset FMA.
   - Gestisce gli errori di file non trovato o altri problemi di caricamento/filtro.
5. **Ricerca dei Brani Audio Validi e Preparazione Dati**:
   - Itera attraverso gli ID dei brani nei metadati, verificando l'esistenza dei file audio.
   - Per ogni file esistente, estrae il genere principale come descrizione testuale.
   - Limita il numero di brani da controllare (`max_tracks_to_check`) e da processare (`max_tracks_to_process`) per ottimizzare i tempi.
6. **Estrazione degli Embedding Audio e Testo**:
   - Per ogni brano valido, usa il modello CLAP per generare **embedding audio** e **embedding di testo** dal genere corrispondente.
   - Gli embedding vengono raccolti in liste e poi convertiti in tensori PyTorch.
   - La gestione degli errori permette di saltare brani problematici.
7. **Calcolo della Similarità Coseno (Pura)**:
   - Normalizza gli embedding audio e di testo.
   - Calcola la somiglianza coseno come prodotto scalare degli embedding normalizzati.
   - I risultati vengono convertiti in un array NumPy.

8. **Visualizzazione della Distribuzione di Similarità**:
    - Genera un istogramma dei punteggi di somiglianza coseno per visualizzare la loro distribuzione.
9. **Statistiche**:
    - Calcola e stampa la media, la mediana e la deviazione standard dei punteggi di somiglianza.
10. **Visualizzazione Esempi Migliori e Peggiori**:
    - Ordina gli indici per somiglianza.
    - Stampa i dettagli (ID del brano, genere, somiglianza) per i 5 brani con la somiglianza più alta e i 5 brani con la somiglianza più bassa.

4 Programma 2: Analisi di Similarità Immagine-Testo con CLIP

Questo programma analizza la somiglianza tra immagini e i loro titoli utilizzando il modello CLIP.

4.1 Librerie Specifiche per CLIP

- **PIL (Pillow)**: Per il caricamento e la manipolazione delle immagini.

```
pip install Pillow

```

- **clip**: La libreria di OpenAI per il modello CLIP.

```
pip install openai-clip

```

- **pyarrow** e **fastparquet**: Necessari per leggere i file `.parquet`.

```
pip install pyarrow fastparquet

```

4.2 Descrizione Dettagliata del Programma CLIP

1. **Fix Conflitto OpenMP**:
    - Imposta la variabile d'ambiente `KMP_DUPLICATE_LIB_OK` a `"TRUE"` per risolvere potenziali conflitti con le librerie OpenMP, comuni in ambienti come Anaconda o con l'uso di TensorFlow/PyTorch.
2. **Percorsi Dati**:
    - Carica il dataframe dei metadati da un file `.parquet` (`artgraph_metadata.parquet`).

- Definisce il percorso della directory contenente i file immagine (`images_small`). ***Devi aggiornare questi percorsi affinchè corrispondano alla tua configurazione.***

3. **Filtra Immagini Esistenti e Titoli Validi**:
   - Filtra il dataframe dei metadati per includere solo le immagini i cui file esistono effettivamente nella directory specificata.
   - Rimuove le righe dove il titolo dell'opera d'arte (`ArtworkTitle`) è nullo.
4. **Caricamento Modello CLIP**:
   - Determina il dispositivo (`cuda` o `cpu`).
   - Carica il modello CLIP pre-allenato (`"ViT-B/32"`) e la sua funzione di preprocessamento (`preprocess`).
5. **Embedding e Similarità**:
   - Inizializza liste vuote per gli embedding delle immagini e del testo.
   - Itera su ogni riga del dataframe filtrato:
     - Carica e pre-processa l'immagine.
     - Tokenizza il titolo dell'opera d'arte.
     - Utilizza il modello CLIP per codificare l'immagine e il testo in vettori di embedding.
     - Memorizza gli embedding (convertiti in NumPy arrays) nelle rispettive liste.
     - La gestione degli errori permette di saltare le immagini o i testi che causano problemi.
6. **Calcolo Similarità Coseno**:
   - Converte le liste di embedding in array NumPy.
   - Normalizza gli array di embedding (norma L2).
   - Calcola la somiglianza coseno elemento per elemento (prodotto scalare degli embedding normalizzati).
7. **Visualizzazione Distribuzione**:
   - Genera e visualizza un istogramma della distribuzione dei punteggi di somiglianza.
8. **Statistiche**:
   - Stampa la media, la mediana e la deviazione standard dei punteggi di somiglianza.
9. **Visualizzazione Esempi Migliori e Peggiori**:

- Definisce la funzione `show_examples` per visualizzare le immagini con il loro titolo e punteggio di somiglianza.
- Ordina i punteggi di somiglianza.
- Mostra le 5 immagini con i punteggi di somiglianza più bassi (peggiori match) e le 5 immagini con i punteggi più alti (migliori match), insieme ai loro titoli e punteggi di somiglianza.

5 Utilizzo Generale

1. Assicurati di avere tutte le librerie elencate nella sezione `Librerie Comuni Necessarie` e le sezioni `Librerie Specifiche` per il programma che intendi eseguire installate.
2. Scarica e posiziona i dataset necessari (FMA per CLAP, ArtGraph per CLIP) e i pesi dei modelli nelle cartelle specificate.
3. **Modifica i percorsi dei file e delle cartelle all'inizio di ciascun script Python per riflettere la tua configurazione locale.** Questo è il passo più importante per far funzionare i programmi.
4. Esegui lo script Python desiderato.
5. Osserva l'output della console per i progressi, le statistiche e i messaggi di errore.
6. Le finestre di Matplotlib si apriranno per mostrare gli istogrammi e gli esempi di immagini (per il programma CLIP).

Questi programmi offrono un'ottima base per esplorare le capacità dei modelli multimodali CLAP e CLIP nella comprensione della relazione tra diversi tipi di dati.