AudioCLIP4Rec

Fontana Emanuele

October 16, 2025

Contents

1 Multimodal Recommender System

1.1 Recommender Systems

With **Recommender Systems** we refer to all those techniques that have the effect of guiding users in personalized way to interesting objects in a large space of possible options[5]. There are diffent types of Recommender Systems:

- **Content-Based Filtering**: This approach recommends items similar to those the user has liked in the past. It uses item features and user preferences to make recommendations.
- **Collaborative Filtering**: This method recommends items based on the preferences of similar users. It can be user-based or item-based.
  - **User-Based Collaborative Filtering**: This method finds users with similar preferences and recommends items they have liked.
  - **Item-Based Collaborative Filtering**: This method finds items similar to those the user has liked and recommends them. With respect to Content-Based Filtering, this method doesn't use item features to make recommendations.
- **Hybrid Methods**: These methods combine multiple recommendation techniques to improve accuracy and overcome the limitations of individual approaches.
- **Knowledge-Based Recommender Systems**: These systems use explicit knowledge about users and items to make recommendations.
- **Context-Aware Recommender Systems**: These systems take into account contextual information, such as time, location, or social context, to provide more relevant recommendations.

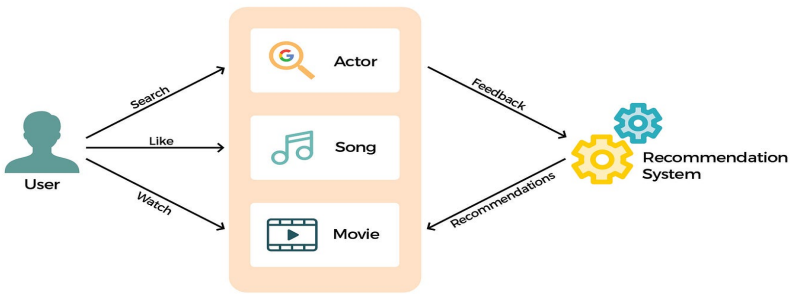

Figure 1: Conceptual representation of a Recommender Systems

Recommender Systems are not perfect. They can suffer of several problems like:

- **Cold Start Problem**: Because new users often have very few records in the system, it is hard to guess their preferences given the insufficient information[8]
- **Data Sparsity**: Most users use the system but do not give rating for feedback to the system in a proper way.[6]. In addition, in many real-word application we have thousand if not milion of user and item, so it's impossible to think that all user with interact with all item.
- **Scalability**: As the number of users and items increases, the computational complexity of recommendation algorithms can become a challenge. So we can divide scalability problems in two categories:
  - **Software Scalability**: There's a need to develop algorithms that can handle milions of users and items
  - **Hardware Scalability**: There's a need to create systems with powerful hardware that can handle the computational cost in time and space
- **Over Specialization** : Sometimes Recommender algorithms can "overfit" users' preferences, which lead to suggest items that are too similar to those already liked by the user not allowing him/her to discover new items that might be relevant

1.2 Multimodal Recommender Systems

Differently from classical recommendation approaches, multimodal recommendation exploits multimodal side information about items to enrich the interaction matrix, alleviating its sparsity, and to better understand the content to be recommended; these advantages, in general, lead to more accurate and precise recommendations[9]. We can think about different types of multimodal information: **Text** in natural language that can be used in every context, **Audio** for Music, **Images** for products such as clothes or fornitures, **Videos** for movies and so on.

The powerfulness of multimodal recommendation is given by the fact that different modalities can provide information about the same item from different perspectives. For example, in a movie recommender system, the textual description of a movie can provide information about its plot, screenshots

from the film can provide visual clues about the style of photography and the type of film (animated, live-action, stop-motion, etc.)

Some DNN architectures known as **Encoders** are used to extract features from different modalities that can be used in different ways like:

- **Concat**: Refers to the concatenation of multimodal features. [1]
- **Fusion and attention**: Refers to the attention mechanisms (like self-attention) to combine multimodal features. [4]
- **GNNs**: They use Graph Neural Networks to model the relationships between items and users, incorporating multimodal features into the graph structure

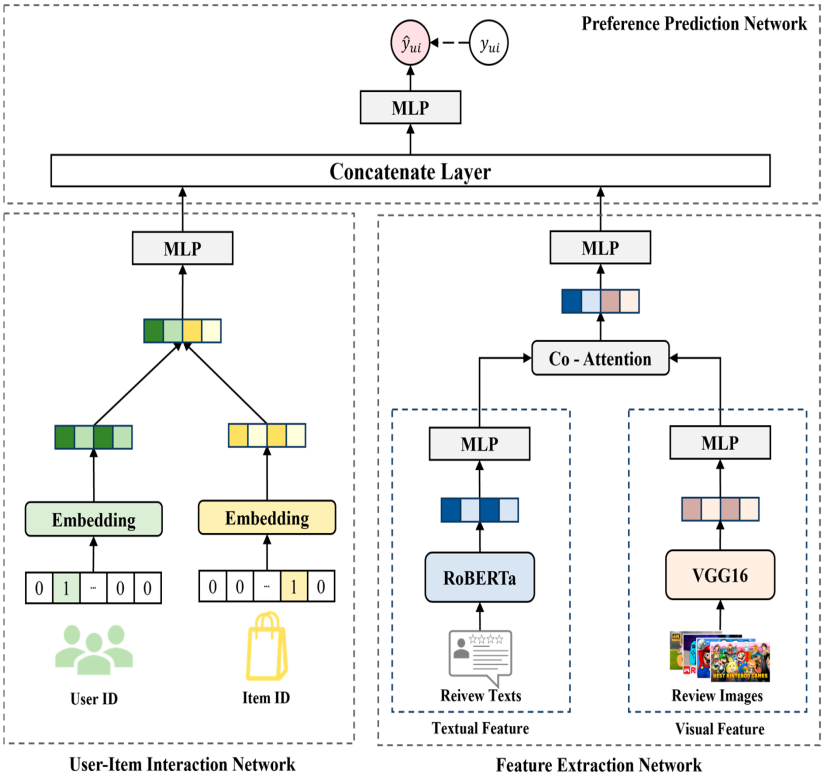

Figure 2: Conceptual representation of a Multimodal Recommender Systems

2 Contrastive Learning and AudioCLIP

2.1 Contrastive Learning

Contrastive Learning is self-supervised representation learning by training a model to differentiate between similar and dissimilar samples.[3]. In few words, contrastive learning is a technique used to learn a feature space where similar samples are close together and dissimilar samples are far apart. To achieve this, a specific loss function known as **Contrastive Loss** is used:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} [k \neq i] \exp(\text{sim}(z_i, z_k)/\tau)}, \tag{1}$$

where $\text{sim}(z_i, z_j)$ represents the similarity between two feature vectors z_i and z_j, τ is a temperature parameter, and N is the batch size. The numerator focuses on the similarity between the positive pair (z_i, z_j), while the denominator normalizes over all pairs excluding z_i itself.

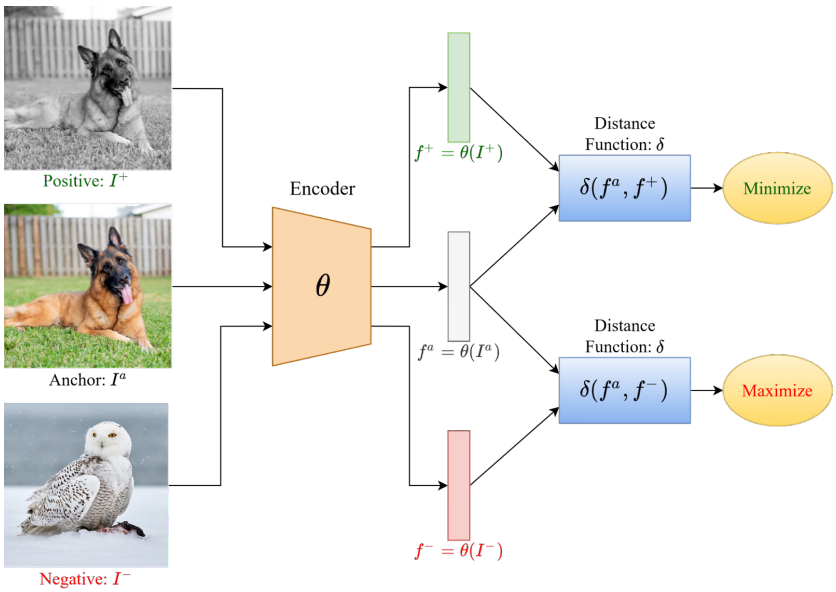

Figure 3: Conceptual representation of Contrastive Learning

2.2 CLIP

CLIP, which stands for Contrastive Language-Image Pre-training, is a neural network architecture developed by OpenAI[7] that uses Contrastive Learning to connect textual and visual information. The main idea behind CLIP is that visual and textual embeddings lives in the same feature space. Given

the textual description of an image and the image itself, the embeddings returned by the model should be close together in the feature space. In few words, CLIP is trained to predict which images are most relevant to a given text prompt, and vice versa.

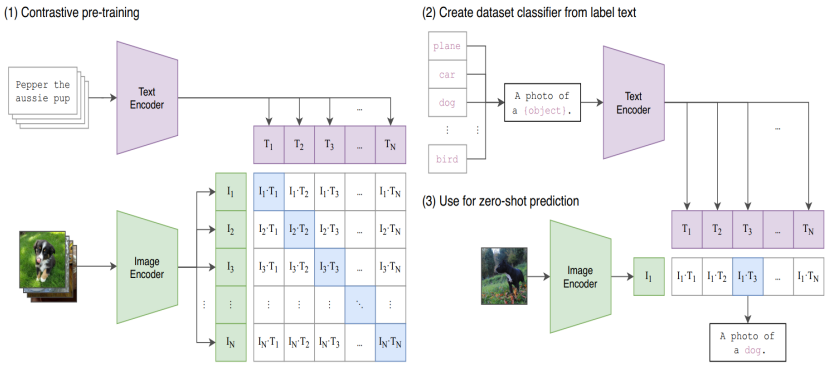

Figure 4: Conceptual representation of CLIP architecture

2.3 AudioCLIP

AudioCLIP [2] is an extension of the CLIP model that incorporates an additional audio modality, enabling it to understand and relate audio data alongside text and images. The architecture of AudioCLIP consists of three main components:

- **A CLIP Model** to handle text and image modalities. In particual a ResNet based CLIP model is used
- **Audio Encoder** to handle audio modality. In particular an ESResNetXt model is used

The idea to realize AudioCLIP is very simple: in addition to the text-image contrastive loss used in CLIP, two additional loss were added: **text-audio** and **image-audio** contrastive loss. In this way the three modalities are connected in the same feature space. The training procedure of AudioCLIP consists of different main steps:

- **Audio-Head Pre-Training**
  - **Standalone**: The audio head (based on ESResNeXt) is first pretrained independently on the *AudioSet* dataset.
  - **Cooperative**:

    * The classification layer of the pre-trained audio head is replaced with a randomly initialized layer whose output size matches CLIP's embedding space.
    * The audio head is then trained jointly with the frozen text and image heads, in a *multi-modal knowledge distillation* setup, making audio embeddings compatible with CLIP embeddings.

- **AudioCLIP Training**
  - After making the audio head compatible with CLIP, the entire tri-modal model (audio, text, image) is trained on *AudioSet*.
  - All three modality-specific heads are updated together, allowing the model to adapt to the distributions of audio samples, image frames, and textual class names.
  - This joint training improves performance compared to training only the audio head.

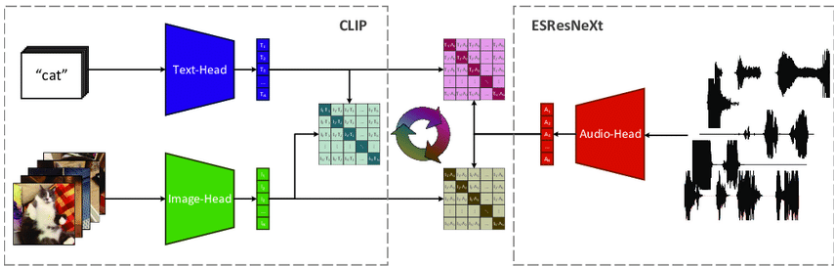

Figure 5: Conceptual representation of AudioCLIP architecture

3 Contrastive Learning arcitetures in Multimodal Recommender Systems

CLIP and AudioCLIP were born, and mainly used, for classification tasks. However, their ability to create a shared feature space for different modalities makes them suitable for multimodal recommender systems as well.

3.1 CLIP in Multimodal Recommender Systems

Zixuan Yi et al. [10] analyzed the use of CLIP in multimodal recommender systems. Up to few years ago, multimodal features were extracted using pretrained models on a single modality, such as ResNet for images and BERT for text. This leads to a problem: the features extracted from different modalities are not aligned in the same feature space. To solve this problem, they proposed to use CLIP to extract aligned image and text features. In particular, they used both a frozen and a fine-tuned CLIP model to extract image and text features. Experiments involved five multimodal recommender systems: VBPR, MMGCN, MMGCL , SLMRec and LATTICE and showed that both frozen and fine-tuned CLIP features significantly outperformed the traditional pretrained features in four out of five models. To be more specific, due to it's learning objective, LATTICE perform better without CLIP features. The fine-tune algorithm proposed by the authors is known as **END-TO-END Training**.

End-To-End Training The end-to-end fine-tuning procedure integrates the CLIP encoder directly into the recommendation model and jointly optimizes both components using recommendation losses. The training steps are as follows:

1. **Load Data:** Prepare the dataset by loading raw data
2. **Initialize CLIP Encoder:** Load a pre-trained CLIP model and its corresponding weights.
3. **Generate Embeddings:** Use the CLIP encoder to extract image embeddings and text embeddings from the dataset:
4. **Integrate Embeddings:** Feed the extracted embeddings into the recommendation model as initial item representations.
5. **Joint Optimisation:** For each training epoch, jointly update both the CLIP encoder and the recommendation model:

(a) Perform a forward pass to compute user–item scores.

(b) Compute the recommendation loss

(c) Backpropagate the loss and update both the CLIP encoder and recommendation model parameters:

6. **Evaluation:** After each epoch, evaluate and log recommendation performance metrics (e.g., NDCG, Recall) to monitor progress.

This end-to-end strategy allows the CLIP encoder to adapt its visual and textual representations specifically to the recommendation task, leading to improved alignment between modalities and better overall recommendation performance.

3.2 AudioCLIP in Multimodal Recommender Systems

In literature there are evidence of the use of AudioCLIP in multimodal recommender systems.

4 Dataset

The dataset used in this project is **MovieLens 1M**, a collection of explicit movie ratings widely used in recommender systems research.

4.1 Dataset Characteristics

MovieLens 1M contains:

- **1,000,209 explicit ratings**
- **6,040 users**
- **3,952 movies**

4.2 Multimodal Extension

In the context of this project, the MovieLens 1M dataset has been enriched with multimodal data for each movie:

- **Images**: movie posters or representative images (.jpg format)
- **Audio**: audio clips or associated soundtracks (.wav format)
- **Texts**: textual descriptions, synopses, or reviews (.txt format)

All multimodal files are organized with the same basename corresponding to the Movie ID, ensuring correspondence between different modalities. For example, for the movie with ID `1`:

- Image: `ml1m/_images/1.jpg`
- Audio: `ml1m/_audios/1.wav`
- Text: `ml1m/_texts/1.txt`

4.3 Multimodal Embeddings

The multimodal embeddings were extracted using the **AudioCLIP** (see Section 2.3)

4.3.1 Extraction Process

The embedding extraction process is implemented in a script, which ensures robust handling of multimodal data through the following pipeline:

File Discovery and Alignment The script first scans three directories with raw data and identifies files by their basename (e.g., movie ID). From the initial scan:

- 3,196 image files were found
- 3,535 audio files were found
- 3,197 text files were found
- 3,635 unique names across all modalities

The script performs an **intersection** of basenames to ensure only movies with *all three modalities* are processed, resulting in **3,096 candidate items**.

Per-Modality Extraction with Error Handling The extraction pipeline processes each modality with specific procedures:

- **Images**: Resized to 224×224, normalized with CLIP statistics, batch processed
- **Audio**: Sliding window approach (2s windows, 1s stride), mean aggregation across windows
- **Text**: Adaptive chunking for 77-token CLIP limit, sentence-based splitting,
- **Error handling**: Corrupted files logged and excluded from final dataset

Consistency Enforcement After extraction, the script enforces **strict alignment**:

- Only items successfully extracted in *all three modalities* are retained
- Failed extractions (e.g., 1 audio file corrupted) result in removal from all modalities
- Final output arrays have **identical row counts** and order

In the current extraction:

- 3,096 items had all three modalities available
- 1 audio file failed extraction (corrupted EOF)
- Final dataset: **3,095 items** with complete embeddings

4.3.2 Output Files

The extracted embeddings are saved in NumPy format (.npy)

- `images.npy`: visual embeddings, shape $(3095, 1024)$
- `audios.npy`: audio embeddings, shape $(3095, 1024)$
- `texts.npy`: textual embeddings, shape $(3095, 1024)$
- `item_features.csv`: mapping between Movie ID and array index
- **row** i in each file corresponds to the same movie

4.4 Dataset Filtering and Remapping

After extracting multimodal embeddings, the interaction dataset (`movielens_1m.inter`) undergoes a comprehensive filtering and remapping pipeline. This is necessary in order to properly use the MMRec framework

4.4.1 Pipeline Overview

The script first loads the original interaction file. Then, to ensure consistency with the extracted embeddings, the script filters interactions to include only movies present in `item_features.csv` (i.e., movies with all three modalities: audio, images, and text).

After this filtering:

- Valid items with complete multimodal features: **3,095**
- Remaining interactions: depends on how many ratings reference the 3,095 valid movies

Step 3: Core-k Filtering To address data sparsity and ensure sufficient interaction density, the script applies **5-core filtering**, an iterative process that removes users and items with fewer than $k = 5$ interactions until convergence.

Algorithm:

1. Count interactions per user and per item
2. Remove users with < 5 interactions
3. Remove items with < 5 interactions
4. Repeat steps 1–3 until no further removals occur (convergence)

Step 4: ID Remapping After filtering, user and item IDs may no longer be contiguous (e.g., user IDs might be [1, 5, 12, ...]). To ensure compatibility with matrix-based recommender systems, the script remaps all IDs to contiguous ranges starting from 0.

User remapping:

- Original IDs: arbitrary integers
- New IDs: $0, 1, 2, \ldots, m-1$ (where m is the number of users after filtering)

Item remapping:

- Original IDs: arbitrary integers
- New IDs: $0, 1, 2, \ldots, n-1$ (where n is the number of items after filtering)

Step 5: Embedding Reindexing The original `.npy` files (`audios.npy`, `images.npy`, `texts.npy`) are indexed according to the pre-filtering item set. After remapping, the script reconstructs the embedding arrays to match the new item indices:

1. For each new item index $i \in [0, n-1]$:
   - Lookup the original item ID from the inverse mapping
   - Find the old index in `item_features.csv`
   - Copy the corresponding embedding vectors from the original `.npy` files
2. Save the reindexed embeddings as:
   - `audio_filtered.npy`
   - `image_filtered.npy`
   - `text_filtered.npy`

Step 6: Output Files The script generates the following files:

- `movielens_1m_filtered.inter`: filtered interaction file with remapped IDs
- `audio_filtered.npy`: reindexed audio embeddings, shape $(n, 1024)$

- `image_filtered.npy`: reindexed image embeddings, shape $(n, 1024)$
- `text_filtered.npy`: reindexed text embeddings, shape $(n, 1024)$
- `user_mapping.csv`: mapping between old and new user IDs
- `item_mapping.csv`: mapping between old and new item IDs
- `item_features_filtered.csv`: updated item-to-index mapping

5 Experiments

5.1 MovieLens_1M

Here we can find a comparison between the best result of the baseline [9] and the best result of our modalities.

Model	Modality	rec@10	ndcg@10	map@10	rec@20	ndcg@20	map@20
VBPR	Text (baseline)	0.132	0.1975	0.0982	0.2123	0.2127	0.1358
VBPR	Text+Image+Audio	0.1570	0.1922	0.0969	0.2466	0.2147	0.0957

Table 1: Comparison between baseline and our multimodal approach on MovieLens-1M

References

[1] Xi Chen, Yangsiyi Lu, Yuehai Wang, and Jianyi Yang. Cmbf: Cross-modal-based fusion recommendation algorithm. *Sensors*, 21(16), 2021.

[2] Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. Audioclip: Extending clip to image, text and audio. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 976–980. IEEE, 2022.

[3] Haigen Hu, Xiaoyuan Wang, Yan Zhang, Qi Chen, and Qiu Guan. A comprehensive survey on contrastive learning. *Neurocomputing*, 610:128645, 2024.

[4] Peishan Li, Weixiao Zhan, Lutao Gao, Shuran Wang, and Linnan Yang. Multimodal recommendation system based on cross self-attention fusion. *Systems*, 13(1), 2025.

[5] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. *Content-based Recommender Systems: State of the Art and Trends*, pages 73–105. Springer US, Boston, MA, 2011.

[6] Nitin Mishra, Saumya Chaturvedi, Aanchal Vij, and Sunita Tripathi. Research problems in recommender systems. In *Journal of Physics: Conference Series*, volume 1717, page 012002. IOP publishing, 2021.

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[8] Yu Rong, Xiao Wen, and Hong Cheng. A monte carlo algorithm for cold start recommendation. In *Proceedings of the 23rd international conference on World wide web*, pages 327–336, 2014.

[9] Giuseppe Spillo, Elio Musacchio, Cataldo Musto, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. See the movie, hear the song, read the book: Extending movielens-1m, last.fm-2k, and dbbook with multimodal data. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, RecSys '25, page 847–856, New York, NY, USA, 2025. Association for Computing Machinery.

[10] Zixuan Yi, Zijun Long, Iadh Ounis, Craig Macdonald, and Richard Mccreadie. Large multi-modal encoders for recommendation. *arXiv preprint arXiv:2310.20343*, 2023.