



Assignment of master's thesis

Title: Training and Evaluation of Foundation Models for Recommender Systems
Student: Bc. Tomáš Laurin
Supervisor: doc. Ing. Pavel Kordík, Ph.D.
Study program: Informatics
Branch / specialization: Knowledge Engineering
Department: Department of Applied Mathematics
Validity: until the end of summer semester 2026/2027

Instructions

The thesis focuses on how to train and evaluate foundation models for recommender systems, with emphasis on robust multimodal item representations, meta-learning, and transfer between datasets and domains. The goal is to understand how backbone embeddings can adapt to specific recommendation tasks under a unified offline benchmark.

Main objectives:

- Survey foundational approaches in recommender systems and related foundation model literature, with a focus on backbone representations and their role in recommendation pipelines.
- Propose and implement training strategies for foundation models in recsys (e.g., multimodal item encoders, meta-learning and transfer learning across datasets and domains).
- Implement or adapt an evaluation framework for comparing different foundation models on standard recommender datasets, focusing on isolating the evaluation of the foundation model from that of downstream fine-tuned recommenders.
- Use the framework to benchmark how different backbone embeddings adapt to various downstream recommendation tasks and analyze their strengths, weaknesses, and practical implications.

Master's thesis

TRAINING AND EVALUATION OF FOUNDATION MODELS FOR RECOMMENDER SYSTEMS

Bc. Tomáš Laurin

Faculty of Information Technology
Katedra aplikované matematiky
Supervisor: doc. Ing. Pavel Kordík, Ph.D.
September 10, 2026

Czech Technical University in Prague

Faculty of Information Technology

© 2026 Bc. Tomáš Laurin. All rights reserved.

This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Information Technology. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).

Citation of this thesis: Laurin Tomáš. *Training and Evaluation of Foundation Models for Recommender Systems*. Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2026.

Contents

Acknowledgments	viii
Declaration	ix
Abstract	x
List of Abbreviations	xi
Introduction	1
1 Theoretical Background	3
1.1 Recommender Systems	3
1.1.1 Content-Based and Collaborative Recommendation	4
1.1.2 Evaluation of Recommender Systems	4
1.1.3 Challenges in Recommendation	6
1.2 Foundation Models	6
1.2.1 Adapting Foundation Models	7
1.2.2 Meta-Learning	8
1.3 Representation Learning for Recommendation	8
1.3.1 Multimodal Representation Learning	8
1.4 Learning Objectives from Interaction Data	10
1.4.1 Bayesian Personalized Ranking	10
1.4.2 InfoNCE Loss	11
1.4.3 ELSA Loss	11
1.4.4 Interaction-Derived Similarity Matching	11
1.5 Representation Analysis Methods	12
1.5.1 Cosine Similarity	12
1.5.2 Vision Transformer Attention	12
1.5.3 Uniform Manifold Approximation and Projection	13
1.5.4 K-Means Clustering	13
2 Related Work	15
2.1 Foundation-Model Encoders for Recommendation	15
2.1.1 Text Encoders	15
2.1.2 Vision, Audio, and Multimodal Encoders	16
2.1.3 Cold-Start and Cross-Domain Adaptation	17
2.1.4 Meta-Learning for Recommendation	17
2.2 Graph-Based Recommendation	17
2.3 LLM-Based and Generative Recommendation	18
2.4 Datasets and Evaluation Frameworks	18
2.4.1 Datasets for Foundation-Model Recommendation	18
2.4.2 Evaluation Frameworks	19

3	Recommender Embedding Evaluation Framework	21
3.1	Pipeline Overview	21
3.2	Configuration and Component Resolution	23
3.3	Data Model and Loading	23
3.4	Tasks and Task Views	24
3.4.1	Built-In Tasks	26
3.5	Backbone Training and Embedding Generation	27
3.5.1	Backbone Protocol	28
3.5.2	Trainer Protocol	28
3.5.3	Embedding Generation	29
3.6	Downstream Adaptation and Prediction	29
3.7	Evaluation and Result Persistence	30
3.8	Extensibility	30
4	Zero-Shot Evaluation of Foundation Models	33
4.1	Experimental Scope	33
4.2	Evaluated Models	33
4.2.1	Text Encoders	34
4.2.2	Vision Encoders	34
4.2.3	Multimodal Encoders	34
4.2.4	Baseline	34
4.3	Datasets	35
4.3.1	Dataset Preprocessing	35
4.4	Evaluation Scenarios	36
4.4.1	Standard recommendation	36
4.4.2	Cold-start recommendation	37
4.4.3	Sequential recommendation	37
4.4.4	Domain transfer	37
4.4.5	Dataset transfer	37
4.5	Results	38
4.5.1	Results on MovieLens 20M	38
4.5.2	Results on H&M Fashion	39
4.5.3	Results on Amazon Reviews 2023	40
4.5.4	Results Summary	40
5	Fine-Tuning Multimodal Encoders	43
5.1	Fine-Tuning on Interaction Data	43
5.2	Experimental Setup	44
5.2.1	Tuning Methods	45
5.2.2	Fusion Strategies	45
5.2.3	Training Objectives	46
5.3	Implementation Details	47
5.4	Results	47
5.4.1	Results by Tuning Method	48
5.4.2	Results by Fusion Strategy	48
5.4.3	Results by Training Objective	52
5.4.4	Best-Performing Configuration	54
5.4.5	Summary of Results	54

6	Analysis of Multimodal Embedding Spaces	57
6.1	Global Embedding Space Structure	57
6.2	Local Neighborhood Case Examples	59
6.3	Attention Preservation in Fine-Tuned Encoders	61
6.3.1	Algorithm for Analyzing Attention Behavior	61
6.3.2	Comparison of Attention Patterns	62
6.4	Summary of Findings	62
7	Discussion	65
7.1	Summary of Main Findings	65
7.2	REEF and Evaluation Methodology	65
7.3	Evaluation of Zero-Shot and Fine-Tuning Results	65
7.4	Representation Analysis	66
7.5	Limitations and Practical Implications	67
8	Conclusion	69
A	Sample REEF Configuration	81

List of Figures

1.1	Conceptual illustration of a recommender system.	3
1.2	Illustration of the cold-start problem.	6
3.1	End-to-end pipeline of REEF.	22
3.2	Overview of the five evaluation scenarios.	27
5.1	Overview of fine-tuning on interaction data.	44
6.1	UMAP projection of item embeddings before and after fine-tuning.	58
6.2	Top-ten nearest neighbors on H&M Fashion: frozen vs. fine-tuned.	59
6.3	Top-ten nearest neighbors on Amazon Reviews 2023: frozen vs. fine-tuned. . . .	60
6.4	Top-ten nearest neighbors on MovieLens 20M: frozen vs. fine-tuned.	60
6.5	Attention patterns on H&M items: frozen vs. fine-tuned.	63
6.6	Attention patterns on MovieLens items: frozen vs. fine-tuned.	64

List of Tables

4.1	Encoder configurations evaluated in the zero-shot regime.	35
4.2	Zero-shot results on MovieLens 20M.	39
4.3	Zero-shot results on H&M Fashion.	40
4.4	Zero-shot results on Amazon Reviews 2023.	40
5.1	Tuning-method comparison across the five evaluation scenarios.	49
5.2	Fusion-strategy comparison across the five evaluation scenarios.	51
5.3	Training-objective comparison across the five evaluation scenarios.	53
5.4	Final BGE-M3 + SigLIP2 fine-tuned configuration vs. frozen baseline.	54

List of code listings

3.1	Interaction row schema.	23
3.2	Dataset and data bundle.	24
3.3	Task protocol.	25
3.4	Training exclusions and task view.	25
3.5	Query envelope.	26
3.6	Run context carried through the pipeline.	27
3.7	Backbone protocol.	28
3.8	Trainer protocol.	28
3.9	Embedding bundle.	29
3.10	Downstream model protocol.	30
3.11	Evaluator protocol.	30
A.1	Sample REEF configuration.	81

I would like to thank my supervisor, doc. Ing. Pavel Kordík, Ph.D., for his guidance, constructive feedback, and patience throughout the preparation of this thesis.

I also thank my family and my girlfriend for their love, encouragement, and support in all the other aspects of life, which made this journey much easier.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Praze on September 10, 2026

Abstract

Modern recommender systems rely heavily on how users and items are represented. The rise of foundation models has introduced large pretrained encoders that can generate representations directly from item content, but their integration into recommendation pipelines is difficult to evaluate. Performance differences between encoders are often confounded by variations in downstream models, training budgets, and other pipeline components, making it unclear which backbone is truly better. We refer to this as the *backbone isolation problem*. This thesis addresses the problem with three main contributions. First, we introduce the Recommender Embedding Evaluation Framework (REEF), which fixes all components of the pipeline except the backbone, allowing for fairer comparison. Second, we use REEF to evaluate existing pretrained text, vision, and multimodal encoders. Third, we perform a fine-tuning study of multimodal backbones, to examine how interaction data can be incorporated into pretrained item encoders. Part of this analysis is also an examination of the embedding spaces of the fine-tuned encoders to understand how adaptation to interaction data changes the structure of the learned representations.

Keywords recommender systems, foundation models, representation evaluation, multimodal learning, transfer learning, benchmark

Abstrakt

Moderní doporučovací systémy do značné míry závisejí na tom, jak jsou reprezentováni uživatelé a položky. Nástup velkých modelů přinesl předtrénované enkodéry, které dokážou vytvářet reprezentace přímo z obsahu položek. Jejich začlenění do doporučovacích algoritmů je však obtížné spolehlivě vyhodnotit. Rozdíly ve výkonu jednotlivých enkodérů jsou často ovlivněny rozdíly v navazujících modelech, trénovacím rozpočtu a dalších částech algoritmu, a není tedy zřejmé, který model je skutečně lepší. Tento problém v práci označujeme jako *problém izolace modelu*. Práce tento problém řeší pomocí tří hlavních příspěvků. Nejprve představujeme Recommender Embedding Evaluation Framework (REEF), který fixuje všechny části algoritmu kromě modelu, a umožňuje tak objektivnější srovnání. Dále pomocí REEF vyhodnocujeme existující předtrénované textové, obrazové a multimodální enkodéry. Nakonec se zaměřujeme na dotrénování multimodálních modelů a zkoumáme, jak lze předtrénované enkodéry přizpůsobit pomocí interakčních dat. Součástí této analýzy je také rozbor embeddingových prostorů dotrénovaných enkodérů, který ukazuje, jak adaptace na interakční data mění strukturu naučených reprezentací.

Klíčová slova doporučovací systémy, základní modely, hodnocení reprezentací, multimodální učení, učení přenosem, testovací aplikace

List of Abbreviations

BPR	Bayesian Personalized Ranking
CNN	Convolutional Neural Network
DCG	Discounted Cumulative Gain
FT	Fine-Tuning
GELU	Gaussian Error Linear Unit
GPU	Graphics Processing Unit
GTE	General Text Embeddings
HR	Hit Rate
ID	Identifier
IDCG	Ideal Discounted Cumulative Gain
InfoNCE	Information Noise-Contrastive Estimation
LLM	Large Language Model
LoRA	Low-Rank Adaptation
MAE	Mean Absolute Error
MLP	Multi-Layer Perceptron
MRR	Mean Reciprocal Rank
MSE	Mean Squared Error
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
PMI	Pointwise mutual information
REEF	Recommender Embedding Evaluation Framework
UMAP	Uniform Manifold Approximation and Projection
VRAM	Video Random-Access Memory

Introduction

Recommender systems are one of the most largely deployed machine learning algorithms. Online stores, streaming platforms, social networks, and news platforms all rely on them to help users find relevant items in catalogs [1]. Historically, recommender systems learned representations separately for each task, often from collaborative filtering signals derived from the user-item interaction matrix [2]. Modern recommender systems depend strongly on how users and items are represented, since these representations determine how preferences, item similarities, and user-item relevance are modeled [3, 4].

The rise of foundation models has introduced a new paradigm for representation learning. Large encoders pretrained on broad, heterogeneous data with self-supervised or weakly supervised objectives are now publicly available for many modalities, like text [5, 6], vision [7], and aligned text-image pairs [8, 9]. These encoders provide general backbones that can represent catalog items through their textual and visual content without requiring specific supervision. They are attractive for recommendation systems as they can easily encode previously unseen items based on their characteristics, are interchangeable as drop-in feature extractors, and bring in rich semantic structure learned from data far larger than any single recommendation catalog.

Integrating foundation backbones into recommender systems, however, brings several evaluation challenges. In production systems, foundation models are often used as representation extractors for different downstream tasks. This raises the question of how the foundation model part of the pipeline should be evaluated. An observed difference in performance may come from the backbone itself, from its embedding dimensionality, from the downstream architecture selected for each backbone, from the training budget, or from any number of confounded hyperparameters. Existing evaluation practice rarely separates these factors. Backbones are usually assessed end-to-end, inside the recommendation pipeline. As a result, it is difficult to determine which backbone is better suited for a given task. We refer to this as the *backbone isolation problem*.

This thesis addresses the backbone isolation problem with three contributions. First, we introduce Recommender Embedding Evaluation Framework (REEF), a protocol based benchmarking framework for evaluating foundation model representations in recommendation, under fixed downstream conditions. Within a single REEF run, multiple backbones are passed through the same data loader, the same task, the same downstream ranker, and the same evaluator, so that the only variable across configurations is the backbone itself. Second, we use REEF to evaluate existing pretrained encoders spanning text, vision, and multimodal modalities on three public datasets, across five recommendation scenarios. Third, we create a study on multimodal backbones fine-tuning, where we experiment with bringing interaction information to the backbone. We evaluate fusion strategies, tuning methods, and training objectives, bringing interaction data to the model weights. We extend this study with an analysis of how the embedding space changes with fine-tuning.

Theoretical Background

This chapter introduces the main concepts used later in the thesis. It first defines recommender systems and standard offline ranking metrics. We then describe foundation encoder models, common adaptation strategies, representation learning, and learning objectives used to adapt representations from user-item interaction data. Lastly we cover common visualization algorithms.

1.1 Recommender Systems

Recommender systems are machine learning algorithms that help users find relevant items in large catalogs. They are used in online stores, streaming platforms, social networks, news platforms, and other environments where the number of available items is much larger than what a user can inspect manually. Their purpose is to estimate which items are likely to be useful, interesting, or engaging for a given user and to rank candidate items accordingly. Figure 1.1 shows the core idea: a recommender system acts as an intermediary between a user and a catalog of candidate items. While recommendation can be formulated in several ways, including user-to-user recommendation, this thesis focuses on the item-to-user setting [10, 11].

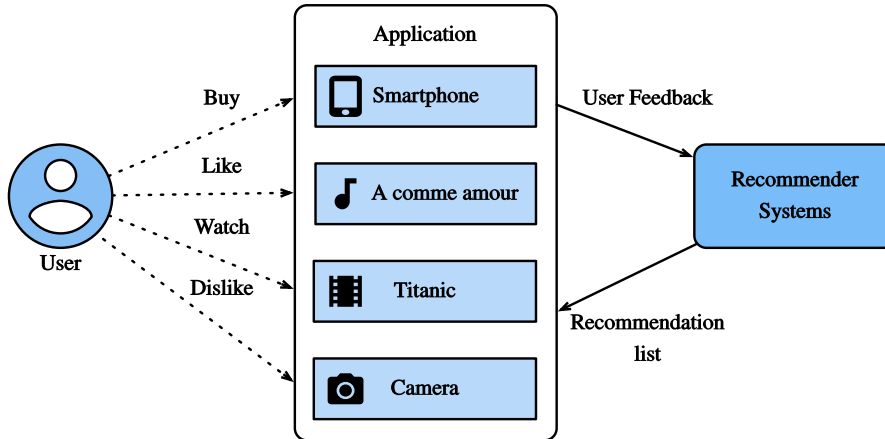


Figure 1.1 Conceptual illustration of a recommender system. A user is associated with a set of candidate items, and the recommender system selects items that are likely to be relevant [1].

Although recommendation appears in many forms, the underlying problem is usually similar. Given observed user behavior, such as clicks, purchases, ratings, or other interaction events, the system predicts future preferences. This is challenging since recommendation data is usually sparse, incomplete, and biased. Most users interact with only a small fraction of the catalog, many items have little to no interaction history, and observed interactions reflect both user preferences and what the system previously exposed to the user [12, 13].

From a machine learning perspective, recommendation is commonly formulated as a ranking problem over users and items. Let $\mathcal{U}$ denote the set of users, $\mathcal{I}$ the set of items, and let $R \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ be the user-item interaction matrix, where rows represent users and columns represent items. The entry R_{ui} records an observed signal of preference, such as a rating or an implicit interaction event. The goal is to learn a scoring function

$$f_\theta : \mathcal{U} \times \mathcal{I} \rightarrow \mathbb{R},$$

which assigns a relevance score to item i for user u . Items with higher scores are ranked higher and are more likely to be recommended.

1.1.1 Content-Based and Collaborative Recommendation

Recommender systems are often described through two approaches: collaborative filtering and content-based recommendation.

Collaborative filtering learns from user behavior. Its central assumption is that users who behaved similarly in the past are likely to prefer similar items in the future, and that items consumed by similar users are likely to be related. This approach does not require any knowledge of item content. Matrix factorization is one of the best-known collaborative filtering methods and represents users and items as latent vectors whose inner product predicts preference [14].

Content-based recommendation instead relies on item attributes, such as titles, descriptions, categories, images, or structured metadata. The system recommends items whose content is similar to items the user previously interacted with. This approach is useful when item content is informative and when interaction history is limited [15].

In practice, many recommender systems are *hybrid*, they combine interaction signals with item content. Collaborative signals are often strong when sufficient historical data are available, while content signals help with new, rare, or sparsely observed items [10].

1.1.2 Evaluation of Recommender Systems

Evaluating recommender systems means measuring how well the model performs. Given a user u , the system produces an ordered list of candidate items, and this ranking is compared against a set of relevant items derived from held-out interactions. The goal is not only to retrieve relevant items, but also to place them as high as possible in the recommendation list [16].

Formally, let π_u denote the ranked list of items for user u , and let G_u denote the set of relevant ground-truth items. Evaluation metrics quantify the agreement between π_u and G_u , typically focusing on the top- K positions, because users rarely inspect the full candidate list. This is commonly written as $\text{Metric}@K$, meaning that performance is measured within the first K ranked items. Different metrics emphasize different aspects of ranking quality.

1.1.2.1 Precision and Recall

Precision@K measures the proportion of the top- K recommended items that are relevant. For example, if four out of ten recommended items are relevant, then $\text{Precision}@10$ is 0.4.

Recall@K measures the proportion of relevant items that appear in the top- K results. If a user has five relevant held-out items and the recommender retrieves three of them in the top ten,

then Recall@10 is 3/5. Formally,

$$\text{Recall@}K(u) = \frac{|G_u \cap \pi_u^K|}{|G_u|}.$$

Precision reflects the quality of the recommended list itself, whereas recall reflects how much of the relevant set has been recovered. Precision is useful when avoiding irrelevant recommendations is important, while recall is useful when covering as many relevant items as possible is the priority.

1.1.2.2 Hit Rate

HitRate@K measures whether at least one relevant item appears in the top- K recommendation list:

$$\text{HR@}K(u) = \mathbb{I}[G_u \cap \pi_u^K \neq \emptyset].$$

The metric is useful in scenarios where a single successful recommendation is sufficient, for example often in session-based recommendation or first-click settings.

1.1.2.3 Mean Reciprocal Rank

Mean Reciprocal Rank (MRR) emphasizes how early the first relevant item appears in ranking. For a user u , let r_u be the rank of the first relevant item. Then

$$\text{MRR}(u) = \frac{1}{r_u},$$

it is therefore useful in settings where users are likely to interact with the first satisfactory result.

1.1.2.4 Discounted Cumulative Gain

Recommendation quality depends not only on whether relevant items are retrieved, but also on where they appear in the ranking. *Discounted Cumulative Gain* (DCG) addresses this by assigning higher value to relevant items that appear earlier in the list. For a ranking truncated at position K , DCG is defined as

$$\text{DCG@}K(u) = \sum_{r=1}^K \frac{\mathbb{I}[\pi_u(r) \in G_u]}{\log_2(r+1)},$$

where $\pi_u(r)$ denotes the item at rank r , and $\mathbb{I}[\cdot]$ is an indicator function equal to 1 if the condition inside the brackets is true and 0 otherwise.

The logarithmic discount reduces the contribution of relevant items that appear lower in the ranking. As a result, DCG rewards systems that not only retrieve relevant items, but place them near the top of the recommendation list.

Since the value of DCG depends on the number of relevant items available for a given user, it is commonly normalized by the score of the ideal ranking. This yields *Normalized Discounted Cumulative Gain* (NDCG):

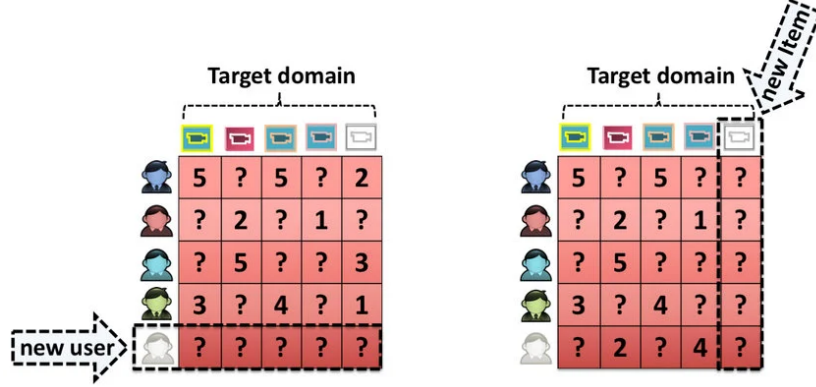
$$\text{NDCG@}K(u) = \frac{\text{DCG@}K(u)}{\text{IDCG@}K(u)},$$

where $\text{IDCG@}K(u)$ is the maximum possible DCG for user u , obtained by ranking all relevant items as highly as possible.

NDCG is particularly useful in recommendation because it captures both relevance and rank position [17].

1.1.3 Challenges in Recommendation

Several challenges appear repeatedly in recommendation settings. The most widely studied is the *cold-start problem*, which occurs when the system must recommend a new item or serve a new user with little or no historical interaction data. In such cases, purely interaction-based methods struggle because the model has little behavioral evidence for that user/item. Figure 1.2 illustrates this problem.



■ **Figure 1.2** Illustration of the cold-start problem in recommender systems, showing the new-user problem and the new-item problem [18, Fig. 2].

A related issue is *data sparsity*. In most real systems, each user interacts with only a tiny fraction of the available catalog. This makes it difficult to estimate preferences, especially for infrequent users and long-tail items.

Recommendation data are also affected by bias. *Popularity bias* means that popular items receive more interactions and are therefore recommended more frequently. *Exposure bias* means that observed interactions depend not only on what a user likes, but also on what the system previously chose to show. Thus, missing interactions cannot always be interpreted as negative preference [13, 19].

Finally, recommendation models can face *domain shift*. A model trained in one domain or dataset may not transfer well to another because users, items, interaction patterns, and catalog structure differ. This issue is even more relevant when studying whether learned item representations transfer across recommendation tasks [20].

1.2 Foundation Models

Foundation models are large models trained on broad and diverse datasets, typically using self-supervised or weakly supervised objectives. Rather than being optimized for a single task, they are designed to learn general representations that can be reused across many downstream applications [21].

Formally, let $x \sim \mathcal{D}_{\text{pretrain}}$ denote samples from a broad pretraining distribution. A foundation model with parameters ϕ is trained by minimizing

$$\phi^* = \arg \min_{\phi} \mathbb{E}_{x \sim \mathcal{D}_{\text{pretrain}}} [\mathcal{L}_{\text{pretrain}}(x; \phi)].$$

The exact form of $\mathcal{L}_{\text{pretrain}}$ depends on the model family.

We focus on *encoder* foundation models, which transform raw input data into fixed vector representations. These models are trained on large-scale datasets and learn to encode semantic properties of the input into a dense embedding space.

Formally, an encoder h with parameters ϕ is a function

$$h_\phi : \mathcal{X} \rightarrow \mathbb{R}^d,$$

where $x \in \mathcal{X}$ is an input instance and $h_\phi(x)$ is its vector representation. The input space $\mathcal{X}$ depends on the modality. Different encoder architectures are used for different modalities, but the shared purpose is to produce representations that can be reused for different tasks.

1.2.1 Adapting Foundation Models

Foundation models are trained on broad datasets, so their representations are not directly optimized for any specific task. To make them suitable for a target task, the encoder can be further adapted using task-specific data. This adaptation process is commonly referred to as fine-tuning. The fine-tuning strategy determines how much of the pretrained model is updated and how strongly the original representations are modified.

1.2.1.1 Full Fine-Tuning

In full fine-tuning, all parameters of the pretrained encoder are updated on the target task. Let h_ϕ denote the encoder and g_ψ a task-specific prediction head. The objective is:

$$(\phi^*, \psi^*) = \arg \min_{\phi, \psi} \mathbb{E}_{(x, y)} [\mathcal{L}_{\text{task}}(y, g_\psi(h_\phi(x)))].$$

This approach provides maximum flexibility because the entire encoder can adapt to the target data distribution. It can be especially effective when the target dataset is sufficiently large and the target task differs from the pretraining setting, since deeper layers of neural networks tend to become more task-specific and may require adaptation when transferred to a new domain [22].

However, full fine-tuning is also the most expensive adaptation strategy. Since all encoder parameters are updated, it requires substantial memory and computation, especially for large foundation models [23]. It also increases the risk of overfitting when the target dataset is small relative to the number of updated parameters [22]. In addition, updating the entire encoder can lead to catastrophic forgetting, where useful information learned during pretraining is partially overwritten during adaptation [24].

1.2.1.2 Low-Rank Adaptation

Low-Rank Adaptation (LoRA) modifies selected weight matrices using trainable low-rank updates instead of updating the full pretrained weights. For a pretrained weight matrix W , the adapted weights are given by:

$$W' = W + BA,$$

where A and B are trainable low-rank matrices, while W remains fixed.

This approach is based on the observation that task-specific updates can often be represented with low-rank matrices [23]. Compared with full fine-tuning, LoRA is more parameter-efficient because only the low-rank update matrices are trained, while the original pretrained weights remain frozen. This reduces the number of trainable parameters and lowers memory requirements during adaptation [23]. Since the pretrained weights are preserved, LoRA also reduces the risk of overwriting pretrained knowledge and allows multiple task-specific adaptations to be stored compactly as separate low-rank updates [23].

The main limitation of LoRA is that it restricts the form of the parameter update. Because adaptation is constrained to a low-rank subspace, LoRA may be less expressive than full fine-tuning when the target task requires substantial changes to the pretrained representation. Its performance also depends on design choices such as the selected layers, the rank of the update matrices, and the scaling of the low-rank update [23].

1.2.1.3 Adapter-Based Tuning

Adapter-based tuning introduces small trainable modules into the pretrained network while keeping the original backbone weights fixed. A typical adapter uses a bottleneck architecture: the representation is projected to a lower-dimensional space, transformed, and then projected back to the original dimensionality [25].

Unlike LoRA, which modifies existing weight matrices through low-rank updates, adapters add new task-specific layers to the model. This makes adapter-based tuning parameter-efficient while preserving the pretrained backbone [25]. Since only the adapter modules are trained, the method reduces the number of trainable parameters and the amount of task-specific storage compared with full fine-tuning. It also makes it possible to maintain several task-specific adaptations for the same backbone by storing only the corresponding adapter modules [25].

1.2.2 Meta-Learning

A different way to adapt a model to new tasks is meta-learning, often described as “learning to learn”. Instead of training a model on a single task, meta-learning trains across a distribution of related tasks, with the goal of producing a model that can be quickly adapted to a new task from a small number of examples. The model is exposed to many tasks during training, and at evaluation time, it receives a small *support set* of examples from a previously unseen task and must perform well on a separate *query set* from the same task.

Meta-learning methods fall into two broad families. Optimization-based methods, such as Model-Agnostic Meta-Learning (MAML) [26], learn a model initialization from which a small number of gradient steps on the support set yields a strong task-specific model. Metric-based methods, such as Prototypical Networks [27], learn an embedding space in which simple distance comparisons against the support set solve the new task without any inner-loop gradient updates.

In recommendation, meta-learning is most commonly used to address cold-start. Each user or each item can be treated as a task, with the small set of available interactions serving as the support set, and the meta-learner adapts the recommender to the new user or item from those few examples.

1.3 Representation Learning for Recommendation

Encoder models produce general-purpose representations that are not by themselves aligned with the ranking objectives. While these encoders effectively map raw inputs to feature spaces, items in modern datasets are often multimodal, these inputs must somehow be combined into a single item representation. That is the task of multimodal representation learning.

1.3.1 Multimodal Representation Learning

Products may include a title, description, image, category, and price, a movie may include a poster, cast list, and a trailer. Text is well suited for expressing category, function, and abstract semantics, whereas images capture visual properties such as appearance, style, and structure, so the signals are largely complementary [28]. Because recommenders expect a single item vector, the modality-specific embeddings produced by separate encoders must be fused into one representation $\mathbf{z}_i$.

We simplify the discussion to two modality cases, these are text and images. Let $h_t(t_i) \in \mathbb{R}^{d_t}$ denote the text embedding and $h_v(v_i) \in \mathbb{R}^{d_v}$ the image embedding. A fusion function F maps the pair $(h_t(t_i), h_v(v_i))$ to a joint representation $\mathbf{z}_i$, optionally followed by a projection into the embedding space used by the recommender.

1.3.1.1 Averaging

A simple parameter-free strategy is *averaging*, which assumes that both modality encoders produce vectors of the same dimensionality $d_t = d_v = d$ and combines them by an element-wise mean:

$$\mathbf{z}_i = \frac{1}{2} (h_t(t_i) + h_v(v_i)).$$

Averaging introduces no trainable parameters and is particularly well suited to encoders like CLIP, whose text and image branches are explicitly trained to produce embeddings in a shared space [8]. Because the operation is linear, it preserves the geometric structure of the original embedding space, so neighborhood relations from each modality are largely retained in the fused representation.

1.3.1.2 Concatenation

Another simple approach is *concatenation*, which stacks the modality representations into a single vector:

$$\mathbf{z}_i = [h_t(t_i); h_v(v_i)],$$

where $[\cdot; \cdot]$ denotes the concatenation operation. This method preserves all modality-specific information in its raw form, resulting in a joint representation $\mathbf{z}_i \in \mathbb{R}^{d_t+d_v}$.

1.3.1.3 Weighted Sum with Learned Projections

The *weighted sum* strategy first projects both modalities into a shared space and then combines them linearly:

$$\mathbf{z}_i = w_t \cdot P_t h_t(t_i) + w_v \cdot P_v h_v(v_i)$$

where $P_t \in \mathbb{R}^{d \times d_t}$ and $P_v \in \mathbb{R}^{d \times d_v}$ are modality-specific projections, w_t and w_v are scalar weights satisfying $w_t + w_v = 1$. This approach enforces a shared representation space and introduces direct interaction between modalities through a weighted combination.

This method assumes that the importance of text and image information is fixed across items. For example, if w_t is larger than w_v , the fused representation will generally rely more on textual information for every item.

1.3.1.4 Multi-Head Attention Fusion

The *attention-based* strategy provides a more flexible way to combine modalities. As in the weighted-sum approach, both modality embeddings are first projected into a shared d -dimensional space. The projected text and image embeddings are treated as a short sequence of two tokens:

$$\begin{aligned} \mathbf{s} &= [P_t h_t(t_i); P_v h_v(v_i)] \in \mathbb{R}^{2 \times d}, \\ \mathbf{s}' &= \text{LayerNorm}(\mathbf{s} + \text{MHA}(\mathbf{s})), \\ \mathbf{z}_i &= \frac{1}{2} \sum_{k=1}^2 \mathbf{s}'_k. \end{aligned}$$

Here, the first token corresponds to the projected text embedding and the second token corresponds to the projected image embedding. The multi-head self-attention layer allows each modality to attend to the other before the final item representation is computed. The residual connection preserves the original projected representations, while layer normalization stabilizes the transformed sequence.

Unlike the weighted sum, attention fusion can make the interaction between modalities depend on the item itself. For one item, the model may rely more strongly on the visual token, while for another item it may use the text token more heavily [29].

1.3.1.5 Gated Fusion

The *gated fusion* strategy combines modalities using a learned sigmoid gate applied element-wise:

$$\begin{aligned}\mathbf{g} &= \sigma(W_g [P_t h_t(t_i); P_v h_v(v_i)]), \\ \mathbf{z}_i &= P(\mathbf{g} \odot P_t h_t(t_i) + (1 - \mathbf{g}) \odot P_v h_v(v_i)).\end{aligned}$$

This makes gated fusion more flexible than a weighted sum. Instead of assigning one global weight to text and one global weight to images, the model can decide that some dimensions should rely more on textual information while others should rely more on visual information. At the same time, gated fusion is usually lighter than attention-based fusion, because it does not compute pairwise attention interactions between tokens. This makes it a practical compromise between simplicity and adaptivity [30].

1.4 Learning Objectives from Interaction Data

To adapt a model for recommendation, we must define a learning objective. A learning objective tells the model which predictions are encouraged, which are penalized, and how the model should adapt its parameters to the recommendation task. In recommendation, this usually means assigning higher scores to items a user interacted with than to items that are unobserved, irrelevant or sampled as negatives.

Learning objectives in recommendation can be broadly categorized based on how they use interaction signals. These include pointwise, pairwise, and listwise families [31].

Pointwise objectives treat each user–item interaction independently and predict a target value, such as a binary relevance label or a rating. These methods are simple and easy to optimize, but they do not directly model the relative order between items.

Pairwise objectives learn by comparing pairs of items for the same user. Given a positive item and a sampled negative item, the objective encourages the model to assign a higher score to the positive item than to the negative one. This makes the objective more closely aligned with ranking, although each training step still considers only one positive–negative comparison.

Listwise objectives optimize over a set or list of candidate items for the same user. Instead of comparing individual items independently or in pairs, they consider the relative ordering of multiple items at once. This makes them more directly connected to ranking metrics, since the model is trained to place relevant items higher within an entire candidate list. Below, we describe several losses used in recommendation.

1.4.1 Bayesian Personalized Ranking

Bayesian Personalized Ranking (BPR) is a pairwise ranking objective designed for implicit-feedback recommendation. For a user u , a positive item i , and a sampled negative item j , the objective is

$$\mathcal{L}_{\text{BPR}} = -\frac{1}{|\mathcal{B}|} \sum_{(u,i,j) \in \mathcal{B}} \log \sigma(\mathbf{e}_i^\top \bar{\mathbf{e}}_u - \mathbf{e}_j^\top \bar{\mathbf{e}}_u).$$

The loss encourages the score of the observed item to exceed that of the unobserved one. Compared to contrastive objectives, BPR focuses on local ranking consistency between sampled item pairs rather than classification among a larger candidate set. Its simplicity and effectiveness make it a standard objective for implicit-feedback recommendation [2].

1.4.2 InfoNCE Loss

InfoNCE is a widely used contrastive objective introduced in contrastive predictive coding. It compares one positive item against multiple negatives at the same time. For a user representation $\bar{\mathbf{e}}_u$, a positive item embedding $\mathbf{e}_i$, and a set of negative items $\mathcal{N}_u$, the loss is

$$\mathcal{L}_{\text{InfoNCE}} = -\frac{1}{|\mathcal{B}|} \sum_{(u,i) \in \mathcal{B}} \log \frac{\exp(\mathbf{e}_i^\top \bar{\mathbf{e}}_u / \tau)}{\exp(\mathbf{e}_i^\top \bar{\mathbf{e}}_u / \tau) + \sum_{j \in \mathcal{N}_u} \exp(\mathbf{e}_j^\top \bar{\mathbf{e}}_u / \tau)}.$$

The objective encourages the model to assign higher similarity to the positive item than to all sampled negatives. The temperature parameter τ controls how sharply similarities are contrasted. In practice, InfoNCE can be interpreted as a classification problem in which the model must identify the correct item among a set of candidates [32].

1.4.3 ELSA Loss

ELSA is an interaction-driven objective based on a linear shallow autoencoder defined over the user–item interaction matrix. Rather than matching item embeddings to a separately constructed target similarity matrix, ELSA uses the embeddings to parameterize a low-rank item–item operator that reconstructs user interaction vectors. Let $R \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ be the user–item interaction matrix, where R_{ui} indicates the interaction between user u and item i . Let $E \in \mathbb{R}^{|\mathcal{I}| \times d}$ denote the item embedding matrix, where the i -th row $\mathbf{e}_i \in \mathbb{R}^d$ is the embedding of item i . ELSA defines an item–item weight matrix by taking inner products of item embeddings and zeroing out the diagonal:

$$W = EE^\top - \text{diagMat}(EE^\top),$$

where $\text{diagMat}(\cdot)$ returns a matrix containing only the diagonal entries of its argument, with all off-diagonal entries set to zero. Removing the diagonal ensures that each item is reconstructed only from other items rather than from itself, which would otherwise admit a trivial identity solution. The predicted interaction matrix is then computed as:

$$R_{\text{pred}} = RW.$$

The loss compares normalized original and reconstructed interaction vectors:

$$\mathcal{L}_{\text{ELSA}} = \|\text{norm}(R) - \text{norm}(R_{\text{pred}})\|_F^2.$$

This objective encourages item embeddings to encode collaborative relationships that allow user interaction histories to be reconstructed through similar items. In beeFormer, the item embedding matrix is not optimized directly. Instead, it is produced by an encoder:

$$E = g(T; \theta),$$

where T denotes item text inputs and θ are the encoder parameters. The ELSA loss is then backpropagated through the encoder, allowing pretrained semantic representations to adapt to the collaborative structure present in the interaction data [33, 34].

1.4.4 Interaction-Derived Similarity Matching

The reconstruction view of ELSA can be interpreted as implicitly learning an item–item similarity structure induced by user interactions. This motivates a broader class of objectives that operate directly on item–item similarity matrices, which we refer to as interaction-derived similarity matching.

In this formulation, the item embedding matrix $E \in \mathbb{R}^{n \times d}$ induces a predicted similarity matrix

$$\hat{S} = EE^\top, \quad \hat{S}_{ij} = \langle \mathbf{e}_i, \mathbf{e}_j \rangle.$$

The model is trained to align this predicted similarity with a target similarity matrix constructed from interaction data:

$$\mathcal{L}_{\text{sim}} = D(\hat{S}_B, S_B^{\text{target}}),$$

where B denotes a subset of items, and $D(\cdot, \cdot)$ is a matrix discrepancy measure such as mean squared error or a divergence between normalized rows.

This formulation can be viewed as a direct relaxation of ELSA. While ELSA learns a similarity operator implicitly through reconstruction of interaction vectors, similarity matching objectives supervise the similarity structure explicitly by comparing predicted and target item–item relationships. Both approaches rely on the same underlying signal: item co-occurrence patterns derived from the interaction matrix R .

The target similarity matrix S^{target} is constructed from the same user–item interaction matrix R . Let $R_{:,i} \in \mathbb{R}^{|\mathcal{U}|}$ denote the interaction vector of item i .

One option is Jaccard similarity:

$$S_{ij}^{\text{jac}} = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}, \quad U_i = \{u \mid R_{ui} > 0\},$$

which considers only the overlap of user sets and ignores interaction counts.

Pointwise mutual information (PMI) compares observed co-occurrence against what would be expected under independence:

$$S_{ij}^{\text{pmi}} = \log \frac{P(i, j)}{P(i)P(j)}.$$

In practice, PMI is often clipped at zero to suppress weak or noisy associations.

These constructions define different notions of item relatedness. Jaccard focuses on binary overlap, PMI highlights unexpectedly strong associations.

1.5 Representation Analysis Methods

Since we also include analyses of the learned representations, we need to briefly introduce several algorithms used to understand the structure of the embedding space.

1.5.1 Cosine Similarity

The cosine similarity between two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ is the inner product of their L_2 -normalised forms:

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}^\top \mathbf{v}}{\|\mathbf{u}\|_2 \|\mathbf{v}\|_2}.$$

Because it depends only on the angle between the vectors, cosine similarity is unaffected by their magnitudes. This makes it a standard choice for comparing high-dimensional representations.

1.5.2 Vision Transformer Attention

A Vision Transformer (ViT) splits an input image into a regular grid of patches, embeds each patch into a token, and processes the resulting sequence with a stack of L transformer layers, each containing H self-attention heads [29, 35]. At every layer the model produces row-stochastic attention matrices $\mathbf{A}_h^{(l)} \in \mathbb{R}^{N \times N}$, where N is the number of patch tokens (excluding the [CLS] token used for sequence-level pooling). Entry $[\mathbf{A}_h^{(l)}]_{ij}$ records how much patch i attends to patch j in head h of layer l .

1.5.3 Uniform Manifold Approximation and Projection

Uniform Manifold Approximation and Projection (UMAP) is a non-linear dimensionality reduction method used to project high-dimensional data into a low-dimensional space [36]. UMAP provides a way to visualize high-dimensional embeddings, keeping local neighbourhoods relationships as much as possible.

UMAP first builds a neighbourhood graph in the original embedding space, where each point is connected to its nearest neighbours. It then finds a low-dimensional projection that preserves this neighbourhood structure. Points that are close in the original space should therefore also appear close in the projection, while groups of related points may form visible clusters.

1.5.4 K-Means Clustering

K-means is a clustering algorithm used to partition data points into k groups based on their similarity [37]. Given a set of inputs, the algorithm assigns each input to the nearest cluster centroid and then updates each centroid as the mean of the assigned points. This process is repeated until the assignments converges. The objective minimized by k-means is the within-cluster sum of squared distances:

$$\sum_{c=1}^k \sum_{\mathbf{x}_i \in C_c} \|\mathbf{x}_i - \boldsymbol{\mu}_c\|_2^2,$$

where C_c is the set of points assigned to cluster c and $\boldsymbol{\mu}_c$ is the corresponding centroid.

Related Work

This chapter reviews existing work on representation learning and foundation models for recommender systems. It focuses on how the literature has moved from ID-based collaborative filtering toward content-aware, multimodal, transferable, and generative recommendation. The chapter first discusses foundation models for item representation learning. For completeness, we also review graph-based recommendation, language-model and generative approaches to recommendation. The chapter concludes with datasets and evaluation frameworks, with emphasis on how well existing benchmarks support representation evaluation.

2.1 Foundation-Model Encoders for Recommendation

Traditional collaborative filtering learns embeddings from interaction data and treats users and items as identifiers. Content-based recommendation instead derives representations from item attributes such as text, images, audio, or metadata. Modern industrial systems often combine both views, they learn representations from behavioral sequences, catalog structure, item content, context, and downstream task feedback.

This is visible in large-scale production systems. Netflix describes recommendation as a collection of specialized deep learning models for different personalization tasks, where learned user and item representations are reused across applications [38]. More recently, Netflix introduced a personalization foundation model that produces user and item embeddings from behavioral sequences and passes these embeddings to downstream models, such as ranking, search, and homepage personalization [39]. Spotify follows a similar representation centric direction in music recommendation, they learn user and session embeddings from listening sequences and contextual information [40].

This shows that text and image encoders are only one part of a larger representation-learning landscape. The focus of this thesis is narrower, we study foundation item encoders, specifically text, vision, and multimodal backbones. These encoders are convenient as they can represent new items directly from attributes, without requiring interaction history.

2.1.1 Text Encoders

Recent advances in natural language processing have produced embedding models designed for semantic similarity and dense retrieval. Sentence-BERT introduced a siamese sentence-encoder architecture that made BERT-style models practical for similarity search [41]. Dense Passage Retrieval then showed that dual encoders trained contrastively can support large-scale neural

retrieval [42]. Later models such as Contriever, E5, BGE, BGE-M3, and GTE improved training data, objectives, multilinguality, and retrieval robustness [43, 44, 6, 45, 46].

In recommendation text-based item encoders first became prominent in news settings. News articles are short lived, frequently published, and often have little interaction history, which makes cold-start the central problem [47]. Early neural news recommenders therefore focused on learning stronger article representations from titles, categories, and entities. DKN introduced a knowledge-aware CNN architecture that enriches news titles with entity information from a knowledge graph [48]. NAML represented news through multiple textual and categorical views, using attention to combine titles, bodies, and metadata [49]. NRMS moved from convolutional encoders to multi-head self-attention, reflecting the broader shift from local text features toward contextual representations [50].

The next step was to replace recommendation-specific text encoders with pretrained language models. This follows the same motivation as in other NLP applications. Models pretrained on large corpora contain linguistic and semantic knowledge that task specific encoders trained on recommendation may not learn. In news recommendation, PLM-based models showed that pretrained language encoders can improve article understanding, but they also exposed a practical problem. Repeatedly encoding the same articles during training and serving is computationally expensive [51]. This pushed later work toward efficient training, caching, distillation, and domain adaptation.

More recent work treats text as a transferable item representation rather than a content feature. ZESRec uses item descriptions as continuous item indices for zero-shot sequential recommendation, addressing the fact that item IDs do not transfer across domains [52]. UniSRec extends this idea with multi-domain contrastive pretraining, aiming to learn universal sequence representations from item descriptions [53]. VQ-Rec adds vector quantization between textual descriptions and item representations, reducing direct dependence on textual similarity and improving robustness under domain shift [54]. Recformer similarly textualizes item attributes and models user histories as sequences of item descriptions [55].

This line of work also reveals a limitation of directly importing semantic encoders into recommendation. General purpose sentence encoders are usually trained to capture semantic similarity, but semantic similarity and interaction similarity are not identical. Two products may be textually similar but appeal to different users, or textually different but frequently co-consumed. BeeFormer addresses this by training item encoders with interaction supervision, explicitly aligning semantic item representations with collaborative similarity [34].

2.1.2 Vision, Audio, and Multimodal Encoders

Some domains require representations beyond text. Fashion recommendation depends heavily on item appearance, music recommendation can use audio signals, and video recommendation may combine visual, textual, and audio information.

Early visual recommenders used image features to complement collaborative filtering. VBPR integrated CNN-derived image features into Bayesian Personalized Ranking, showing that appearance is useful in domains like fashion where user preferences are strongly visual [56]. DVBPR later trained image representations directly for recommendation, instead of relying on fixed CNN features extracted from image classification models [57].

Modern multimodal recommendation pipelines increasingly use large pretrained encoders. CLIP learns aligned image and text representations from natural language, making it possible to compare visual and textual content in a shared embedding space [8]. SigLIP modifies contrastive image-text pretraining with a sigmoid loss, while DINOv2 provides strong self-supervised visual representations trained without language supervision [9, 7]. The paper *Large Multi-modal Encoders for Recommendation* studies CLIP-like and VLM-style backbones in recommendation settings and finds that stronger multimodal encoders and end-to-end adaptation can improve recommendation performance over shallow modality-specific feature extraction [58]. LIRDRec

pushes further toward identifier free multimodal recommendation by learning item representations directly from multimodal features rather than relying on learned item ID embeddings [59].

2.1.3 Cold-Start and Cross-Domain Adaptation

Cold-start recommendation is one of the main motivations to use content encoders. If an item has no interactions, a collaborative model cannot learn a reliable ID embedding. Content-based representations can still be computed from text, images, or metadata. DropoutNet simulates cold-start during training by removing collaborative inputs and forcing the model to rely on content features [60]. Heater aligns content-based and collaborative representations through a transfer procedure designed for cold-start recommendation [61]. CLCRec formulates cold-start recommendation as contrastive learning between collaborative and content views [62].

Transfer learning is a related but broader problem. Item IDs are specific to a catalog, so ID embeddings learned in one domain cannot be directly reused in another. Modality-based representations, by contrast, can transfer if item content is comparable across domains. ZESRec and UniSRec demonstrate this idea for text-based sequential recommendation, while multimodal approaches such as TransRec and PMMRec explore transfer through raw modality features and cross-modal alignment [63, 64].

2.1.4 Meta-Learning for Recommendation

Meta-learning has been applied to recommendation primarily as a way to handle user and item cold-start, framing each user or item as a task and learning how to adapt a recommender from a small number of interactions. MeLU treats each user as a task and uses MAML [26] to learn a recommender initialization that can be adapted to a new user from a handful of interactions through a few gradient steps [65]. MAMO extends this with a memory-augmented meta-optimizer to better handle users with diverging preferences [66]. Item-side cold-start has been addressed by similar techniques, where the meta-learner produces an item representation that can be refined from a few observed interactions.

A second line of work treats domains as meta-learning tasks for cross-domain recommendation. PTUPCDR [67] learns a meta-network that produces personalized user-preference transforms from a small number of source-domain interactions, conceptually close to the dataset and domain transfer scenarios studied in this thesis.

Most of the methods above are optimization-based and require an explicit inner gradient loop at evaluation time. Amortized alternatives like the Conditional Neural Processes [68] have received less attention in recommendation, although attention- and gating-based modules that produce task-conditioned representations from a support set fall into this category and are computationally much cheaper, since adaptation reduces to a forward pass.

2.2 Graph-Based Recommendation

Graph recommendation can be understood as a structural extension of collaborative filtering, representing interactions as a user–item bipartite graph and propagate information through edges.

PinSage showed that graph convolutions could scale to web-scale recommendation by combining random walk sampling with efficient graph convolution over item neighborhoods [69]. NGCF then made collaborative signal explicit by propagating embeddings over the user–item graph [70]. LightGCN simplified this architecture by removing feature transformations and nonlinearities, showing that much of the benefit comes from neighborhood aggregation itself rather than heavy neural layers [71].

Later work introduced self-supervised objectives for graph recommendation. SGL applies graph contrastive learning by creating augmented views of the user–item graph, improving ro-

bustness under sparse interactions [72]. SimGCL further simplifies this idea by perturbing embeddings directly rather than relying on graph augmentations [73].

Multimodal graph recommendation incorporates content features into graph collaborative learning. MMGCN propagates modality-specific representations over user-item graphs, allowing visual, textual, and other modality signals to interact with collaborative structure [74]. FREE-DOM argues that item-item graphs built from content features can be useful when kept fixed, while training focuses on denoising the user-item graph [75]. MENTOR adds multi-level cross-modal alignment, aiming to preserve modality-specific information while still learning a unified recommendation representation [76].

2.3 LLM-Based and Generative Recommendation

The advancements of NLP and Large language models (LLMs) introduced a new way to think about recommendation. Instead of producing dense item vectors, LLM-based systems can express user histories, item attributes, explanations, and recommendations in natural language. This makes them attractive for conversational and explainable recommendation, but it also creates new challenges: item grounding, hallucination, ranking quality, and the mismatch between language modeling and personalized preference prediction.

P5 reformulated several recommendation tasks as text-to-text problems, using prompts to unify rating prediction, sequential recommendation, explanation, and review-related tasks [77]. This work argues, that recommendation data can be converted into natural language instructions and solved with an encoder-decoder language model.

A second line of work adapts large language models through instruction tuning. TALLRec shows that tuning LLMs on recommendation-specific instructions can improve their ability to interpret user preference and item relevance [78]. Recommendation as Instruction Following similarly frames recommendation tasks as instruction-following problems, emphasizing that LLMs need recommendation-specific supervision rather than generic prompting alone [79]. These approaches are useful when recommendation is interactive, but they do not by themselves solve the problem of efficient retrieval over large item catalogs.

2.4 Datasets and Evaluation Frameworks

The development of large models for recommendation depends heavily on the available datasets. A useful dataset for this area should contain user-item interactions, rich item content, sufficient scale, and preferably multiple domains or categories. In practice, such datasets are limited, which makes systematic comparison difficult.

2.4.1 Datasets for Foundation-Model Recommendation

The Amazon Reviews 2023 collection is one of the most important recent datasets for recommendation. It contains large-scale user reviews, item metadata, product description, and product images across many categories [80]. Because the same source provides interactions and rich item information, Amazon Reviews 2023 is useful for evaluating multimodal, cross-category, and retrieval-oriented recommendation. Its category structure also makes it suitable for transfer experiments.

NineRec is a recent benchmark suite designed for transferable multimodal recommendation. It contains a large-scale source domain and nine target domains, with items accompanied by textual descriptions and high resolution cover images [81].

For fashion recommendation, the H&M Personalized Fashion Recommendations dataset provides a large retail setting with user transactions, article metadata, images and detailed product

attributes [82]. Apparel recommendation depends strongly on visual and categorical properties like product type, color, style, and garment group.

MovieLens 20M remains a standard collaborative filtering benchmark because it is stable, widely used, and well understood [83]. By itself, it is primarily an explicit-feedback dataset with limited item-side metadata, mainly movie titles, genres, tags, and external identifiers. However, it can be enriched with external information using the MovieLens 20M Posters for Machine Learning dataset from Kaggle, which provides poster images for MovieLens 20M movies [84].

For news recommendation, the Microsoft News Dataset (MIND) is one of the main public benchmarks. It is built from anonymized behavior logs from Microsoft News and contains user impression logs together with article information such as titles, abstracts, categories, and subcategories [85]. MIND is especially relevant for text-based recommendation because news articles are frequently new, short-lived, and described primarily through text.

Short video datasets provide settings where recommendation can be studied with richer behavioral signals than simple implicit feedback. KuaiRec was designed with a fully observed user-item interaction matrix for a subset of users and videos [86]. KuaiSAR extends this direction by combining search and recommendation behavior within the same platform [87].

Other public datasets support narrower but still relevant evaluation scenarios. Yelp provides business metadata, reviews, photos, and user feedback [88]. RetailRocket contains e-commerce event logs and item properties, making it useful for session-based or event-based retail recommendation [89]. Book Crossing and Goodbooks10k provide book interactions and metadata [90, 91]. LFM-1B provides large-scale music listening histories and can be paired with artist, tag, or audio information, making it relevant for studying sequential and content-aware music recommendation [92].

2.4.2 Evaluation Frameworks

Evaluation frameworks play an important role for reliable comparability of recommenders. An evaluation framework usually standardizes data loading, train-validation-test splitting, negative sampling, model implementation, hyperparameter search, and ranking metrics. This reduces the amount of hidden experimental variation and makes results easier to reproduce. However, different frameworks emphasize different paradigms, which affects what models and scenarios they are suitable for.

Several frameworks target general recommender system experimentation. LensKit is one of the older research oriented toolkits that focuses on flexible experimentation, algorithm comparison, and education [93]. Surprise follows a scikit-learn interface and is mainly aimed at explicit feedback rating prediction, making it useful for classical experiments rather than modern multi-modal recommendation [94].

More recent libraries focus on neural recommendation and broader task coverage. RecBole provides a unified PyTorch framework with implementations of many recommendation models, common benchmark datasets, and standardized evaluation protocols [95]. RecBole 2.0 extends this ecosystem with packages for meta-learning, data augmentation, debiasing, fairness, cross-domain recommendation, transformers, and graph neural networks [96]. Elliot focuses on reproducibility, it allows complete experimental pipelines to be specified through configuration files and supports many splitting strategies, filtering methods, hyperparameter optimization procedures, metrics, and statistical tests [97]. Microsoft Recommenders provides utilities, examples, notebooks, baseline implementations, dataset processing tools, and evaluation functions for practical recommender-system development [**msrecommenders**]. TensorFlow Recommenders supports the workflow of building retrieval and ranking models in TensorFlow, including data preparation, model definition, training, evaluation, and deployment [98]. TorchRec targets large-scale production recommendation in PyTorch by providing primitives for sparse features, embedding tables, sharding, and distributed training [99].

Other frameworks specialize in particular recommendation paradigms. Cornac focuses on

multimodal and auxiliary information recommendation, where models can use text, images, social links, or other side information in addition to interactions [100]. MMRec is a more recent multimodal recommendation framework that canonicalizes the implementation and comparison of models using visual, textual, and structural signals [101]. OpenP5 provides an experimental platform for LLM-based generative recommendation under the P5 paradigm, covering different downstream tasks, datasets, and item indexing choices [102]. GRID focuses on generative recommendation with semantic identifiers and makes it possible to compare tokenization, codebook construction, sequential modeling, and decoding strategies [103].

Overall, existing frameworks cover much of the recommender system landscape, including classical collaborative filtering, neural recommendation, multimodal recommendation, production-scale ranking, and generative recommendation. Their main focus, however, is the complete recommendation pipeline. They standardize models, datasets, training procedures, and metrics, but they rarely separate the contribution of the representation model from the contribution of the downstream recommender. As a result, it is difficult to determine whether a performance difference comes from a stronger embedding backbone, a better ranking model, a different loss function, or other pipeline choice. This limitation motivates REEF a new evaluation framework, introduced in Chapter 3. This framework treats the representation model as an explicit and interchangeable component. Instead of evaluating each backbone inside a different recommender architecture, it evaluates different embeddings under the same offline recommendation protocol. The datasets, splits, candidate construction, downstream rankers, and metrics are kept fixed across backbones.

Recommender Embedding Evaluation Framework

This chapter introduces Recommender Embedding Evaluation Framework (REEF), a framework designed to evaluate how much do representations produced by an embedding foundation model contribute to downstream recommendation tasks. An overview of the REEF pipeline can be seen in Figure 3.1. It supports direct comparison of multiple foundation models within a single run, under identical data, task, and evaluation conditions, while supporting reproducible experimental setups through controlled configuration, seeding, and artifact tracking. REEF does so by training task-specific downstream recommenders on top of the embeddings generated by each foundation model and measuring the performance of the resulting system. The foundation model transforms item content, such as text or images, and where supported, structured metadata, into embeddings. The downstream model then uses these representations to produce predictions for a particular recommendation scenario. These scenarios may correspond to any recommendation setting for which compatible task, downstream model, and evaluator components are provided.

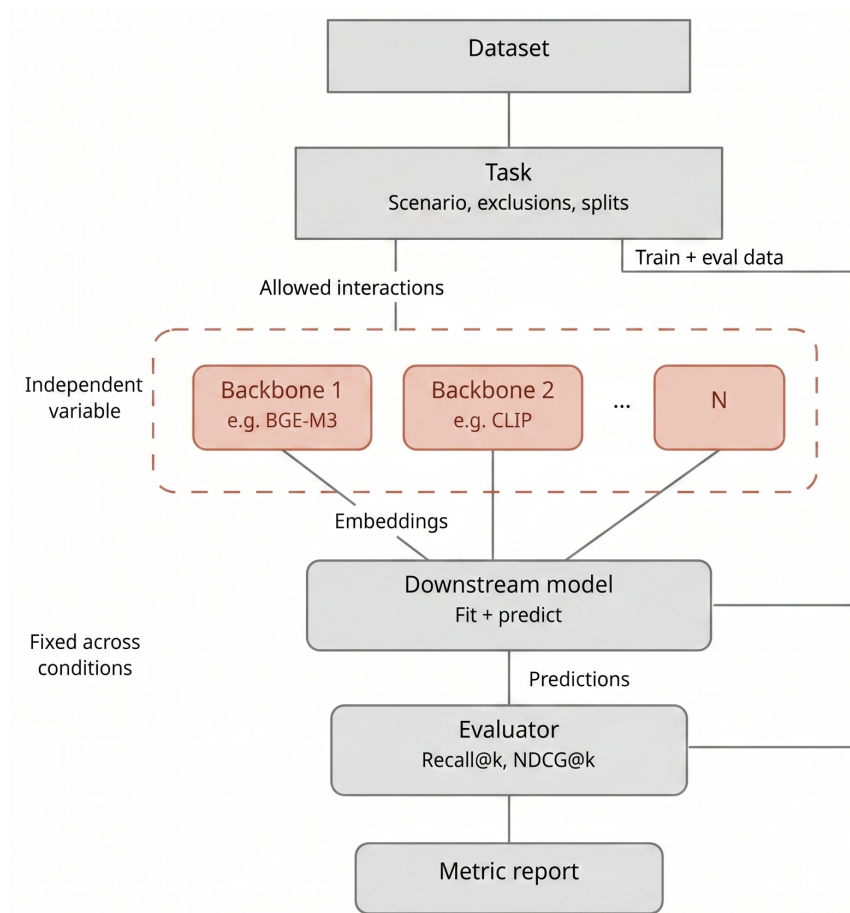
REEF is not intended to replace existing recommender frameworks, its purpose is to provide a focused and reproducible framework for evaluating foundation-model representations in recommendation. To achieve this, REEF defines a small set of protocols needed for an experiment. These interfaces are intentionally minimal, so that components developed in other frameworks can be incorporated with little additional work.

3.1 Pipeline Overview

A REEF experiment is specified through a YAML configuration file. The configuration describes which dataset should be loaded, which recommendation tasks should be evaluated, which foundation backbones should be compared, how these backbones should be fine-tuned, which downstream models should be fitted on top of the resulting embeddings, and which metrics should be computed. In this way, the configuration file defines the full experimental setup without requiring changes to the framework runtime.

At a high level, a REEF run proceeds through a fixed sequence of stages. First, the configured data loader reads the raw dataset files and converts them into a common internal representation. Then, each configured task declares what data it needs and which interactions must be excluded from backbone training in order to preserve the validity of the evaluation scenario. After these exclusions are resolved, each task constructs a task-specific view of the data used later for downstream training and evaluation.

The framework then evaluates each configured foundation-model backbone. Depending on



■ **Figure 3.1** End-to-end pipeline of REEF.

the selected trainer, the backbone may be used without adaptation or fine-tuned on the allowed interaction data. After this stage, the backbone is frozen and used to encode the required items into embeddings. These embeddings form the representation artifact that is passed to the downstream recommendation stage.

For each task, REEF trains the configured downstream model on top of the embeddings produced by the backbone. The downstream model receives only the task-specific data view and the embedding bundle, which keeps the evaluation focused on how useful the foundation model representations are for the given recommendation scenario. The trained downstream model then produces predictions for the task queries, and the evaluator computes the selected ranking or retrieval metrics.

The main user defined components in a REEF configuration are:

- **Data loader:** reads the raw dataset files and constructs the shared data representation used by the rest of the pipeline.
- **Tasks:** define the recommendation scenarios, specify task-specific data requirements, and construct the data views used for downstream training and evaluation.
- **Backbones:** define the foundation models whose item representations are being evaluated.
- **Trainers:** specify whether and how a backbone is adapted before embeddings are generated.

- **Downstream models:** consume the generated embeddings, train on task-specific supervision, and produce predictions for the given evaluation scenario.
- **Evaluators:** compute the selected metrics from the generated predictions.

REEF treats these components as interchangeable modules with typed interfaces. The runtime is responsible for resolving the components from the configuration, checking that they satisfy the expected protocols, executing them in the correct order, and storing the resulting artifacts. This design keeps the execution logic fixed while allowing new datasets, tasks, backbones, trainers, downstream models, and evaluators to be added without modifying the core pipeline.

The pipeline can therefore be understood as a sequence of artifact transformations. The data loader produces a prepared data bundle. Tasks transform this bundle into task-specific views. Trainers adapt backbones using the permitted training interactions. Backbones produce embedding bundles. Downstream models transform task views and embeddings into predictions. Evaluators transform predictions into metric reports. Finally, REEF persists the results together with the configuration and references to the generated artifacts, making experiments easier to inspect, reproduce, and compare.

3.2 Configuration and Component Resolution

REEF is driven by a YAML configuration file that specifies an experiment in its entirety. The file declares the data bundle loader, one or more backbone-trainer pairs, one or more task-downstream-evaluator combinations. Each component is referenced by its Python import path via the `module:attribute` pattern, together with a parameter mapping. An example configuration file is shown in Appendix A.

3.3 Data Model and Loading

The basic unit of data in REEF is an **Interaction**, a frozen dataclass representing one observed event between a user and an item. Interactions are not stored as separate Python objects. Instead, each **Dataset** keeps them in an **InteractionTable** backed by Apache Arrow, and **Interaction** instances are created only when individual rows need to be accessed. This makes the representation more memory efficient and allows interactions to be filtered and processed efficiently.

■ Code listing 3.1 Interaction row schema.

```
@dataclass(frozen=True)
class Interaction:
    interaction_id: InteractionId
    user_id: str
    item_id: str
    timestamp: float | None = None
    label: float | None = None
    domain_id: str | None = None
    session_id: str | None = None
    sequence_id: str | None = None
    metadata: Mapping[str, Any] = field(default_factory=dict)
```

The only required fields are the user and item identifiers. All other fields are optional and are used only by tasks that require them. The `timestamp` field supports chronological ordering and sequential evaluation. The `label` field stores an explicit supervision signal, such as a rating or binary outcome. The `domain_id` field distinguishes interactions that belong to different domains,

which is required in transfer settings. The `session_id` and `sequence_id` fields identify local interaction sequences when tasks depend on session-based or ordered behaviour.

The `interaction_id` is a stable integer identifier derived from the content of the interaction rather than from its position in the table. It is computed from a canonical representation of the fields that distinguish one event from another, such as the user, item, timestamp, and selected metadata. Missing values are represented explicitly, and repeated events remain distinct only if the loader includes an additional field that separates them. Because the identifier depends on content rather than row order, the same interaction keeps the same identifier even if the dataset is reloaded, shuffled, or rewritten. This allows exclusions and cached task views to remain valid even when the table order changes. The `metadata` field stores any additional task-specific information that does not belong in the core schema, such as search queries, conversational context, or impression features.

A `Dataset` combines an `InteractionTable` with its user and item catalogues and is identified by a stable `dataset_id`. All identifiers in REEF are scoped to the dataset in which they appear. As a result, two datasets may contain the same `user_id` or `item_id` string without the framework treating them as the same entity. When cross-dataset user identity is required, REEF represents it explicitly through an optional `UserLinkTable`. Its rows declare equivalences of the form (`dataset_a`, `user_a`, `dataset_b`, `user_b`) and are loaded alongside the bundle only for tasks that require cross-dataset user modeling.

■ **Code listing 3.2** Dataset and data bundle.

```
@dataclass(frozen=True)
class Dataset:
    dataset_id: str
    items: ItemCatalog
    users: UserCatalog
    interactions: InteractionTable
    domains: Mapping[str, DomainSpec] = field(default_factory=dict)
    sessions: Mapping[str, SessionSpec] = field(default_factory=dict)
    metadata: Mapping[str, Any] = field(default_factory=dict)

@dataclass(frozen=True)
class DataBundle:
    bundle_id: str
    datasets: Mapping[str, Dataset]
    user_links: UserLinkTable | None = None
    metadata: Mapping[str, Any] = field(default_factory=dict)
```

The loading stage is defined by a `LoaderProtocol`, whose `load` method takes a `RunContext` and returns a `DataBundle`. The protocol does not restrict how the data are obtained, it can read local files, download remote resources, or combine multiple sources. Once loading is complete, the runtime validates the bundle by checking interaction identifiers, verifying that `dataset_ids` are unique, and ensuring that the dataset-level requirements declared by the active tasks are met.

3.4 Tasks and Task Views

A task defines a single evaluation condition in REEF. It specifies what is being evaluated, what data are needed to construct that evaluation, and which interactions must be excluded from backbone training in order to keep the evaluation valid. Each task also declares a `kind`, an open string that identifies the prediction format it produces. In other words, the task defines the meaning of the scenario, while the `kind` defines the structure of the queries, predictions, and evaluation. The built-in evaluators described in Section 3.7 support the `ranking` kind, including

sequential next-item prediction expressed as ranking over candidate items. The framework itself does not impose a fixed set of valid kinds.

■ **Code listing 3.3** Task protocol.

```
@runtime_checkable
class TaskProtocol(Protocol):
    task_id: str
    kind: str

    def describe_requirements(
        self,
        ctx: RunContext
    ) -> TaskRequirements: ...

    def declare_exclusions(
        self,
        bundle: DataBundle
    ) -> TrainingExclusions: ...

    def build_view(
        self,
        bundle: DataBundle,
        trainable: Mapping[str, InteractionIdSet],
        ctx: RunContext,
    ) -> TaskView: ...
```

In the first phase, the task declares its data requirements. These may specify, for example, that timestamps must be present, that the `domain_id` field is required, or that a particular dataset must be included in the bundle. The runtime checks the loaded bundle against these requirements after loading.

In the second phase, the task declares which interactions must be excluded from backbone training. These exclusions are stored in a `TrainingExclusions` artifact keyed by `dataset_id`. Each value is an `InteractionIdSet`, represented as a roaring bitmap over stable interaction identifiers. The runtime merges the exclusions declared by all active tasks and resolves a single trainable set for each dataset. If the requirements of the active tasks cannot be satisfied together, the run fails.

In the third phase, the task receives the resolved trainable set and uses it to construct a `TaskView`. This view is the only task-specific data artifact passed to downstream models.

■ **Code listing 3.4** Training exclusions and task view.

```
@dataclass(frozen=True)
class TrainingExclusions:
    excluded_ids: Mapping[str, InteractionIdSet]
    reason: str = ""

@dataclass(frozen=True)
class TaskView:
    task_id: str
    kind: str
    exclusions: TrainingExclusions
    fit_payload: Any
    eval_queries: Sequence[Query]
    val_queries: Sequence[Query] | None
    auxiliary: Mapping[str, Any] = field(default_factory=dict)
```

The `TaskView` separates `fit_payload` from `eval_queries` because the data used for downstream training does not have to have the same structure as the data used for prediction and evaluation. The fit payload contains the task-specific inputs provided to the downstream model during fitting, while the evaluation queries define the supervised prediction targets on which metrics are computed. Each evaluation query is represented by a small common wrapper whose internal structure depends on the task `kind`.

■ **Code listing 3.5** Query envelope.

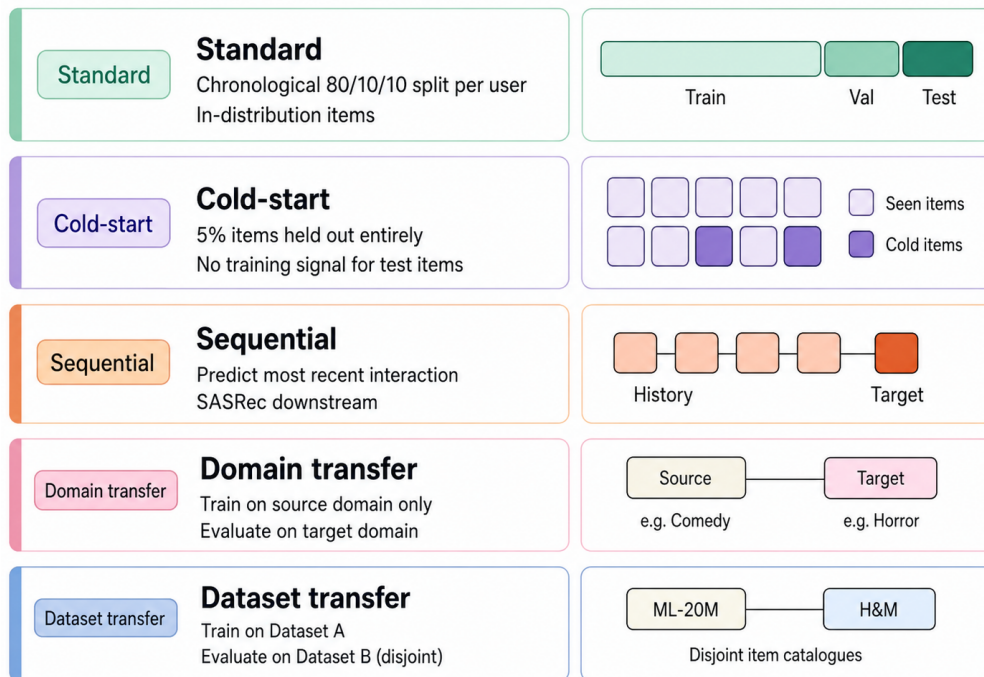
```
@dataclass(frozen=True)
class Query:
    query_id: str
    dataset_id: str
    kind: str
    payload: Any
    target: Any
```

For ranking tasks, the payload typically contains a user representation, an interaction history, and a candidate pool, while the target is the set of relevant items. For sequential next-item prediction, the history is ordered and the target is the next item, although the prediction is still formulated as ranking over candidate items. For search tasks, the payload may consist of a textual or multimodal query and the target of a graded relevance set.

3.4.1 Built-In Tasks

REEF provides five built-in task implementations corresponding to the evaluation settings used in Chapters 4 and 5. These tasks cover standard in-distribution evaluation, cold-start item evaluation, sequential next-interaction prediction, domain transfer, and dataset transfer. An overview of the five evaluation scenarios is shown in Figure 3.2.

- **Standard ranking** holds out a configurable fraction of interactions for evaluation. Each held-out interaction (u, i^*) becomes a query in which the user's earlier interactions form the history and the target item i^* is ranked against sampled negative items. The held-out interactions are excluded from backbone training, while the remaining interactions can be used for downstream fitting.
- **Cold-start ranking** selects a configurable fraction of items as cold items. All interactions involving these items are excluded from backbone training and from downstream model fitting, and are used only for evaluation.
- **Sequential ranking** holds out a configurable number of the most recent interactions per user, given by `holdout_per_user`, for evaluation. The earlier interactions remain available as history for training and prediction. Each held-out interaction becomes a query in which the ordered interaction prefix is used to predict the next item.
- **Domain transfer** splits a dataset into source and target domains using `domain_id`, with the target specified by `target_domain`. Interactions from the target domain are excluded from both backbone training and downstream model fitting, and are used only for evaluation.
- **Dataset transfer** uses two datasets, specified by `source_dataset` and `target_dataset`, with disjoint item catalogues. All interactions from the target dataset are excluded from both backbone training and downstream model fitting. Evaluation is then performed on the target dataset, so the model must rank target items without being adapted on target-dataset interactions.



■ **Figure 3.2** Overview of the five evaluation scenarios implemented. The standard setting uses a chronological per-user split with in-distribution items. The cold-start setting withholds a subset of items entirely from training. The sequential setting evaluates prediction of the most recent interaction from user history. The domain transfer setting trains on a source domain and evaluates on a target domain. The dataset transfer setting trains and evaluates across disjoint datasets with separate item catalogs.

3.5 Backbone Training and Embedding Generation

For each selected backbone and trainer, REEF passes the training bundle together to the trainer, which adapts the foundation model according to the chosen training procedure. Once training is complete, the backbone is used to encode the required items and produce an `EmbeddingBundle`, which stores item embeddings for each dataset and, where available, may also provide user embeddings or access to encoding of non-item inputs. All methods in this section receive the same `RunContext` instance, which is created once at the start of the run and passed through the entire pipeline.

■ **Code listing 3.6** Run context carried through the pipeline.

```
@dataclass(frozen=True)
class RunContext:
    run_id: str
    rng: np.random.Generator
    device: Device
    artifact_store: ArtifactStore
    cache: ArtifactCache
    logger: StructuredLogger
    clock: Clock
    seed: int
    time_window: TimeWindow | None = None
```

3.5.1 Backbone Protocol

The backbone protocol defines a minimal interface for the foundation models:

■ **Code listing 3.7** Backbone protocol.

```
@runtime_checkable
class BackboneProtocol(Protocol):
    supported_modalities: frozenset[str]
    def embed_items(
        self,
        items: Sequence[ItemInput]
    ) -> EmbeddingArray:
        ...

    def encode(
        self,
        payload: Any, dataset_id: str
    ) -> EmbeddingArray:
        ...

@dataclass(frozen=True)
class ItemInput:
    item_id: str
    dataset_id: str
    text: TextSpec | None = None
    image: ImageSpec | None = None
    tabular: TabularSpec | None = None
    categorical: CategoricalSpec | None = None
    metadata: Mapping[str, Any] = field(default_factory=dict)
```

`ItemInput` stores item features in separate modality fields, such as text or image. The `embed_items` method produces item embeddings in the model's shared embedding space. The `encode` method applies the same model to non-item inputs, such as search queries or conversational context, and returns vectors in that same space. Some backbones may also provide user embeddings through implementation-specific extensions. When available, these can be stored in the resulting embedding bundle, but they are not required by the basic protocol.

3.5.2 Trainer Protocol

The training stage is defined by the `TrainerProtocol`. Its `train` method receives the backbone, the data bundle, and the trainable interaction sets produced after applying the task exclusions described in Section 3.4. The trainer returns the adapted backbone together with a `TrainingTrace` recording information like training step, loss, and validation metric.

This interface also makes multi-dataset training explicit. A trainer may iterate jointly over all datasets in the bundle using the interactions allowed for training, and any cross-dataset sampling strategy is treated as part of the trainer configuration.

■ **Code listing 3.8** Trainer protocol.

```
@runtime_checkable
class TrainerProtocol(Protocol):
    def train(
        self,
        model: Any,
        bundle: DataBundle,
```

```

    trainable: Mapping[str, InteractionIdSet],
    config: Mapping[str, Any],
    ctx: RunContext,
) -> tuple[BackboneProtocol, TrainingTrace]: ...

```

3.5.3 Embedding Generation

After training, REEF constructs `ItemInput` objects for all items required by the active tasks and passes them through the backbone in batches. The embeddings are then stored in an `EmbeddingBundle`, grouped by `dataset_id`. To support large item catalogues, these embeddings are written as on-disk shards rather than kept as dense in-memory matrices. The bundle provides a lookup interface that loads only the shards needed for the requested items.

■ **Code listing 3.9** Embedding bundle.

```

@dataclass(frozen=True)
class EmbeddingBundle:
    backbone_id: str
    item_embeddings: Mapping[str, ShardedEmbeddingStore]
    user_embeddings: Mapping[str, ShardedEmbeddingStore] | None
    encode: Callable[[Any, str], EmbeddingArray]
    metadata: Mapping[str, Any] = field(default_factory=dict)

```

The `encode` field provides runtime access to the trained and frozen backbone for non-item inputs. A downstream model can use it to embed a search query, a conversation history, or another task-specific input into the same space as the item embeddings. The `EmbeddingBundle` is therefore a runtime-facing composite artifact: its stored embedding shards are persistent, while the `encode` callable is a live interface to the frozen backbone used during downstream prediction.

3.6 Downstream Adaptation and Prediction

Each task is evaluated independently with a downstream model. For every backbone–task pair, REEF passes the corresponding `TaskView` and `EmbeddingBundle` to the downstream model, which fits on the task data and produces predictions.

■ **Code listing 3.10** Downstream model protocol.

```
@runtime_checkable
class DownstreamModelProtocol(Protocol):
    kind: str

    def fit(
        self,
        view: TaskView,
        embeddings: EmbeddingBundle | None,
        config: Mapping[str, Any],
        run_context: RunContext,
    ) -> None: ...

    def predict(
        self,
        view: TaskView,
        embeddings: EmbeddingBundle | None,
        run_context: RunContext,
    ) -> TaskPredictions: ...
```

Each downstream model is trained and evaluated for one task at a time, so supervision from one task cannot affect another. The downstream model receives the data prepared specifically for that task. Where needed, it uses the backbone embeddings as input representations, for example for items in a user’s history or candidate items to be scored. If the task also contains non-item inputs, such as a search query or conversational context, these can be embedded into the same space through the frozen backbone. The downstream model is then free to apply any task-specific learning and scoring procedure on top of these representations.

3.7 Evaluation and Result Persistence

Finally, evaluation is carried out, the `EvaluatorProtocol` defines this stage through an `evaluate` method, which receives the task predictions together with the corresponding task views and returns an `EvaluationReport`, typically computed per task or over a set of tasks.

■ **Code listing 3.11** Evaluator protocol.

```
@runtime_checkable
class EvaluatorProtocol(Protocol):
    supported_kinds: frozenset[str]

    def evaluate(
        self,
        predictions: Mapping[str, TaskPredictions],
        views: Mapping[str, TaskView],
        run_context: RunContext,
    ) -> EvaluationReport: ...
```

3.8 Extensibility

REEF is designed to be easily extended. A user can add a new loader, task, backbone, trainer, downstream model, or evaluator by implementing the corresponding interface. These protocols are intentionally kept as small as possible, so that new components can vary widely in their internal design while still remaining compatible with the runtime. This allows new methods

to be evaluated within the existing pipeline without modifying the rest of the framework. The same mechanism also supports integration of components from external libraries. As long as a task, downstream model, and evaluator agree on the prediction kind and its schema, the new component can be inserted into the existing runtime.

Zero-Shot Evaluation of Foundation Models

This chapter uses the REEF framework introduced in Chapter 3 to evaluate pretrained foundation encoders in recommendation. The objective is to determine how effectively existing text, image, and multimodal encoders perform when used directly, without any task-specific adaptation. Each encoder is treated as a frozen feature extractor.

4.1 Experimental Scope

The evaluation covers seven encoders and one non-semantic baseline. All configurations are evaluated on three public recommendation datasets, namely Amazon Reviews 2023, H&M Fashion, and MovieLens 20M, under five evaluation scenarios: standard recommendation, cold-start recommendation, sequential recommendation, domain transfer, and dataset transfer.

We report Recall@10 and NDCG@10 as the primary ranking metrics in this and all subsequent chapters. The two metrics capture complementary aspects of top- k recommendation. Recall@10 measures whether relevant items are retrieved within the first ten positions, whereas NDCG@10 evaluates not only their presence but also their ordering. We chose to only report two metrics as the experimental design already spans eight encoder conditions, three datasets, and five downstream scenarios, which gives 120 evaluation cells before considering any training configuration. Adding additional metrics would make the results harder to interpret.

4.2 Evaluated Models

The evaluation is done on three modality groups: text, image, and multimodal. The encoders used are summarised in Table 4.1. The models were selected to cover a range of pretraining approaches and parameter scales. Every encoder is implemented as a REEF backbone. Each encoder is then followed by a per-encoder linear alignment $A_e : \mathbb{R}^{d_e} \rightarrow \mathbb{R}^{256}$ that maps its native embedding to a shared 256-dimensional space. This projection ensures that all downstream models receive inputs of the same dimensionality, and therefore have the same number of trainable parameters. The resulting vectors are L_2 -normalised.

4.2.1 Text Encoders

Three text encoders are included in the experiment. They differ in model scale, training data, and pretraining strategy. All three consume concatenated textual item metadata and produce a single text embedding per item.

E5-base-v2 is a text embedding model trained with weakly supervised contrastive pretraining on a large corpus of text pairs. It produces 768-dimensional sentence embeddings and serves as a contrastive retrieval baseline at moderate scale [44].

BGE-M3 is a larger multilingual embedding model that supports dense, sparse, and multi-vector retrieval within a single encoder. Its dense output has dimension 1024, and the model is designed to cover more than 100 languages and substantially longer inputs than standard encoders [45]. It represents a higher-capacity model with a more elaborate training pipeline based on unified fine-tuning and self-knowledge distillation.

GTE-base is a text embedding model trained with multi-stage contrastive learning on a large corpus of relevance pairs [46]. Like *E5-base-v2*, it produces 768-dimensional embeddings and is therefore comparable in scale, while differing in training data and optimisation strategy.

4.2.2 Vision Encoders

Two vision encoders are included.

DINOv2-base is a ViT-B vision encoder trained without language supervision using the DINOv2 self-distillation framework [7]. The base variant has 86 M parameters and embedding dimension 768. DINOv2 represents a self-supervised visual backbone that captures semantic and structural properties of images without relying on image–text alignment.

SigLIP2-base is a multilingual vision–language encoder that extends the original SigLIP training with additional objectives, including decoder-based pretraining, self-distillation, and masked prediction [9, 104]. Its vision tower produces 768-dimensional image embeddings. In this benchmark, SigLIP2 is used both as a vision-only backbone and as the vision tower in the multimodal configuration described below.

4.2.3 Multimodal Encoders

Two multimodal configurations are included.

BGE-M3 + SigLIP2 combines the two separately pretrained encoders by L_2 -normalising each modality vector and concatenating them into a single 1792-dimensional item embedding.

CLIP ViT-B/16 is a natively aligned vision–language model whose text and image encoders are jointly pretrained with a contrastive objective over image–text pairs [8]. The text and image embeddings exist in a shared 512-dimensional space and are averaged into a single 512-dimensional item embedding.

4.2.4 Baseline

The *Baseline* serves as a non-foundation reference. Each item is assigned a random 256-dimensional embedding drawn from a standard normal distribution. The baseline carries no semantic information about the items, it isolates the contribution of the downstream ranker from that of the encoder, and any encoder that fails to outperform the baseline is providing no useful signal.

■ **Table 4.1** Encoder configurations evaluated in the zero-shot regime. The output dimension column reports each encoder’s native dimensions.

Encoder	Modality	Parameters	Output dim	Pretraining
E5-base-v2	text	110 M	768	Contrastive (web text)
BGE-M3	text	568 M	1024	Multi-stage contrastive + instruction
GTE-base	text	110 M	768	Multi-stage contrastive
DINOv2-base	image	86 M	768	Self-supervised distillation
SigLIP2-base	image	86 M	768	Sigmoid contrastive (image-text)
BGE-M3 + SigLIP2	text + image	654 M	1792	Concatenation of two pretrained encoders
CLIP ViT-B/16	text + image	150 M	512	Joint contrastive (image-text)
Baseline	—	—	256	None (random vectors)

4.3 Datasets

The experiments are conducted on three public recommendation datasets, all introduced in Section 2.4.1. These are Amazon Reviews 2023, H&M Fashion, and MovieLens 20M. The choice was driven by four requirements. First, every dataset must provide both textual and visual item metadata, so that the same items can be encoded by text-only, vision-only, and multimodal backbones. Second, every dataset must contain enough interactions to support stable splitting and the scenario-specific exclusions. Third, every dataset must expose meaningful domain labels that allow domain transfer experiments. Fourth, the suite of datasets should vary in how textual and visual aspects of items contribute.

MovieLens 20M, extended with poster images, is included as a media dataset with a media recommendation setting characterized by explicit feedback and weaker, less direct content signals [83, 84]. Each item has a title, genre metadata, and a poster image.

H&M Fashion captures purchase events from the H&M online fashion retailer [82]. It is included as a setting in which visual information plays a central role in user choice, making it good for evaluating vision-only and multimodal backbones. Each item is described by short textual metadata, such as article name, product type, and color group, together with a high-resolution product image on a neutral background.

Amazon Reviews 2023 is a large and diverse retail benchmark in which the textual metadata typically carries the strongest semantic signal [80]. The dataset provides product categories, to keep the experimental cost manageable, this thesis uses only three categories: **All.Beauty**, **Amazon.Fashion**, and **Toys.and.Games**. Each item is associated with textual metadata, including title, description, and brand information, together with one or more product images.

4.3.1 Dataset Preprocessing

A shared preprocessing pipeline is applied to all three datasets. Timestamps are normalized to a common UTC representation. For the textual modality, dataset-specific metadata fields are concatenated into a single string per item using newline separators, with repeated fragments removed. The exact source fields differ by dataset but the result is always a single textual input per item. For the visual modality, images are kept at their original resolution, and each model applies its own image preprocessing pipeline. Items with a missing modality are given a zero-valued embedding. Then, we apply filtering, making sure every user and item has at least 10 interactions.

4.4 Evaluation Scenarios

This section explains the five downstream scenarios, all of which are implemented as REEF tasks. Item embeddings are reused across all five scenarios. The scenarios differ in interactions split, the candidate pool, and how user history is constructed at evaluation time. Unless stated otherwise, every evaluation query ranks the target item against 999 uniformly sampled negative items drawn from the dataset-specific candidate pool defined by scenarios.

For the four non-sequential scenarios (standard, cold-start, domain transfer, and dataset transfer), the downstream ranker is a trainable model operating on the shared 256-dimensional aligned item embeddings. The model first maps each aligned item embedding through a shared item tower $f_\theta : \mathbb{R}^{256} \rightarrow \mathbb{R}^{128}$, implemented as a two-layer MLP with hidden dimension 256, GELU activations, layer normalization, dropout, and a final L_2 normalization. The same item tower is applied to both history items and candidate items, ensuring that all encoders are evaluated with the same downstream architecture and parameter count.

For a user u with trainable history H_u , the transformed history embeddings $h_j = f_\theta(\tilde{a}_j)$ are aggregated using a learnable attention pooling module. Attention scores are computed as:

$$e_{uj} = v^\top \tanh(W h_j + b),$$

where $W \in \mathbb{R}^{128 \times 128}$, $b \in \mathbb{R}^{128}$, and $v \in \mathbb{R}^{128}$ are learned parameters. The attention weights are then:

$$\alpha_{uj} = \text{softmax}_{j \in H_u}(e_{uj}),$$

and the user representation is:

$$r_u = \sum_{j \in H_u} \alpha_{uj} h_j.$$

Candidate item i is represented as $h_i = f_\theta(\tilde{a}_i)$, and the ranking score is computed by a bilinear matching head:

$$s(u, i) = r_u^\top M h_i,$$

where $M \in \mathbb{R}^{128 \times 128}$ is learned. Both r_u and h_i are L_2 -normalised before scoring.

The ranker is trained with a BPR loss on the trainable interactions of the active task, using AdamW with learning rate 10^{-3} , batch size 1,024, and up to 100 epochs with patience-3 early stopping on validation NDCG@10. All architecture choices and hyperparameters are fixed between encoders and datasets. Since all encoders are first aligned to the same 256-dimensional space, the downstream model receives inputs of identical shape and has the same number of trainable parameters in every encoder condition.

For the sequential scenario, the downstream model is SASRec [105], a self-attentive sequential recommender that consumes the same fixed item embeddings but additionally conditions on the temporal order of the user’s interaction history. We use a compact configuration maximum sequence length 50, hidden size 128, two transformer blocks, two attention heads, feed-forward dimension 256, and dropout rate 0.2. The aligned 256-dimensional item embeddings are projected into the 128-dimensional SASRec hidden space with a trainable linear layer, and learned positional embeddings are added before the self-attention blocks. Candidate items are scored in the same projected space. SASRec is trained with BPR loss using AdamW with learning rate 10^{-3} , batch size 1,024, and up to 100 epochs with patience-3 early stopping on validation NDCG@10. All SASRec hyperparameters are fixed across encoders and datasets.

4.4.1 Standard recommendation

The standard scenario evaluates how well the encoder supports retrieval under in-distribution conditions, where both users and candidate items appear during training. For each user, interactions are split chronologically into training, validation, and test portions using an 80/10/10 split.

The training interactions are used for downstream fitting, the validation interactions are used for early stopping and model selection, and the test interactions are used only for final evaluation.

4.4.2 Cold-start recommendation

The cold-start scenario tests how well the encoder represents items for which no interactions are available during downstream training. For each dataset, 5 % of the item catalog is selected as the cold-item set. All interactions involving these cold items are excluded from backbone training and from downstream ranker fitting. User representations are constructed only from non-cold training interactions. Each held-out interaction involving a cold item becomes an evaluation query.

4.4.3 Sequential recommendation

The sequential scenario evaluates how well the encoder supports order-aware prediction. For each user, interactions are sorted chronologically. The most recent interaction is held out as the test target, and the second-most recent interaction is used as the validation target. All earlier interactions form the training prefix. The maximum sequence length is fixed to 50 interactions and longer histories are truncated. Each held-out interaction becomes a next-item query in which SASRec ranks the target item.

4.4.4 Domain transfer

The domain-transfer scenario evaluates whether the embedding generalizes across categories within a single dataset. Each dataset is partitioned by a single `domain_id` field, for each dataset a single domain is selected as the target.

- **Amazon Reviews 2023** (3 domains, one per product category): the target domain is `Toys_and_Games`, the source pool is `All_Beauty` and `Amazon_Fashion`. `Toys_and_Games` is chosen because it is the most semantically distant from the other two categories, which both describe wearable or cosmetic items.
- **H&M Fashion** (14 domains, one per garment-group cluster derived from the dataset's `garment_group_name` field): the target domain is `Swimwear`, the source pool is the remaining 13 groups. `Swimwear` is chosen because it is visually distinct from typical apparel.
- **MovieLens 20M** (8 domains, one per super-genre cluster derived from the original genre tags): the target domain is `Horror/Thriller`, the source pool is the remaining seven genres. `Horror/Thriller` is chosen because it forms a relatively self-contained genre cluster whose audience does not always overlap with the rest of the catalog.

For each dataset, all interactions involving target-domain items are excluded from backbone training and from downstream fitting, the probe is therefore trained only on source-domain interactions and never sees a target-domain item during training.

4.4.5 Dataset transfer

The dataset-transfer scenario evaluates whether the recommendation-aligned signal learned by the downstream model generalizes across entirely different recommendation problems with disjoint item catalogs. Each benchmark dataset serves as the source exactly once and as the target exactly once.

- Source = Amazon Reviews 2023; target = MovieLens 20M. Tests transfer from the broadest and most varied retail source to the most semantically distant target.
- Source = H&M Fashion; target = Amazon Reviews 2023. Tests cross-retail transfer from a narrow, visual-first source to a broad target.
- Source = MovieLens 20M; target = H&M Fashion. Tests transfer from a media source with a comparatively weak content signal to a retail target where visual signal dominates.

The “Dataset tr.” column for a given dataset in the per-dataset result tables corresponds to the configuration in which that dataset is the *source*, so the column reads as how well a probe trained on this dataset transfers out.

All interactions from the target dataset are excluded from backbone training and from downstream fitting, and the trainable interaction set therefore consists only of source interactions. The two-tower probe is fit on the source dataset and applied at evaluation time to the target dataset.

4.5 Results

This section reports the zero-shot performance of the seven encoders and the baseline. All metrics are computed against a 1000-candidate ranking pool. As a reference point for interpretation, a uniformly random ranker over 999 negatives and one positive item would obtain Recall@10 of approximately 0.010 and NDCG@10 in the range 0.005. Values close to this floor should therefore be read as no useful signal. The Baseline however is not a random ranker, it uses fixed random item vectors but is paired with the same trainable downstream ranker as every other condition, so it can still encode item identity and collaborative regularities in scenarios where train and test items overlap. Its absolute level therefore depends on how much the downstream model can learn from item identity alone, and varies between datasets and scenarios.

4.5.1 Results on MovieLens 20M

Table 4.2 reports the zero-shot results on MovieLens 20M. In the standard scenario, BGE-M3 + SigLIP2 obtains the highest Recall@10, with BGE-M3 alone very close behind. Text-based encoders dominate this setting, while the vision-only models are weaker, with DINOv2 falling slightly below the baseline on Recall@10 and SigLIP2 only marginally above it.

In the cold-start scenario, the text and multimodal encoders perform relatively similarly: GTE-base obtains the highest Recall@10, followed by CLIP, BGE-M3 + SigLIP2, BGE-M3, and E5-base-v2. The vision-only encoders are noticeably weaker, with DINOv2 and SigLIP2 dropping clearly below the text and multimodal models. Cold items are excluded from downstream fitting, so the result depends almost entirely on whether the encoder’s geometry generalizes to items the ranker has never observed.

In the sequential scenario, the strongest encoders are E5-base-v2, GTE-base, BGE-M3 + SigLIP2, and BGE-M3, all within a narrow range. The vision-only encoders are noticeably weaker, consistent with movie posters carrying limited information once temporal order has to be modeled.

Domain transfer to the held-out Horror/Thriller cluster is highest for BGE-M3, followed by E5-base-v2, GTE-base, CLIP, and BGE-M3 + SigLIP2, all in a similar range. SigLIP2 still provides a noticeable signal, while DINOv2 performs very poorly. This is the only transfer column on any of the three datasets where the absolute scores rise far enough above the random-ranker floor to support encoder-level conclusions. This is partly explained by the analysis performed in Chapter 6. Dataset transfer out of MovieLens is the second case in which transfer carries usable signal, although the absolute level is much smaller. The multimodal aligned configurations

are clearly above the Baseline and the random-ranker floor, with BGE-M3 + SigLIP2 reaching 0.042/0.021, roughly four times the floor, followed by CLIP, SigLIP2, and DINOv2. The fact that only the MovieLens transfer columns produce usable signal suggests that the transfer scenarios on H&M and Amazon were defined too aggressively, and that less distant source–target pairs would likely have yielded more informative results.

■ **Table 4.2** Zero-shot results on MovieLens 20M across the five evaluation scenarios. Each cell reports Recall@10 / NDCG@10.

Encoder	Standard	Cold-start	Sequential	Domain tr.	Dataset tr.
Baseline	0.138 / 0.072	0.011 / 0.005	0.035 / 0.016	0.011 / 0.006	0.008 / 0.004
E5-base-v2	0.276 / 0.145	0.206 / 0.123	0.282 / 0.150	0.254 / 0.129	0.021 / 0.010
BGE-M3	0.295 / 0.158	0.213 / 0.096	0.255 / 0.133	0.274 / 0.149	0.024 / 0.011
GTE-base	0.285 / 0.149	0.234 / 0.112	0.273 / 0.146	0.243 / 0.122	0.019 / 0.009
DINOv2-base	0.125 / 0.071	0.055 / 0.023	0.072 / 0.035	0.112 / 0.058	0.033 / 0.016
SigLIP2-base	0.148 / 0.082	0.042 / 0.019	0.159 / 0.078	0.194 / 0.071	0.036 / 0.017
BGE-M3 + SigLIP2	0.301 / 0.158	0.219 / 0.118	0.263 / 0.137	0.212 / 0.092	0.042 / 0.021
CLIP ViT-B/16	0.267 / 0.143	0.224 / 0.111	0.185 / 0.094	0.228 / 0.111	0.038 / 0.018

4.5.2 Results on H&M Fashion

Table 4.3 reports the results on H&M Fashion. In the standard scenario, BGE-M3 + SigLIP2 produced the highest Recall@10, followed by BGE-M3 and CLIP, with SigLIP2 just slightly below.

In the cold-start scenario, the strongest encoders are BGE-M3 + SigLIP2, CLIP, and SigLIP2. The relative position of the multimodal encoders rises noticeably compared to the standard scenario, which is consistent with the visual nature of fashion items, since useful representations of unseen products rely heavily on their image content rather than on textual product type alone.

The sequential scenario is the strongest setting overall on H&M, with Recall@10 roughly doubling compared to the standard scenario. A likely explanation is that fashion purchase histories tend to be short and topically narrow which makes the next item closely tied to recent history and well suited to SASRec recency-aware modeling. BGE-M3 + SigLIP2 leads, followed by BGE-M3, SigLIP2, and E5-base-v2, all clearly above the remaining encoders. The order suggests that combining textual category and product-type signal with visual style is particularly helpful when temporal order has to be modeled in a fashion catalog.

Domain transfer to the Swimwear cluster sits at the random-ranker floor for every encoder, with values ranging from 0.013/0.006 to 0.027/0.013. The vision-oriented encoders (SigLIP2, DINOv2, CLIP) perform better, which is consistent with the visually distinctive character of swimwear, but the absolute differences are small. Dataset transfer out of H&M is at the floor across the entire column, every encoder sits in the 0.011–0.017 Recall@10 / 0.005–0.008 NDCG@10 band, including the Baseline. We therefore do not interpret the small differences between encoders as a meaningful ordering.

■ **Table 4.3** Zero-shot results on H&M Fashion across the five evaluation scenarios. Each cell reports Recall@10 / NDCG@10.

Encoder	Standard	Cold-start	Sequential	Domain tr.	Dataset tr.
Baseline	0.094 / 0.056	0.010 / 0.005	0.016 / 0.008	0.013 / 0.006	0.008 / 0.004
E5-base-v2	0.132 / 0.070	0.105 / 0.051	0.267 / 0.152	0.019 / 0.008	0.014 / 0.006
BGE-M3	0.150 / 0.081	0.109 / 0.052	0.279 / 0.163	0.014 / 0.007	0.011 / 0.005
GTE-base	0.135 / 0.072	0.096 / 0.046	0.258 / 0.147	0.015 / 0.007	0.012 / 0.006
DINOv2-base	0.120 / 0.064	0.089 / 0.045	0.214 / 0.120	0.023 / 0.012	0.013 / 0.006
SigLIP2-base	0.140 / 0.076	0.117 / 0.058	0.277 / 0.160	0.027 / 0.013	0.016 / 0.007
BGE-M3 + SigLIP2	0.164 / 0.089	0.128 / 0.061	0.303 / 0.175	0.018 / 0.009	0.015 / 0.007
CLIP ViT-B/16	0.149 / 0.080	0.122 / 0.061	0.212 / 0.114	0.021 / 0.010	0.017 / 0.008

4.5.3 Results on Amazon Reviews 2023

Table 4.4 reports the results on Amazon Reviews 2023. Absolute scores in the standard scenario are visibly lower than on the other two datasets. This is consistent with Amazon being substantially larger, broader in topic, and sparser than MovieLens or H&M, which makes general retrieval more difficult, as reflected in the very low Baseline value of 0.021/0.011. CLIP obtains the highest Recall@10 in the standard scenario, closely followed by GTE-base and BGE-M3 + SigLIP2. BGE-M3 + SigLIP2 proves to be the strongest cold-start encoder on Amazon, closely followed by CLIP and BGE-M3.

In the sequential scenario, BGE-M3 leads, followed by GTE-base and E5-base-v2. The two transfer columns on Amazon both sit at the random-ranker floor and we do not interpret encoder-level differences in either of them. Domain transfer to the held-out **Toys and Games** category ranges from 0.012/0.004 to 0.019/0.013 across encoders, including the Baseline at 0.012/0.004, so no encoder produces a usable signal. Dataset transfer out of Amazon is even tighter, with every encoder in the 0.007–0.012 Recall@10 / 0.003–0.006 NDCG@10 band.

■ **Table 4.4** Zero-shot results on Amazon Reviews 2023 across the five evaluation scenarios. Each cell reports Recall@10 / NDCG@10.

Encoder	Standard	Cold-start	Sequential	Domain tr.	Dataset tr.
Baseline	0.021 / 0.011	0.018 / 0.007	0.012 / 0.006	0.012 / 0.004	0.007 / 0.003
E5-base-v2	0.112 / 0.056	0.089 / 0.046	0.060 / 0.030	0.014 / 0.008	0.010 / 0.005
BGE-M3	0.115 / 0.057	0.095 / 0.049	0.071 / 0.041	0.019 / 0.013	0.011 / 0.005
GTE-base	0.125 / 0.063	0.089 / 0.046	0.064 / 0.033	0.012 / 0.007	0.009 / 0.004
DINOv2-base	0.074 / 0.037	0.041 / 0.021	0.033 / 0.018	0.012 / 0.005	0.008 / 0.004
SigLIP2-base	0.094 / 0.050	0.050 / 0.026	0.045 / 0.024	0.015 / 0.008	0.010 / 0.005
BGE-M3 + SigLIP2	0.121 / 0.061	0.100 / 0.052	0.054 / 0.027	0.017 / 0.009	0.012 / 0.006
CLIP ViT-B/16	0.133 / 0.068	0.098 / 0.051	0.034 / 0.017	0.019 / 0.009	0.011 / 0.005

4.5.4 Results Summary

Pretrained foundation encoders give a clear benefit over the non-semantic baseline in most in-distribution and cold-start settings, but the strongest modality depends on the dataset. MovieLens and Amazon are dominated by textual semantics, since titles, genres, brands, and product descriptions describe item identity directly. H&M on the other hand is led by visual and multimodal representations, since fashion preference depends heavily on color, cut, and style that short categorical metadata cannot fully convey.

Among text encoders, BGE-M3 is usually the strongest individual model, but the gap to E5-base-v2 and GTE-base is small enough that model size alone does not determine performance.

Among multimodal configurations, the simple BGE-M3 + SigLIP2 concatenation is the strongest standard-recommendation model on MovieLens and H&M, demonstrating that strong text and image embeddings can be combined without task-specific adaptation.

Cold-start confirms that foundation embeddings are most useful when interaction information is unavailable. The Baseline collapses to the random-ranker floor while every pretrained encoder stays above it, with the leading model differing by dataset (GTE-base on MovieLens, BGE-M3 + SigLIP2 on H&M and Amazon). The sequential scenario depends more on the catalog: H&M reaches the highest scores in the whole zero-shot table, MovieLens benefits from text encoders, and Amazon is weakest, reflecting how stable short-term preference patterns are in each catalog.

The transfer scenarios are by far the hardest, and most of them fail outright in our zero-shot setting. Out of the six transfer columns in the per-dataset tables, only two carry signal that is clearly separable from the random-ranker floor: MovieLens domain transfer to the Horror/Thriller cluster, where text encoders reach 0.243–0.274 Recall@10, and MovieLens dataset transfer (target = H&M), where multimodal encoders reach 0.033–0.042 Recall@10, three to four times the floor. A plausible explanation is that the MovieLens contains a strong interaction signal, even though the source and target domains are very different. The other four transfer columns all sit at or just above the floor for every encoder. The Baseline in those columns is itself at 0.008–0.015 Recall@10, and the best encoder is at most a few thousandths of a point higher. We therefore do not draw any conclusions from those four columns, only saying that the embeddings did not prove useful for the transfer settings.

Overall, multimodal concatenation proved to be the strongest in-domain strategy, text encoders are a reliable fallback on metadata-rich datasets, and vision becomes important when appearance drives user choice.

Fine-Tuning Multimodal Encoders

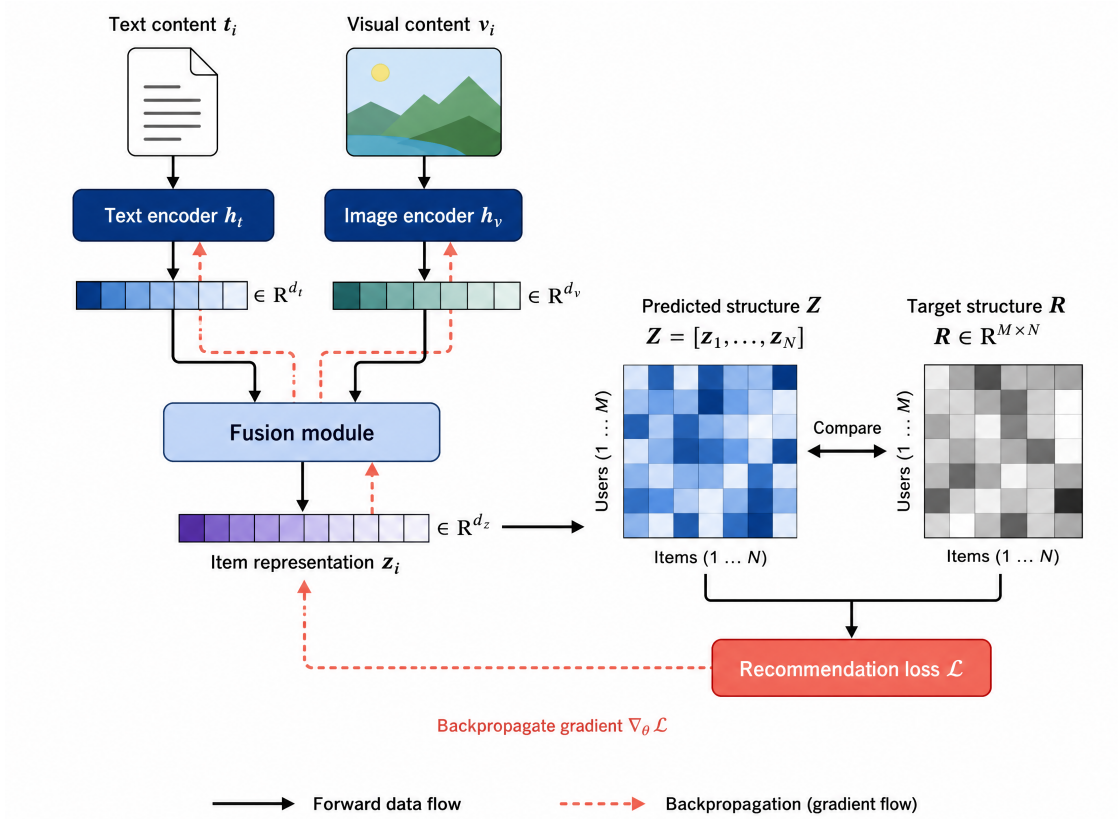
This chapter explores how the multimodal foundation encoders from Chapter 4 can be adapted to recommendation through fine-tuning on the interaction data.

5.1 Fine-Tuning on Interaction Data

The frozen encoders from Chapter 4 produce item representations that reflect content similarity, but not interaction similarity. Two items can be visually and textually close while rarely being co-interacted with, and two items can be co-interacted with while looking very differently. The goal of this chapter is to push these representations towards the second view of similarity by exposing the encoders to user-item interactions during training, while keeping the multimodal item content as the input.

The idea is illustrated in Figure 5.1. For each item i , the textual content t_i is processed by a text encoder h_t and the visual content v_i is processed by an image encoder h_v . The two modality embeddings are then combined by a fusion module into a single item representation $\mathbf{z}_i$. The matrix of all item representations $Z = [\mathbf{z}_1, \dots, \mathbf{z}_N]$ is compared against a target derived from the user-item interaction matrix R . The discrepancy between the predicted and target structure produces a recommendation loss $\mathcal{L}$, whose gradient is propagated back through the fusion module and into both encoders. In this way, a supervision signal that originates entirely in the interaction data is delivered to the parameters that govern the text and image representations.

Pretrained multimodal encoders are usually optimized to organize items according to semantic similarity: similar descriptions, similar images, or shared high-level concepts end up closer together. Recommendation, however, often depends on complementarity, popularity patterns, user intent, and domain-specific behavior that may not be directly visible in the item content. Fine-tuning on interactions therefore acts as a bridge between content-based representation learning and collaborative recommendation. The multimodal content remains the input to the encoder, but the model is updated so that its embedding outputs become more aligned with recommendation behavior. In other words, fine-tuning pushes the representation space away from purely content-based similarity and toward similarities that are more useful for downstream recommendation tasks.



■ **Figure 5.1** Overview of fine-tuning on interaction data. Item content (text and image) is encoded by two modality-specific encoders, fused into a single item embedding, and supervised by a loss derived from the user-item interaction matrix R . Gradients flow back through the fusion module into both encoders.

5.2 Experimental Setup

This chapter studies how the multimodal foundation encoders can be adapted to recommendation through fine-tuning on interaction data. The experiments focus only on the two multimodal configurations, since multimodal item representations are the main focus of this thesis. The same fine-tuning procedure could also be applied to the text-only and vision-only encoders, but they are not evaluated in this chapter.

The benchmark conditions from Chapter 4 are kept unchanged. We use the same three datasets, five evaluation scenarios with their per-task holdout rules, same candidate construction with one target item and 999 sampled negatives, and the downstream models. The fine-tuning study is organized around three design choices:

1. *Tuning method* used to adapt the pretrained encoders.
2. *Fusion strategy* used to combine modality-specific representations into a single item embedding.
3. *Training objective* used to bring interaction information to the encoder.

A full evaluation of all combinations is too expensive on the available hardware. We therefore use a staged evaluation. In each stage, one design choice is varied while the remaining choices are fixed. After the separate stages, the strongest choices are combined into a final configuration and evaluated against other fine-tuned models, and their frozen baselines.

5.2.1 Tuning Methods

The first stage compares three ways of updating the multimodal encoder: full fine-tuning, LoRA, and Houlby adapters. The purpose is to evaluate whether parameter-efficient tuning methods can adapt pretrained item representations as effectively as updating the full model. The general definitions of LoRA and adapters are introduced in Sections 1.2.1.2 and 1.2.1.3. The tuning methods are configured as follows:

- **Full fine-tuning:** all trainable parameters of the text encoder, image encoder, fusion module, and output projection head are updated.
- **LoRA:** low-rank updates with rank $r = 16$ are inserted into the query and value projection matrices of each attention layer in both the text encoder and the image encoder. The original backbone weights remain frozen, while the LoRA parameters, the fusion module, and the output projection head are updated during training.
- **Houlby adapters:** adapters with dimension $m = 16$ are inserted after each transformer layer in both encoders. The original backbone weights remain frozen, while the adapter parameters, the fusion module, and the output projection head are updated during training.

For this stage, the fusion strategy is fixed to the simplest variant, concatenation, and the training objective is fixed to ELSA.

5.2.2 Fusion Strategies

The second stage studies how text and image representations should be combined during fine-tuning. We compare content-only fusion strategies and interaction-conditioned fusion strategies. Their mathematical definitions are introduced in Section 1.3.1. The content-only fusion strategies are configured as follows:

- **Averaging:** the text and image embeddings are L_2 -normalized, projected to the same dimensionality when necessary, and averaged with equal weights.
- **Concatenation:** the text and image embeddings are L_2 -normalized and concatenated.
- **Weighted sum:** the text and image embeddings are projected to the shared fusion dimension and combined using fixed equal weights, $w_t = w_v = 0.5$.
- **Attention fusion:** the projected modality embeddings are treated as a short sequence of modality tokens and fused using a single multi-head attention block with four attention heads. A dropout rate of 0.1 and a final LayerNorm are applied.
- **Gated fusion:** the projected text and image embeddings are combined using a learned gate computed from their concatenation. The gate produces dimension-wise weights that determine how much information is taken from each modality.

We also evaluate interaction-conditioned variants of the learned fusion modules. These variants add a co-interaction context vector to the fusion module. For item i , the context vector is defined as:

$$\mathbf{c}_i = \frac{1}{|N(i)|} \sum_{j \in N(i)} \mathbf{z}_j^{\text{sg}},$$

where $N(i)$ is the set of top- k items most often co-interacted with item i in the training data, and $\mathbf{z}_j^{\text{sg}}$ is the stop-gradient fused embedding of item j from the previous epoch. We use $k = 32$ and refresh the context vectors at the start of each epoch.

For conditioned gated fusion, the gate is computed from the text embedding, image embedding, and interaction context:

$$\mathbf{g} = \sigma(W_g[P_t h_t(t_i); P_v h_v(v_i); \mathbf{c}_i]).$$

For conditioned attention fusion, $\mathbf{c}_i$ is used as an additional token in the attention block. The context vectors are computed only from the training interactions and are treated as stop-gradient inputs, so they provide interaction context without introducing an additional trainable recommender module.

Cold items, target-domain items in domain transfer, and target-dataset items in dataset transfer are by construction excluded from the training interaction graph, so they have no co-interaction neighborhood $N(i)$. For these items we set $\mathbf{c}_i$ to the zero vector at inference. For conditioned gated fusion this drives the gate input toward its content-only configuration, and for conditioned attention fusion the context token contributes no information, in both cases the module degrades gracefully toward its unconditioned counterpart while still routing through the same projection and attention parameters. The cold-start, domain-transfer, and dataset-transfer rows of the conditioned-fusion blocks should therefore be read as a fallback regime in which only the projection and attention parameters benefit from the conditioning learned during training, not as evaluations of conditioning itself on unseen items.

The fusion strategies have different assumptions depending on how the modality-specific encoders were pretrained. BGE-M3 + SigLIP2 combines two independently pretrained encoders, so its text and image embeddings are not naturally aligned before the fusion module learns to combine them. In this setting, concatenation and learned fusion strategies are expected to be safer than direct averaging, because they do not assume that the two modalities already occupy a shared coordinate system. In contrast, CLIP ViT-B/16 is pretrained with an image-text alignment objective, so its text and image embeddings already lie in a shared representation space. Averaging is therefore natural for CLIP, because it preserves the model’s native cross-modal geometry while producing a single item embedding. The tuning method is fixed to LoRA and the training objective is fixed to ELSA loss.

The interaction-conditioned fusion variants can also be interpreted as a lightweight, amortized form of meta-learning. Each item is treated as a task, its top- k co-interaction neighbors form a small support set, and the fusion module performs a single forward-pass adaptation that produces an item representation conditioned on this support set. The model is exposed to many such item-specific tasks during training, which forces the fusion parameters to encode how a support set of co-interaction neighbors should be combined with the item’s text and image content into a useful recommendation embedding. Unlike optimization-based meta-learning methods, no inner-loop gradient steps are taken at inference time. The adaptation is amortized into the fusion parameters during training.

5.2.3 Training Objectives

The third stage compares the training objectives used to adapt item embeddings to recommendation tasks. This stage includes both the basic objectives and the variants that change how item-item supervision is constructed, all of these methods define the supervision signal used during fine-tuning. We compare the following objectives:

- **BPR:** for each positive user-item interaction, one negative item is sampled. The model is trained to assign a higher score to the positive item than to the sampled negative. This is the sparsest objective considered and serves as the baseline ranking loss.
- **InfoNCE:** each positive item is contrasted against the other items in the batch using a temperature of $\tau = 0.07$.

- **ELSA:** the fine-tuned item embedding matrix E defines an item–item operator $W = EE^\top - \text{diagMat}(EE^\top)$, where the diagonal is zeroed out to prevent trivial self-reconstruction. User interaction vectors are reconstructed as $R_{\text{pred}} = RW$, and the loss compares the normalized original and reconstructed interaction vectors.
- **Interaction Jaccard similarity:** an interaction-derived similarity matching variant (Section 1.4.4) in which the predicted item–item similarity $\hat{S} = EE^\top$ is regressed against a target matrix built from the Jaccard overlap between item interaction sets. This emphasises shared users relative to the total number of users interacting with either item.
- **Interaction PMI similarity:** the same similarity matching framework, with the target built from pointwise mutual information. This reduces the dominance of globally popular items and gives more weight to unexpectedly strong co-interaction patterns.
- **Joint ELSA and InfoNCE:** item–item structural supervision and user–item discriminative supervision are combined into a single loss,

$$\mathcal{L}_{\text{joint}} = 0.5 \mathcal{L}_{\text{ELSA}} + 0.5 \mathcal{L}_{\text{InfoNCE}}.$$

ELSA and the interaction-derived similarity matching variants both supervise the encoder using item–item co-interaction structure, but in different ways: ELSA backpropagates through a reconstruction of user interaction vectors, while Jaccard and PMI compare the predicted item–item matrix against an explicit target. For this stage, the fusion strategy is fixed to concatenation and the tuning method is fixed to LoRA.

5.3 Implementation Details

The same training setup is used across all fine-tuning configurations. Backbone fine-tuning uses AdamW with a learning rate of 2×10^{-4} and weight decay of 10^{-2} . The effective batch size is 2,048 items. The learning rate follows a linear warm-up over the first 10 % of training steps, followed by cosine decay. Backbone training runs for at most 100 epochs and uses early stopping with patience 5, monitored on validation NDCG@10.

To make multimodal fine-tuning feasible within the available GPU memory, we use gradient checkpointing. During training, intermediate activations from the text and image encoders are not stored for the entire forward pass. Instead, the relevant encoder computations are recomputed during backpropagation when their gradients are needed. This trades additional computation for lower memory usage, while still allowing the recommendation loss to propagate through the fusion module and into the trainable encoder parameters, such as LoRA or adapter weights.

The downstream models that consume the fine-tuned embeddings are kept identical to those used in Chapter 4. All fine-tuning experiments are implemented as a REEF `TrainerProtocol`, described in Section 3.5.2. Each run is executed on a single NVIDIA RTX 5090 GPU with 32 GB of VRAM.

5.4 Results

This section reports the fine-tuning results for the multimodal encoders. The analysis follows the staged experimental design. We first compare tuning methods, then evaluate the effect of the fusion strategy, analyze the training objectives, and finally evaluate the combined configuration against the frozen multimodal baseline from Chapter 4.

5.4.1 Results by Tuning Method

The first stage compares full fine-tuning, LoRA, and Housby adapters on the two multimodal backbones. Frozen values are reproduced from Chapter 4 for reference.

Table 5.1 reports the results. The tuning generally tends to improve over the frozen baseline in-domain. LoRA is the strongest in-domain method on both backbones, with Housby adapters generally close behind and full fine-tuning weaker. The broad ordering is consistent across datasets.

Cold-start gains for LoRA and adapters are visible across datasets and backbones, they are smaller than the standard-scenario gains on MovieLens and H&M but larger on Amazon. Full fine-tuning, in contrast, often nudges below the frozen baseline on cold-start, likely because updating every backbone parameter on the source catalog erodes the content-only signal that cold-start ranking has to fall back on.

Sequential gains depend more on the dataset and backbone. On H&M, SASRec already extracts a strong signal from the frozen BGE-M3 + SigLIP2 embeddings, so the gain from tuning is small. With LoRA, adapters ranked better on Recall while losing on NDCG. CLIP improves more on H&M sequential, and Amazon shows visible gains for all methods because its frozen sequential baseline is much weaker.

On the domain-transfer column, MovieLens 20M reaches the highest absolute level, where LoRA and adapters go above the frozen baseline on both backbones and full fine-tuning stays roughly flat. The same ordering also appears on H&M and Amazon, just at a much smaller absolute scale: LoRA reaches the per-column maximum on both datasets and both backbones, adapters sit close to it, and full fine-tuning does not improve over baseline. The differences are small, but they are consistent enough across datasets and backbones to be read as a real trend. The dataset-transfer column, by contrast, looks too noisy to interpret.

Overall, LoRA turned out as the most reliable tuning method by far in these experiments.

5.4.2 Results by Fusion Strategy

The second stage studies how text and image embeddings should be combined during fine-tuning. The tuning method is fixed to LoRA and the training objective is fixed to ELSA loss. Table 5.2 reports the seven fusion strategies on both multimodal backbones, with the frozen rows reproduced from Chapter 4 as a reference.

On BGE-M3 + SigLIP2, fusion strategies separate into a clear ordering. Averaging is the weakest fine-tuned strategy and barely improves over the frozen baseline, which is consistent with the fact that the two encoders are not pretrained into a shared space and therefore lose information when their vectors are added directly. Concatenation and weighted sum improve in-domain results without a learned interaction between modalities, but they are matched or surpassed by the learned alternatives. Gated and attention fusion both gain over the fixed strategies, with attention fusion stronger on H&M, where product metadata and images carry complementary signals, and roughly on par with gated fusion on the more text-dominated MovieLens. Conditioning the fusion module on co-interaction context adds a further small gain on top of plain attention in standard and sequential, making conditioned attention the strongest fusion choice on this backbone for the in-domain scenarios. On the domain-transfer column, MovieLens still reaches the highest absolute level, and the same broad ordering between fusion strategies holds at a much smaller scale on H&M and Amazon: the learned fusion modules nudge above the fixed strategies, and conditioned attention reaches the per-column maximum on most cells across all three datasets. The differences are once again small in absolute terms. The dataset-transfer column, by contrast, looks too noisy to interpret.

On CLIP ViT-B/16, the picture is shifted by the fact that the two towers are already aligned. Averaging is competitive with concatenation on MovieLens, where the cross-modal geometry CLIP learned during pretraining is largely preserved, but falls behind concatenation on H&M

■ **Table 5.1** Tuning-method comparison across the five evaluation scenarios. Fusion: concatenation; objective: ELSA loss. Frozen rows are reproduced from Chapter 4. Each cell reports Recall@10 / NDCG@10.

Configuration	Standard	Cold-start	Sequential	Domain tr.	Dataset tr.
<i>MovieLens 20M</i>					
BGE-M3 + SigLIP2, Frozen	0.301 / 0.158	0.219 / 0.118	0.263 / 0.137	0.212 / 0.092	0.042 / 0.021
BGE-M3 + SigLIP2, Full FT	0.305 / 0.160	0.214 / 0.124	0.270 / 0.140	0.213 / 0.093	0.029 / 0.013
BGE-M3 + SigLIP2, LoRA	0.323 / 0.171	0.235 / 0.137	0.281 / 0.147	0.218 / 0.096	0.045 / 0.022
BGE-M3 + SigLIP2, Adapter	0.310 / 0.163	0.226 / 0.131	0.275 / 0.142	0.211 / 0.092	0.045 / 0.022
CLIP ViT-B/16, Frozen	0.267 / 0.143	0.224 / 0.111	0.185 / 0.094	0.228 / 0.111	0.038 / 0.018
CLIP ViT-B/16, Full FT	0.272 / 0.144	0.215 / 0.106	0.196 / 0.099	0.230 / 0.112	0.026 / 0.012
CLIP ViT-B/16, LoRA	0.288 / 0.154	0.236 / 0.118	0.207 / 0.105	0.234 / 0.114	0.041 / 0.019
CLIP ViT-B/16, Adapter	0.281 / 0.150	0.229 / 0.114	0.201 / 0.102	0.229 / 0.112	0.040 / 0.019
<i>HM Fashion</i>					
BGE-M3 + SigLIP2, Frozen	0.164 / 0.089	0.128 / 0.061	0.303 / 0.175	0.018 / 0.009	0.015 / 0.007
BGE-M3 + SigLIP2, Full FT	0.176 / 0.094	0.124 / 0.058	0.310 / 0.180	0.019 / 0.010	0.010 / 0.004
BGE-M3 + SigLIP2, LoRA	0.197 / 0.106	0.144 / 0.070	0.319 / 0.184	0.024 / 0.012	0.017 / 0.008
BGE-M3 + SigLIP2, Adapter	0.184 / 0.100	0.140 / 0.067	0.322 / 0.182	0.020 / 0.010	0.014 / 0.006
CLIP ViT-B/16, Frozen	0.149 / 0.080	0.122 / 0.061	0.212 / 0.114	0.021 / 0.010	0.017 / 0.008
CLIP ViT-B/16, Full FT	0.158 / 0.084	0.121 / 0.060	0.224 / 0.120	0.022 / 0.011	0.012 / 0.005
CLIP ViT-B/16, LoRA	0.183 / 0.099	0.144 / 0.073	0.236 / 0.126	0.027 / 0.013	0.021 / 0.010
CLIP ViT-B/16, Adapter	0.171 / 0.092	0.139 / 0.069	0.231 / 0.123	0.024 / 0.012	0.018 / 0.009
<i>Amazon Reviews 2023</i>					
BGE-M3 + SigLIP2, Frozen	0.121 / 0.061	0.100 / 0.052	0.054 / 0.027	0.017 / 0.009	0.012 / 0.006
BGE-M3 + SigLIP2, Full FT	0.115 / 0.058	0.107 / 0.055	0.062 / 0.031	0.018 / 0.010	0.008 / 0.004
BGE-M3 + SigLIP2, LoRA	0.139 / 0.070	0.128 / 0.069	0.068 / 0.035	0.023 / 0.012	0.015 / 0.007
BGE-M3 + SigLIP2, Adapter	0.131 / 0.065	0.116 / 0.060	0.063 / 0.031	0.020 / 0.010	0.014 / 0.007
CLIP ViT-B/16, Frozen	0.133 / 0.068	0.098 / 0.051	0.034 / 0.017	0.019 / 0.009	0.011 / 0.005
CLIP ViT-B/16, Full FT	0.127 / 0.065	0.103 / 0.053	0.041 / 0.020	0.020 / 0.010	0.007 / 0.003
CLIP ViT-B/16, LoRA	0.152 / 0.077	0.124 / 0.066	0.046 / 0.023	0.025 / 0.012	0.014 / 0.007
CLIP ViT-B/16, Adapter	0.146 / 0.073	0.114 / 0.059	0.041 / 0.021	0.022 / 0.011	0.012 / 0.006

and Amazon, where the more heterogeneous catalogs benefit from keeping both modality vectors intact. The learned fusion modules behave the same way as on BGE-M3 + SigLIP2: gated and attention fusion improve over the fixed strategies, and conditioned attention is again the strongest single choice on standard recommendation. The absolute gains from learned fusion are smaller on CLIP than on BGE-M3 + SigLIP2 on MovieLens but slightly larger on H&M and Amazon, in line with how heterogeneous each catalog is.

Comparing the two backbones, BGE-M3 + SigLIP2 stays stronger in standard and sequential evaluation on MovieLens and H&M, while on Amazon CLIP catches up and overtakes it in the standard scenario, the sequential advantage of BGE-M3 + SigLIP2 holds across all three datasets. Cold-start and standard scenarios respond to fusion changes in a similar way, suggesting that fusion reshapes the item geometry rather than just helping the downstream ranker. Sequential gains are smaller by comparison, since SASRec already exploits interaction order from the frozen embeddings.

Fusion proved to be an important design choice. Fixed strategies preserve the pretrained geometry but limit how much the model can adapt to recommendation signal. Learned fusion adds enough flexibility to re-weight modalities item by item, which matters for heterogeneous catalogs where some items are identified mainly by text and others mainly by visual appearance. On both multimodal backbones, conditioned attention is the strongest single fusion choice in our experiments.

■ **Table 5.2** Fusion-strategy comparison across the five evaluation scenarios. Backbones: BGE-M3 + SigLIP2 and CLIP ViT-B/16; tuning: LoRA; objective: ELSA loss. Frozen rows are reproduced from Chapter 4. Each cell reports Recall@10 / NDCG@10.

Configuration	Standard	Cold-start	Sequential	Domain tr.	Dataset tr.
<i>MovieLens 20M</i>					
BGE-M3 + SigLIP2, Frozen	0.301 / 0.158	0.219 / 0.118	0.263 / 0.137	0.212 / 0.092	0.042 / 0.021
BGE-M3 + SigLIP2, Averaging	0.303 / 0.159	0.221 / 0.129	0.264 / 0.138	0.211 / 0.091	0.037 / 0.018
BGE-M3 + SigLIP2, Concatenation	0.323 / 0.171	0.235 / 0.137	0.281 / 0.147	0.218 / 0.096	0.045 / 0.022
BGE-M3 + SigLIP2, Weighted sum	0.312 / 0.165	0.234 / 0.137	0.273 / 0.143	0.221 / 0.097	0.043 / 0.020
BGE-M3 + SigLIP2, Gated fusion	0.326 / 0.172	0.236 / 0.139	0.280 / 0.146	0.218 / 0.095	0.045 / 0.021
BGE-M3 + SigLIP2, Attention fusion	0.333 / 0.176	0.243 / 0.142	0.286 / 0.150	0.219 / 0.096	0.046 / 0.023
BGE-M3 + SigLIP2, Cond. gated fusion	0.328 / 0.174	0.240 / 0.140	0.282 / 0.148	0.220 / 0.096	0.048 / 0.022
BGE-M3 + SigLIP2, Cond. attention fusion	0.349 / 0.184	0.249 / 0.144	0.293 / 0.153	0.215 / 0.094	0.048 / 0.023
CLIP ViT-B/16, Frozen	0.267 / 0.143	0.224 / 0.111	0.185 / 0.094	0.228 / 0.111	0.038 / 0.018
CLIP ViT-B/16, Averaging	0.290 / 0.156	0.236 / 0.119	0.197 / 0.100	0.235 / 0.115	0.041 / 0.020
CLIP ViT-B/16, Concatenation	0.288 / 0.154	0.236 / 0.118	0.207 / 0.105	0.234 / 0.114	0.041 / 0.019
CLIP ViT-B/16, Weighted sum	0.282 / 0.151	0.234 / 0.117	0.201 / 0.103	0.237 / 0.116	0.040 / 0.019
CLIP ViT-B/16, Gated fusion	0.291 / 0.156	0.238 / 0.120	0.208 / 0.106	0.234 / 0.115	0.042 / 0.020
CLIP ViT-B/16, Attention fusion	0.297 / 0.160	0.243 / 0.123	0.213 / 0.109	0.240 / 0.118	0.044 / 0.022
CLIP ViT-B/16, Cond. gated fusion	0.294 / 0.158	0.245 / 0.124	0.210 / 0.107	0.242 / 0.119	0.043 / 0.021
CLIP ViT-B/16, Cond. attention fusion	0.302 / 0.162	0.247 / 0.125	0.215 / 0.110	0.244 / 0.120	0.045 / 0.021
<i>HM Fashion</i>					
BGE-M3 + SigLIP2, Frozen	0.164 / 0.089	0.128 / 0.061	0.303 / 0.175	0.018 / 0.009	0.015 / 0.007
BGE-M3 + SigLIP2, Averaging	0.169 / 0.091	0.130 / 0.062	0.305 / 0.176	0.016 / 0.008	0.011 / 0.005
BGE-M3 + SigLIP2, Concatenation	0.197 / 0.106	0.144 / 0.070	0.319 / 0.184	0.024 / 0.012	0.017 / 0.008
BGE-M3 + SigLIP2, Weighted sum	0.184 / 0.100	0.139 / 0.067	0.312 / 0.180	0.024 / 0.012	0.017 / 0.008
BGE-M3 + SigLIP2, Gated fusion	0.197 / 0.107	0.147 / 0.071	0.319 / 0.184	0.023 / 0.011	0.016 / 0.007
BGE-M3 + SigLIP2, Attention fusion	0.202 / 0.109	0.150 / 0.073	0.323 / 0.186	0.022 / 0.011	0.018 / 0.009
BGE-M3 + SigLIP2, Cond. gated fusion	0.199 / 0.108	0.152 / 0.074	0.321 / 0.185	0.025 / 0.012	0.017 / 0.008
BGE-M3 + SigLIP2, Cond. attention fusion	0.211 / 0.113	0.158 / 0.077	0.330 / 0.191	0.027 / 0.013	0.014 / 0.006
CLIP ViT-B/16, Frozen	0.149 / 0.080	0.122 / 0.061	0.212 / 0.114	0.021 / 0.010	0.017 / 0.008
CLIP ViT-B/16, Averaging	0.166 / 0.089	0.133 / 0.066	0.225 / 0.120	0.027 / 0.013	0.014 / 0.007
CLIP ViT-B/16, Concatenation	0.183 / 0.099	0.144 / 0.073	0.236 / 0.126	0.027 / 0.013	0.021 / 0.010
CLIP ViT-B/16, Weighted sum	0.174 / 0.094	0.139 / 0.069	0.230 / 0.123	0.028 / 0.014	0.019 / 0.009
CLIP ViT-B/16, Gated fusion	0.183 / 0.099	0.144 / 0.072	0.237 / 0.126	0.027 / 0.013	0.021 / 0.010
CLIP ViT-B/16, Attention fusion	0.188 / 0.102	0.148 / 0.074	0.241 / 0.129	0.029 / 0.014	0.023 / 0.011
CLIP ViT-B/16, Cond. gated fusion	0.184 / 0.100	0.149 / 0.075	0.238 / 0.127	0.030 / 0.015	0.022 / 0.011
CLIP ViT-B/16, Cond. attention fusion	0.193 / 0.104	0.151 / 0.076	0.243 / 0.131	0.029 / 0.014	0.022 / 0.010
<i>Amazon Reviews 2023</i>					
BGE-M3 + SigLIP2, Frozen	0.121 / 0.061	0.100 / 0.052	0.054 / 0.027	0.017 / 0.009	0.012 / 0.006
BGE-M3 + SigLIP2, Averaging	0.123 / 0.062	0.107 / 0.056	0.057 / 0.029	0.020 / 0.010	0.009 / 0.004
BGE-M3 + SigLIP2, Concatenation	0.139 / 0.070	0.128 / 0.069	0.068 / 0.035	0.023 / 0.012	0.015 / 0.007
BGE-M3 + SigLIP2, Weighted sum	0.137 / 0.069	0.118 / 0.063	0.063 / 0.032	0.023 / 0.012	0.014 / 0.007
BGE-M3 + SigLIP2, Gated fusion	0.140 / 0.070	0.126 / 0.068	0.068 / 0.035	0.023 / 0.011	0.013 / 0.007
BGE-M3 + SigLIP2, Attention fusion	0.144 / 0.073	0.130 / 0.071	0.071 / 0.036	0.025 / 0.012	0.015 / 0.007
BGE-M3 + SigLIP2, Cond. gated fusion	0.142 / 0.072	0.127 / 0.069	0.069 / 0.035	0.026 / 0.013	0.014 / 0.007
BGE-M3 + SigLIP2, Cond. attention fusion	0.154 / 0.077	0.139 / 0.075	0.076 / 0.039	0.028 / 0.013	0.018 / 0.008
CLIP ViT-B/16, Frozen	0.133 / 0.068	0.098 / 0.051	0.034 / 0.017	0.019 / 0.009	0.011 / 0.005
CLIP ViT-B/16, Averaging	0.146 / 0.074	0.115 / 0.061	0.041 / 0.020	0.024 / 0.012	0.011 / 0.005
CLIP ViT-B/16, Concatenation	0.152 / 0.077	0.124 / 0.066	0.046 / 0.023	0.025 / 0.012	0.014 / 0.007
CLIP ViT-B/16, Weighted sum	0.151 / 0.077	0.117 / 0.062	0.043 / 0.022	0.025 / 0.012	0.012 / 0.005
CLIP ViT-B/16, Gated fusion	0.154 / 0.078	0.124 / 0.066	0.046 / 0.023	0.026 / 0.013	0.014 / 0.007
CLIP ViT-B/16, Attention fusion	0.156 / 0.080	0.128 / 0.069	0.049 / 0.025	0.027 / 0.013	0.015 / 0.007
CLIP ViT-B/16, Cond. gated fusion	0.158 / 0.081	0.125 / 0.067	0.047 / 0.024	0.028 / 0.014	0.014 / 0.006
CLIP ViT-B/16, Cond. attention fusion	0.157 / 0.080	0.132 / 0.071	0.051 / 0.026	0.027 / 0.013	0.016 / 0.008

5.4.3 Results by Training Objective

The third stage compares the supervision signal used during fine-tuning. Table 5.3 reports the six objectives on both multimodal backbones, BGE-M3 + SigLIP2 and CLIP ViT-B/16. The fusion strategy is fixed to concatenation and the tuning method is fixed to LoRA.

On BGE-M3 + SigLIP2, the standalone objectives form a clear ordering. BPR is the weakest and barely improves over the frozen baseline and in some instances performs worse, since its sparse pairwise signal does not constrain the global geometry of the item space. InfoNCE is stronger, because the in-batch negatives provide a denser discriminative signal. The item–item objectives improve further: Jaccard introduces structural supervision but its normalization softens the strongest recommendation signals, leaving it slightly behind PMI, which down-weights globally popular items and gives more weight to informative co-interactions. ELSA is the strongest objective in the table, holding the column maximum on standard, cold-start, and sequential across all three datasets.

The joint ELSA + InfoNCE objective behaves very poorly. Standard, cold-start, and sequential Recall@10 fall well below the frozen baseline. The two losses appear to pull in different directions for the optimizer when the fusion module is a fixed concatenation, leaving the encoder under-trained on both signals.

The same broad ordering holds for CLIP ViT-B/16, with BPR weakest, InfoNCE intermediate, Jaccard and PMI close, and ELSA the strongest, although the relative differences between objectives are smaller. The joint-loss collapse is even more pronounced on CLIP than on BGE-M3 + SigLIP2: the smaller text encoder likely gives the optimizer less capacity to satisfy two competing supervision signals, and standard Recall@10 falls by roughly 20% below the frozen baseline on Amazon and H&M. Under ELSA, CLIP overtakes BGE-M3 + SigLIP2 in cold-start on H&M and in standard recommendation on Amazon, while BGE-M3 + SigLIP2 retains its advantage elsewhere. The two transfer columns behave the same way as in the previous stages. On domain transfer, MovieLens reaches the highest absolute level, and the same objective ordering still appears on H&M and Amazon, just at a smaller scale.

We conclude, that the objective matters as much as the tuning mechanism. Sparse pairwise supervision improves in-domain ranking only marginally, item–item objectives align better with the goal of adapting a foundation encoder, and ELSA is the most reliable target across the in-domain scenarios. Combining structural item–item supervision with user–item contrast through a joint loss did not work well in this setup with a fixed concatenation fusion and an equal 0.5/0.5 weighting: the two signals destabilize the optimization and degrade in-domain ranking sharply. Different relative weights between the two losses, or a curriculum that anneals one signal relative to the other, may make the joint objective more competitive.

■ **Table 5.3** Training-objective comparison across the five evaluation scenarios. Backbones: BGE-M3 + SigLIP2 and CLIP ViT-B/16; fusion: concatenation; tuning: LoRA. ELSA uses the reconstruction loss; Interaction Jaccard and Interaction PMI are similarity matching variants that differ only in the construction of the target item-item similarity matrix. The joint objective combines ELSA reconstruction with InfoNCE. Frozen rows are reproduced from Chapter 4. Each cell reports Recall@10 / NDCG@10.

Configuration	Standard	Cold-start	Sequential	Domain tr.	Dataset tr.
<i>MovieLens 20M</i>					
BGE-M3 + SigLIP2, Frozen	0.301 / 0.158	0.219 / 0.118	0.263 / 0.137	0.212 / 0.092	0.042 / 0.021
BGE-M3 + SigLIP2, BPR	0.297 / 0.155	0.215 / 0.124	0.262 / 0.135	0.207 / 0.090	0.030 / 0.014
BGE-M3 + SigLIP2, InfoNCE	0.314 / 0.166	0.226 / 0.131	0.275 / 0.143	0.211 / 0.093	0.043 / 0.021
BGE-M3 + SigLIP2, Interaction Jaccard	0.316 / 0.167	0.228 / 0.133	0.276 / 0.144	0.213 / 0.094	0.043 / 0.020
BGE-M3 + SigLIP2, Interaction PMI	0.319 / 0.169	0.231 / 0.135	0.279 / 0.146	0.214 / 0.094	0.044 / 0.022
BGE-M3 + SigLIP2, ELSA	0.323 / 0.171	0.235 / 0.137	0.281 / 0.147	0.218 / 0.096	0.045 / 0.022
BGE-M3 + SigLIP2, ELSA + InfoNCE	0.245 / 0.128	0.182 / 0.108	0.221 / 0.116	0.183 / 0.080	0.034 / 0.017
CLIP ViT-B/16, Frozen	0.267 / 0.143	0.224 / 0.111	0.185 / 0.094	0.228 / 0.111	0.038 / 0.018
CLIP ViT-B/16, BPR	0.262 / 0.139	0.218 / 0.108	0.183 / 0.092	0.221 / 0.107	0.024 / 0.011
CLIP ViT-B/16, InfoNCE	0.279 / 0.149	0.229 / 0.114	0.198 / 0.101	0.228 / 0.111	0.037 / 0.017
CLIP ViT-B/16, Interaction Jaccard	0.281 / 0.150	0.230 / 0.115	0.201 / 0.102	0.229 / 0.112	0.038 / 0.018
CLIP ViT-B/16, Interaction PMI	0.285 / 0.152	0.232 / 0.116	0.205 / 0.104	0.231 / 0.113	0.039 / 0.018
CLIP ViT-B/16, ELSA	0.288 / 0.154	0.236 / 0.118	0.207 / 0.105	0.234 / 0.114	0.041 / 0.019
CLIP ViT-B/16, ELSA + InfoNCE	0.218 / 0.119	0.183 / 0.094	0.152 / 0.078	0.193 / 0.096	0.029 / 0.014
<i>H&M Fashion</i>					
BGE-M3 + SigLIP2, Frozen	0.164 / 0.089	0.128 / 0.061	0.303 / 0.175	0.018 / 0.009	0.015 / 0.007
BGE-M3 + SigLIP2, BPR	0.169 / 0.091	0.124 / 0.058	0.302 / 0.174	0.016 / 0.008	0.011 / 0.005
BGE-M3 + SigLIP2, InfoNCE	0.185 / 0.100	0.135 / 0.065	0.310 / 0.179	0.020 / 0.010	0.015 / 0.007
BGE-M3 + SigLIP2, Interaction Jaccard	0.188 / 0.102	0.137 / 0.066	0.313 / 0.180	0.021 / 0.010	0.015 / 0.007
BGE-M3 + SigLIP2, Interaction PMI	0.193 / 0.105	0.142 / 0.069	0.317 / 0.183	0.022 / 0.011	0.016 / 0.008
BGE-M3 + SigLIP2, ELSA	0.197 / 0.106	0.144 / 0.070	0.319 / 0.184	0.024 / 0.012	0.017 / 0.008
BGE-M3 + SigLIP2, ELSA + InfoNCE	0.132 / 0.072	0.105 / 0.052	0.255 / 0.148	0.022 / 0.011	0.016 / 0.008
CLIP ViT-B/16, Frozen	0.149 / 0.080	0.122 / 0.061	0.212 / 0.114	0.021 / 0.010	0.017 / 0.008
CLIP ViT-B/16, BPR	0.154 / 0.082	0.119 / 0.058	0.215 / 0.115	0.018 / 0.008	0.012 / 0.005
CLIP ViT-B/16, InfoNCE	0.171 / 0.092	0.134 / 0.067	0.225 / 0.121	0.024 / 0.012	0.018 / 0.009
CLIP ViT-B/16, Interaction Jaccard	0.174 / 0.094	0.137 / 0.069	0.229 / 0.123	0.025 / 0.012	0.019 / 0.009
CLIP ViT-B/16, Interaction PMI	0.179 / 0.097	0.141 / 0.071	0.234 / 0.125	0.026 / 0.013	0.020 / 0.010
CLIP ViT-B/16, ELSA	0.183 / 0.099	0.144 / 0.073	0.236 / 0.126	0.027 / 0.013	0.021 / 0.010
CLIP ViT-B/16, ELSA + InfoNCE	0.122 / 0.066	0.099 / 0.049	0.176 / 0.095	0.026 / 0.013	0.021 / 0.010
<i>Amazon Reviews 2023</i>					
BGE-M3 + SigLIP2, Frozen	0.121 / 0.061	0.100 / 0.052	0.054 / 0.027	0.017 / 0.009	0.012 / 0.006
BGE-M3 + SigLIP2, BPR	0.116 / 0.058	0.099 / 0.051	0.054 / 0.027	0.014 / 0.007	0.008 / 0.004
BGE-M3 + SigLIP2, InfoNCE	0.130 / 0.065	0.114 / 0.061	0.063 / 0.032	0.020 / 0.010	0.012 / 0.006
BGE-M3 + SigLIP2, Interaction Jaccard	0.133 / 0.067	0.118 / 0.063	0.064 / 0.032	0.021 / 0.011	0.016 / 0.006
BGE-M3 + SigLIP2, Interaction PMI	0.135 / 0.068	0.124 / 0.070	0.067 / 0.034	0.022 / 0.011	0.013 / 0.006
BGE-M3 + SigLIP2, ELSA	0.139 / 0.070	0.128 / 0.069	0.068 / 0.035	0.023 / 0.012	0.015 / 0.007
BGE-M3 + SigLIP2, ELSA + InfoNCE	0.097 / 0.049	0.082 / 0.043	0.043 / 0.022	0.022 / 0.011	0.013 / 0.007
CLIP ViT-B/16, Frozen	0.133 / 0.068	0.098 / 0.051	0.034 / 0.017	0.019 / 0.009	0.011 / 0.005
CLIP ViT-B/16, BPR	0.127 / 0.064	0.097 / 0.050	0.034 / 0.017	0.017 / 0.008	0.007 / 0.003
CLIP ViT-B/16, InfoNCE	0.145 / 0.074	0.114 / 0.060	0.041 / 0.020	0.023 / 0.011	0.012 / 0.006
CLIP ViT-B/16, Interaction Jaccard	0.146 / 0.075	0.117 / 0.062	0.043 / 0.021	0.023 / 0.011	0.012 / 0.006
CLIP ViT-B/16, Interaction PMI	0.149 / 0.076	0.121 / 0.064	0.045 / 0.022	0.026 / 0.014	0.013 / 0.006
CLIP ViT-B/16, ELSA	0.152 / 0.077	0.124 / 0.066	0.046 / 0.023	0.025 / 0.012	0.014 / 0.007
CLIP ViT-B/16, ELSA + InfoNCE	0.108 / 0.055	0.078 / 0.041	0.026 / 0.013	0.024 / 0.012	0.014 / 0.007

5.4.4 Best-Performing Configuration

The final evaluation combines the strongest choices from the staged experiments into a single BGE-M3 + SigLIP2 configuration. We pick BGE-M3 + SigLIP2 over CLIP because it wins standard recommendation on MovieLens and H&M and is the strongest sequential model on all three datasets, while CLIP only overtakes it on Amazon standard recommendation. The configuration uses LoRA as the tuning method, interaction-conditioned attention as the fusion module, and the ELSA loss as the training objective.

Table 5.4 compares the final configuration against the frozen BGE-M3 + SigLIP2 baseline. The fine-tuned model improves on the in-domain scenarios across all three datasets. MovieLens shows moderate gains in standard, cold-start, and sequential, consistent with a frozen baseline that is already strong. H&M sees larger relative gains in standard and cold-start, where learned multimodal fusion aligns product metadata with visual appearance. Amazon sees the largest relative in-domain gains overall, in line with its weak frozen baseline and broad, sparse catalog. Sequential gains are smaller on H&M because SASRec already extracts most of the available behavioral signal from the frozen embeddings.

The transfer columns seem to behave randomly. Four of the six cross-catalog and cross-domain cells (H&M domain, H&M dataset, Amazon domain, Amazon dataset) sit at the random-ranker floor for both the frozen and the fine-tuned model. The two MovieLens transfer columns are the only ones where the absolute level is high enough to support comparison: MovieLens dataset transfer improves modestly under fine-tuning, and MovieLens domain transfer is essentially flat. These findings further support the idea that the transfer scenarios were defined too challengingly, and that simpler source-target pairs might have produced more interpretable results.

■ **Table 5.4** Final BGE-M3 + SigLIP2 fine-tuned configuration evaluated against the frozen BGE-M3 + SigLIP2 baseline across the five evaluation scenarios. The final configuration uses LoRA, interaction-conditioned attention fusion, and the ELSA objective. Each cell reports Recall@10 / NDCG@10; the relative-gain row is computed cell-wise from the two preceding rows.

Configuration	Standard	Cold-start	Sequential	Domain tr.	Dataset tr.
<i>MovieLens 20M</i>					
Frozen	0.301 / 0.158	0.219 / 0.118	0.263 / 0.137	0.212 / 0.092	0.042 / 0.021
Best fine-tuned	0.349 / 0.184	0.249 / 0.144	0.293 / 0.153	0.215 / 0.094	0.048 / 0.023
Relative gain	+15.9 % / +16.5 %	+13.7 % / +12.5 %	+11.4 % / +11.7 %	+1.4 % / +2.2 %	+14.3 % / +9.5 %
<i>H&M Fashion</i>					
Frozen	0.164 / 0.089	0.128 / 0.061	0.303 / 0.175	0.018 / 0.009	0.015 / 0.007
Best fine-tuned	0.211 / 0.113	0.158 / 0.077	0.330 / 0.191	0.027 / 0.013	0.014 / 0.006
Relative gain	+28.7 % / +27.0 %	+23.4 % / +26.2 %	+8.9 % / +9.1 %	+50.0 % / +44.4 %	−6.7 % / −14.3 %
<i>Amazon Reviews 2023</i>					
Frozen	0.121 / 0.061	0.100 / 0.052	0.054 / 0.027	0.017 / 0.009	0.012 / 0.006
Best fine-tuned	0.154 / 0.077	0.139 / 0.075	0.076 / 0.039	0.028 / 0.013	0.018 / 0.008
Relative gain	+27.3 % / +26.2 %	+39.0 % / +44.2 %	+40.7 % / +44.4 %	+64.7 % / +44.4 %	+50.0 % / +33.3 %

5.4.5 Summary of Results

The results show that multimodal foundation encoders can be adapted to recommendation using interaction data, and that fine-tuning is worth the additional cost in our setup. Across all three datasets, the fine-tuned models improve over the frozen baselines on the in-domain scenarios (standard, cold-start, sequential), and these gains hold for both multimodal backbones.

The best configuration combines BGE-M3 + SigLIP2 with LoRA, interaction-conditioned attention fusion, and the ELSA reconstruction loss. Each of these choices addresses a different

limitation of the frozen baseline. LoRA preserves the pretrained backbone while still allowing the encoder to adapt, which avoids the over-fitting and damage to the pretrained geometry that we saw under full fine-tuning. Interaction-conditioned attention provides the model with a learned, per-item way of combining text and image information, which matters most when the two encoders are pretrained independently and the catalog is heterogeneous. ELSA supervises the encoder through an item-item operator that reconstructs user interaction vectors, aligning the embedding geometry with collaborative behavior while keeping enough of the pretrained semantic structure to also help cold-start. Sparse pairwise supervision and the joint ELSA + InfoNCE loss were both clearly weaker, the former because it does not constrain the global geometry of the item space, the latter because the two signals destabilize the optimization under fixed concatenation fusion.

The transfer scenarios show a more limited but still consistent picture. On domain transfer, MovieLens reaches the highest absolute level, and the same per-method, per-fusion, and per-objective orderings reappear on H&M and Amazon at a much smaller scale. The dataset-transfer column is the one place where the experiments do not produce usable signal: values are tightly compressed across datasets, backbones, and methods, and the per-method ordering is inconsistent.

Overall, the chapter shows that interaction-based fine-tuning makes multimodal item representations more useful for in-domain recommendation, and that the combination of LoRA, interaction-conditioned fusion, and ELSA gives consistent gains across datasets and scenarios. Domain transfer follows the same pattern as the in-domain scenarios at a smaller scale, while dataset transfer would benefit from less aggressively defined source-target pairs in future work.

Analysis of Multimodal Embedding Spaces

The previous two chapters explored whether pretrained multimodal encoders can be used as backbones for downstream recommenders, and whether fine-tuning on interaction data can improve their embeddings for recommendation. The results showed that frozen pretrained embeddings already contain useful recommendation signal, but that interaction-based fine-tuning further improves performance across datasets and scenarios.

In this chapter, we analyze how the embedding space changes after fine-tuning. The goal is to understand what happens to the learned representations when the model is trained with recommendation data. We compare the best-performing model, BGE-M3 + SigLIP2, in its frozen and final fine-tuned, best performing variant. We perform three analyses. First, we look at how the global structure of the embedding space changes before and after fine-tuning. Second, we analyze the local structure of the embedding space by examining how item neighborhoods change. Third, we inspect the attention layers of the text and image encoders to see whether fine-tuning changes their internal attention patterns.

6.1 Global Embedding Space Structure

The analysis looks at how the embedding space changes before and after fine-tuning. We use a UMAP projection of the item embeddings, applied separately to the frozen and fine-tuned variants of BGE-M3 + SigLIP2. Figure 6.1 shows UMAP projections of the BGE-M3 + SigLIP2 item embeddings on all three datasets, with the frozen variant on the left and the fine-tuned variant on the right. The broad categorical structure is preserved after fine-tuning. At the same time, fine-tuning changes the global layout of the embedding space. The clusters become more compact and are brought closer together, suggesting that the model reduces some of the separation created by the original pretrained representation. This contraction is consistent with how the model is supervised, the interaction-based objective pulls co-interacted items together regardless of their content category, so directions that do not carry recommendation signal are compressed, while directions that separate items along behavior are preserved. The effect is visible across all three datasets, but with different intensities. On H&M Fashion, the group clusters remain visually distinct, but neighboring categories move closer. On MovieLens 20M, the genre boundaries blur more visibly. On Amazon Reviews 2023, the broad separation between top-level product categories is preserved, while the within-category sub-clusters become more compact.

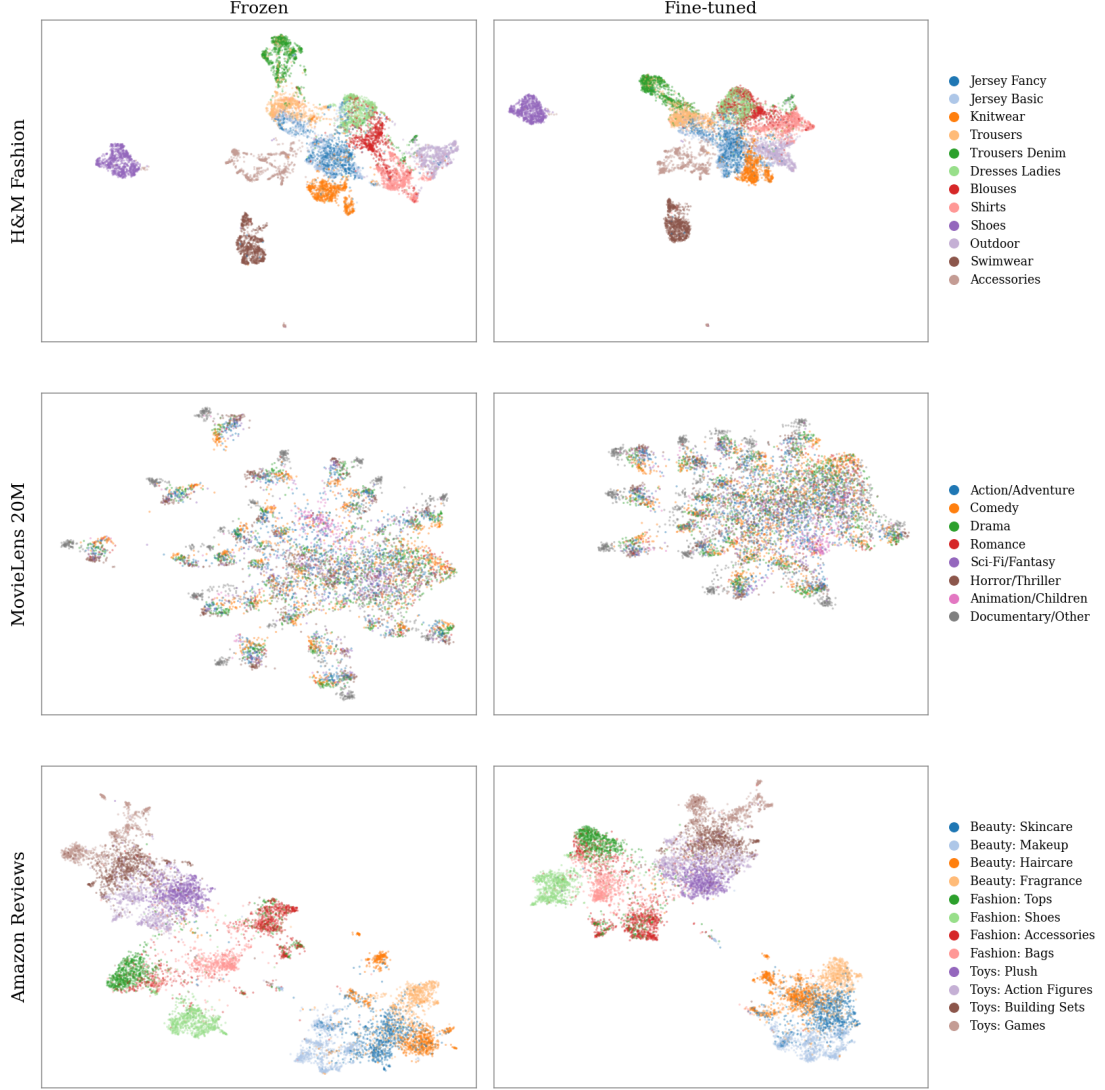


Figure 6.1 UMAP projection of BGE-M3 + SigLIP2 item embeddings before (left) and after (right) fine-tuning. We are showing a sub-sample of 20 000 random items. Rows correspond to H&M Fashion, MovieLens 20M, and Amazon Reviews 2023. Each color denotes one category, with the legend listed beside the fine-tuned panel.

The UMAP projections also help explain the domain-transfer behavior reported in Chapters 4 and 5. The held-out target domains are disjoint from the source domains on H&M Fashion and Amazon Reviews 2023, but heavily mixed with the source on MovieLens 20M. On H&M Fashion, the held-out Swimwear cluster sits clearly apart from the rest of the catalog in both the frozen and fine-tuned projections, with little overlap with adjacent groups. On Amazon Reviews 2023, the Toys and Games sub-clusters occupy a separate region from the Beauty and Fashion categories, again with limited overlap. On MovieLens 20M, by contrast, the genre clusters are strongly intertwined, the Horror/Thriller items overlap substantially with other genres.

6.2 Local Neighborhood Case Examples

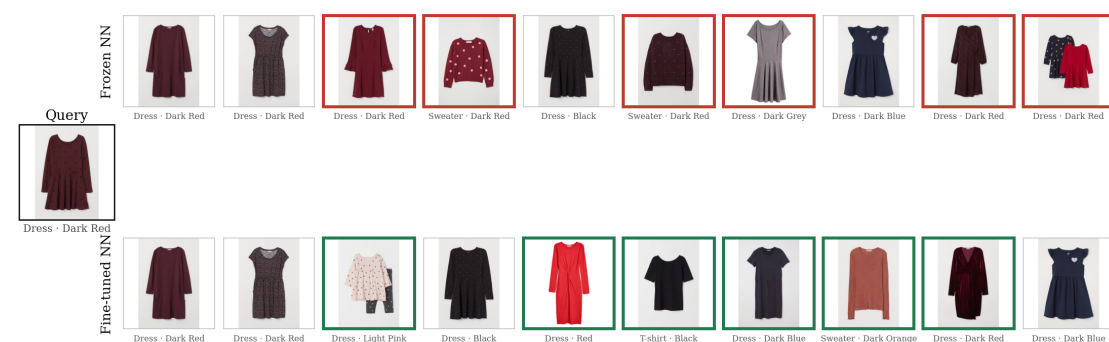
This section explores how the immediate neighborhood of a query item changes after fine-tuning. For each model, we retrieve the ten nearest neighbors of an item under cosine similarity.

Figures 6.2–6.4 show nearest neighbor examples before and after fine-tuning for the selected query items. The frozen model usually retrieves items that are visually or categorically very close to the query. For example, shirts are matched with other shirts, dresses with other dresses, and lipsticks with other lipstick products. This shows that the pretrained embedding space already captures strong content similarity.

After fine-tuning, the local neighborhood changes. The closest items often remain similar to the query, but we can also see new items appearing further down the ranked list. These items are not always from the exact same visual category, but they are often plausible recommendation candidates. For example, the men’s shirt query starts to retrieve trousers, a hoodie, and a belt, which might be useful in the same outfit context. Similarly, the dress example retrieves other clothing items that are less visually identical but still relevant in a fashion recommendation setting. In the Amazon examples, the fine-tuned model keeps similar products near the top, while also moving related cosmetic or personal-care items closer to the query. This suggests that fine-tuning adjusts the embedding space toward recommendation behavior.



(a) Query: men’s purple check button-up shirt.



(b) Query: women’s dark red short-sleeve dress.

Figure 6.2 Top-ten nearest neighbors on H&M Fashion under the frozen (top row of each panel) and fine-tuned (bottom row) BGE-M3 + SigLIP2 embedding space. The query item is shown on the left of each panel. Red frames mark items present in the frozen neighborhood that are dropped after fine-tuning; green frames mark items newly introduced by the fine-tuned model.

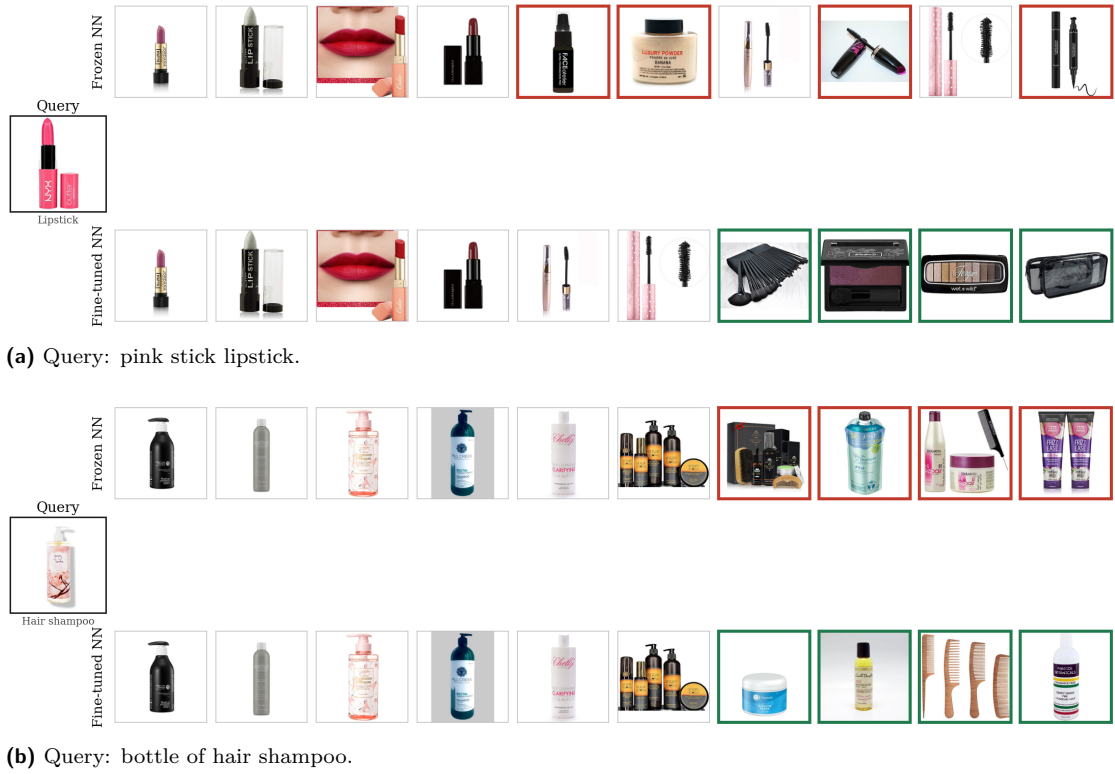
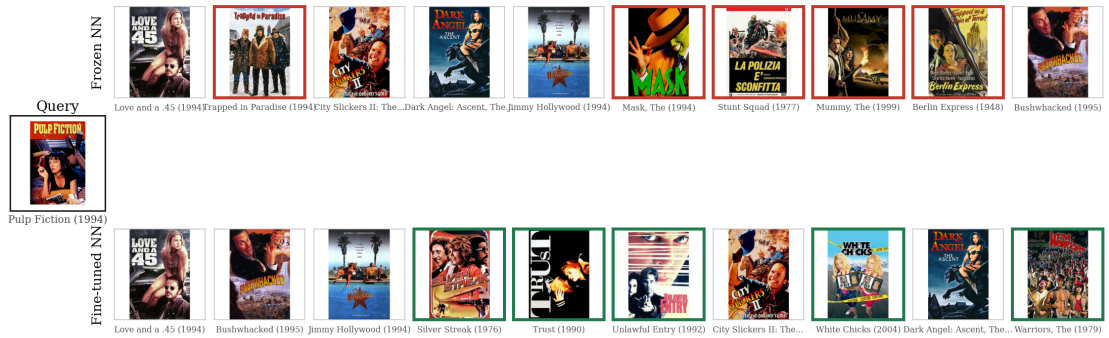


Figure 6.3 Top-ten nearest neighbors on Amazon Reviews 2023 (Beauty category) under the frozen (top row of each panel) and fine-tuned (bottom row) BGE-M3 + SigLIP2 embedding space. The query item is shown on the left of each panel. Red frames mark items present in the frozen neighborhood that are dropped after fine-tuning; green frames mark items newly introduced by the fine-tuned model.



(a) Query: *Pulp Fiction* (1994), a cult crime drama.

Figure 6.4 Top-ten nearest neighbors on MovieLens 20M under the frozen (top row) and fine-tuned (bottom row) BGE-M3 + SigLIP2 embedding space. The query item is shown on the left. Red frames mark items present in the frozen neighborhood that are dropped after fine-tuning; green frames mark items newly introduced by the fine-tuned model.

6.3 Attention Preservation in Fine-Tuned Encoders

The previous two sections examined the embedding spaces produced by the encoders. This section analyzes the internal attention mechanisms of the text and vision encoders. For the text model, we visualize token-level attention directly, for the image model, we propose the algorithm described below.

6.3.1 Algorithm for Analyzing Attention Behavior

We propose an algorithm with the following steps: attention extraction, normalization, clustering, and visualization.

Attention Extraction

For an input image I , a Vision Transformer computes a collection of attention matrices across layers and heads:

$$\left\{ \mathbf{A}_h^{(l)} \in \mathbb{R}^{N \times N} \right\}_{l=1, h=1}^{L, H},$$

where L denotes the number of transformer layers, H represents the number of attention heads per layer, and N is the sequence length corresponding to the number of image patches, excluding the [CLS] token. To form these patches, the original image is divided into a regular grid G of size $\sqrt{N} \times \sqrt{N}$. Each attention matrix $\mathbf{A}_h^{(l)}$ is a row-stochastic matrix in which the entry a_{ij} represents the fraction of attention that patch i allocates to patch j at layer l under head h .

Since the final transformer layer typically encapsulates the richest semantic information our analysis focuses on the attention matrices from layer L . To reduce the dimensionality of the attention representation from H separate matrices to a single matrix, we average over the H attention heads:

$$\bar{\mathbf{A}}^{(L)} = \frac{1}{H} \sum_{h=1}^H \mathbf{A}_h^{(L)}.$$

Normalization of Attention Maps

In self-attention, each row of the attention matrix corresponds to the distribution of attention weights for a specific query patch over all key patches. Because the softmax normalization in the attention computation produces distributions that sum to one, the absolute magnitude of individual attention weights depends on the entropy of the distribution: a highly focused attention pattern (low entropy) will have a few large weights and many small ones, while a diffuse pattern (high entropy) will have uniformly moderate weights. To ensure a consistent comparison across patches with different attention entropy levels, each row is normalized by its maximum value:

$$\tilde{a}_{ij} = \frac{a_{ij}}{\max_k a_{ik}}, \quad \forall j.$$

The normalized matrix $\tilde{\mathbf{A}}^{(L)}$ is the input to the subsequent clustering stage.

Clustering of Attention Patterns

To identify grouped regions within the attention map we apply k-means clustering directly to the normalized attention matrix $\tilde{\mathbf{A}}^{(L)}$. Each row $\tilde{a}_i$ is treated as a feature vector describing the attention profile of patch i , and the clustering algorithm groups patches whose attention profiles are similar into k clusters. Formally, the clustering assigns each patch $g \in G$ to a cluster label c :

$$\forall g \in G, \quad \mathcal{C}(g) = c, \quad c \in \{1, 2, \dots, k\}.$$

Patches that attend to similar regions of the image are grouped together, forming clusters that correspond to semantically meaningful image regions. The number of clusters k is set manually based on visual inspection of the input image, with typical values ranging from 3 to 6 depending on the complexity of the scene.

Visualization

To visualize the attention distribution, the clustered patch grid is overlaid onto the preprocessed input image. Each patch is overlaid with a color according to its cluster assignment.

6.3.2 Comparison of Attention Patterns

Figures 6.5 and 6.6 show the attention patterns of the zero-shot and fine-tuned models. Overall, the attention patterns remain very similar after fine-tuning. In the text tower, the same important tokens receive the strongest attention. The model continues to focus on descriptive tokens such as item type, color, and material words. For example, words such as *dress*, *black*, *leather*, *trousers*, and similar item descriptors remain highlighted in both variants. In MovieLens, the model mainly attends to movie titles, years, and genre tokens. The same behavior is visible after fine-tuning. The same pattern can be seen in the vision tower. On H&M, the attention regions remain aligned with the main product area, such as the dress, sneaker, shirt, bag, and trousers. The attention still follows the object shape.

6.4 Summary of Findings

This chapter analyzed how the BGE-M3 + SigLIP2 embedding space changes after fine-tuning on recommendation data. The goal was to understand what kind of representational change happens inside the model. We looked at the global embedding structure, local nearest-neighbor changes, and attention patterns in the text and image encoders.

The global analysis showed that fine-tuning changes the embedding space, but does not destroy its original structure. In the UMAP projections, the broad category layout remains visible after fine-tuning. At the same time, the fine-tuned embeddings become more compact and the clusters move closer together. This suggests that the model keeps the semantic organization learned during pretraining, while adapting the space toward recommendation behavior.

The local neighborhood examples show this change more clearly. In the frozen model, nearest neighbors are mostly items that are visually or categorically very similar to the query. Shirts are close to other shirts, dresses to other dresses, and lipstick products to other lipstick products. After fine-tuning, the nearest neighbors still preserve much of this content similarity, especially at the top of the ranked list. However, further down the list, new items appear that are not always the same item type, but are often useful in the same recommendation context.

The attention analysis shows that this adaptation does not come from destroying the internal behavior of the encoders. In the text tower, the same important metadata tokens remain highlighted before and after fine-tuning.

Overall, the analysis suggests that fine-tuning acts as a controlled adaptation of the pre-trained embedding space. The model preserves the broad semantic structure and attention patterns learned during pretraining, while changing the final item representations in a way that better matches recommendation data. The global structure becomes more compact, local neighborhoods become more recommendation-oriented, and the internal attention mechanisms remain largely preserved. This supports the result from the previous chapter: the frozen embeddings already contain useful signal, but fine-tuning makes them more aligned to recommendation.

Patterned dress

	ag	uil	era	maxi	dre	ss	dress	light	beige	jersey	basic	strap	less	maxi
Frozen	dress	in	jersey	with	an	elastic	ated	seam	at	the	waist	and	slits	in
	the	sides	.	integral	top	with	elastic	ation	at	the	top	.		
	ag	uil	era	maxi	dre	ss	dress	light	beige	jersey	basic	strap	less	maxi
Fine-tuned	dress	in	jersey	with	an	elastic	ated	seam	at	the	waist	and	slits	in
	the	sides	.	integral	top	with	elastic	ation	at	the	top	.		

Lace-up sneaker

Frozen	mal	ken	lace	-	up	sneakers	black	shoes	trainers	with	lac	ing	at	the
	front	,	mesh	lining	s	and	ins	oles	and	rubber	sole	s	.	
Fine-tuned	mal	ken	lace	-	up	sneakers	black	shoes	trainers	with	lac	ing	at	the
	front	,	mesh	lining	s	and	ins	oles	and	rubber	sole	s	.	

Check shirt

Frozen	bryn	fl	ane	l	check	shirt	dark	purple	shirts	shirt	in	checked	fl	anne	l
	with	a	collar	and	chest	pocket	.	regular	fit	.					
Fine-tuned	bryn	fl	ane	l	check	shirt	dark	purple	shirts	shirt	in	checked	fl	anne	l
	with	a	collar	and	chest	pocket	.	regular	fit	.					

Tote bag

	fix	it	to	te	bag	bag	black	accessories	soft	hand	bag	in	imitation	
Frozen	leather	with	two	handles	,	a	zip	and	a	det	ach	able	shoulder	strap
	at	the	top	,	and	three	inner	compartments	,	one	with	a	zip	.
	size	27	x	31	cm	.								
	fix	it	to	te	bag	bag	black	accessories	soft	hand	bag	in	imitation	
Fine-tuned	leather	with	two	handles	,	a	zip	and	a	det	ach	able	shoulder	strap
	at	the	top	,	and	three	inner	compartments	,	one	with	a	zip	.
	size	27	x	31	cm	.								

Trousers

	jerry	jo	gger	bottoms	trousers	dark	grey	jersey	basic	trousers	in				
Frozen	sweatshirt	fabric	with	an	elastic	ated	draws	tri	ng	waist	,	side			
	pockets	,	a	back	pocket	and	rib	bed	hem	s	.	soft	brushed	inside	.
	jerry	jo	gger	bottoms	trousers	dark	grey	jersey	basic	trousers	in				
Fine-tuned	sweatshirt	fabric	with	an	elastic	ated	draws	tri	ng	waist	,	side			
	pockets	,	a	back	pocket	and	rib	bed	hem	s	.	soft	brushed	inside	.

(a) Text-tower attention on H&M item metadata.

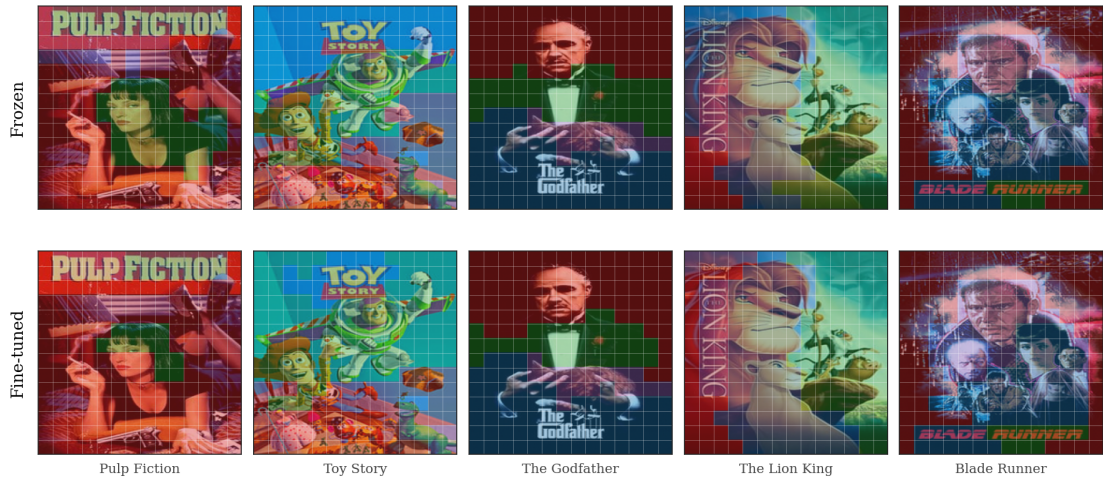


(b) Vision-tower attention on H&M item images.

Figure 6.5 Attention patterns on representative H&M items under the frozen (top) and fine-tuned (bottom) BGE-M3 + SigLIP2 encoders. The vision-tower panels show clustered patch attention from the final transformer layer; the text-tower panels show attention over tokens.

Pulp Fiction (1994)															
Frozen	pulp	fiction	(	1994	)	comedy		crime		drama		thriller			
Fine-tuned	pulp	fiction	(	1994	)	comedy		crime		drama		thriller			
Toy Story (1995)															
Frozen	toy	story	(	1995	)	adventure		animation		children		comedy			
	fantasy														
Fine-tuned	toy	story	(	1995	)	adventure		animation		children		comedy			
	fantasy														
The Godfather (1972)															
Frozen	godfather	,	the	(	1972	)	crime		drama						
Fine-tuned	godfather	,	the	(	1972	)	crime		drama						
The Lion King (1994)															
Frozen	lion	king	,	the	(	1994	)	adventure		animation		children		drama	
	musical		im	ax											
Fine-tuned	lion	king	,	the	(	1994	)	adventure		animation		children		drama	
	musical		im	ax											
Blade Runner (1982)															
Frozen	blade	runner	(	1982	)	action		sci	-	fi		thriller			
Fine-tuned	blade	runner	(	1982	)	action		sci	-	fi		thriller			

(a) Text-tower attention on MovieLens item metadata.



(b) Vision-tower attention on MovieLens posters.

Figure 6.6 Attention patterns on representative MovieLens items under the frozen (top) and fine-tuned (bottom) BGE-M3 + SigLIP2 encoders. The vision-tower panels show clustered patch attention from the final transformer layer; the text-tower panels show attention over tokens.

Discussion

This chapter evaluates the results of Chapters 3–6. It first summarizes the main findings, then discusses the implications of REEF for evaluation methodology, reviews the zero-shot and fine-tuning results, and summarizes what the representation analysis revealed. The chapter concludes with limitations and their practical implications.

7.1 Summary of Main Findings

The thesis focused on how embedding foundation model backbones should be evaluated, adapted, and understood in recommender systems. Chapter 3 introduced REEF, a protocol-based framework designed to isolate the contribution of backbone representations. Chapter 4 used this framework to evaluate pretrained encoders in a zero-shot setting and showed that multimodal representations consistently perform well. Chapter 5 demonstrated that fine-tuning improves the embedding space for recommendation. Chapter 6 then examined how this improvement shows in the structure of the learned representations.

7.2 REEF and Evaluation Methodology

We showed that the backbone isolation problem has direct evaluation consequences. When a backbone is evaluated as part of a full recommendation pipeline, conclusions about its quality become entangled with the design of downstream components. Differences in the downstream model, candidate construction, optimization settings, metrics, or data splits can affect the final result and make it unclear whether the observed performance comes from the backbone representation itself or from the surrounding pipeline.

We therefore built REEF as a controlled evaluation framework for backbone comparison. REEF fixes the data loader, downstream model, candidate construction, optimization settings, metrics, and data splits, so that different backbones can be evaluated under the same conditions. This reduces sources of variance that are unrelated to the representation and makes differences between backbones more directly comparable. In this thesis, REEF provides the evaluation basis for both the zero-shot study and the later fine-tuning experiments.

7.3 Evaluation of Zero-Shot and Fine-Tuning Results

After establishing REEF, we used it to evaluate pretrained foundation-model backbones in a zero-shot setting and then to study whether multimodal backbones can be improved through

fine-tuning. This made it possible to compare both frozen representations and adapted representations under the same evaluation protocol.

The zero-shot results show that modality quality depends strongly on the dataset. Text encoders perform best on MovieLens 20M and remain competitive on Amazon Reviews 2023, where textual metadata captures much of the preference signal. On H&M Fashion, image encoders become competitive or superior, reflecting the importance of visual appearance. Multimodal representations are strong across most evaluated datasets and scenarios. The combination of BGE-M3 and SigLIP2 is the strongest configuration in standard recommendation on MovieLens and H&M and is competitive on Amazon standard recommendation, where CLIP leads. The zero-shot transfer showed that MovieLens domain transfer reaches the highest absolute level and supports clear encoder-level conclusions, while the H&M and Amazon domain-transfer columns sit at a much smaller scale yet show the same broad ordering between encoders. The geometric reason for this is examined in Chapter 6, where the UMAP projections show that the held-out target domains on H&M and Amazon are largely disjoint from the source domains in the embedding space, while on MovieLens they remain mixed. The dataset-transfer columns, by contrast, are tightly compressed across encoders and too noisy to interpret. This suggests that the chosen source–target pairs were too distant for the cross-catalog scenarios, while domain transfer within a single catalog still carries usable signal. Meaningful cross-catalog evaluation would likely require few-shot retraining of the downstream model on the target dataset.

The fine-tuning study then showed that these multimodal representations can be successfully adapted to recommendation data. We performed a series of experiments over fusion strategies, tuning methods, and training objectives. Across the in-domain tasks, the best-performing fine-tuned configuration consistently improved over the frozen multimodal baseline, with the largest relative gains on Amazon (around +27 % Recall@10 in standard, +39 % in cold-start, +41 % in sequential) and smaller but consistent gains on the already strong MovieLens. Domain transfer follows the same pattern as the in-domain scenarios: MovieLens reaches the highest absolute level, where LoRA and adapters nudge above the frozen baseline while full fine-tuning stays flat, and the same per-method, per-fusion, and per-objective orderings reappear on H&M and Amazon at a much smaller scale. The effect sizes on those two datasets are small in absolute terms, but they are consistent enough across datasets and backbones to read as a trend. Dataset transfer is once more the one place where the experiments do not produce a usable signal.

The main fine-tuning findings are that while fixed modality combinations are useful, trainable fusion proved stronger results as it allows the model to learn how to align the text and images. The strongest results came from an interaction-aware adaptation: conditioning the fusion module on co-interaction context improved performance further, showing that user behavior provides useful information about how modalities should be weighted. The training objective also mattered: supervision based on item–item interaction structure was more effective than sparse pairwise ranking alone. Among the item–item objectives, ELSA was the most reliable, taking the per-column maxima on standard, cold-start, and sequential for BGE-M3 + SigLIP2 across all three datasets, and the same ordering reappeared on the domain-transfer column. The naive joint ELSA + InfoNCE loss, on the other hand, destabilized the optimization and performed below the zero-shot baseline.

Taken together, the dataset transfer results suggest that the cross-catalog scenarios were defined too aggressively. Domain transfer within a single catalog is interpretable on all three datasets and shows the same method, fusion, and objective orderings as the in-domain scenarios, just at smaller absolute scales.

7.4 Representation Analysis

The representation analysis explains why fine-tuning mostly improves recommendation performance. The main finding is that adaptation changes the embedding space mostly locally rather

than globally. The broad structure of the representation space remains recognizable after fine-tuning, but nearest-neighbor relationships become more aligned with user interactions. This means that the model does not simply overwrite the pretrained representation, instead, it preserves the semantic structure of the pretrained encoders while adjusting the parts of the space that matter most for recommendation. The attention analysis supports this interpretation. Visual and textual attention patterns remain largely stable after fine-tuning, showing that the internal behavior of the encoders has not fundamentally changed.

7.5 Limitations and Practical Implications

Several limitations of the thesis must be stated. The fine-tuning design space was explored through a staged search rather than a full combinatorial search, so interactions between fusion strategy, tuning method, training objective, target-similarity construction, and fusion conditioning may not have been fully captured. The final configuration therefore combines factors that were not jointly varied in any single experiment. The evaluation was also limited to five recommendation scenarios, other downstream tasks, such as search, were not evaluated, although they could be added to REEF given its modular design. The experiments use only three English-centric datasets, which may limit the generality of the findings for other languages or cultural contexts. The fine-tuning study considers only two multimodal configurations due to computational constraints. Finally, all current evaluations were performed in an offline setting, online evaluation is needed to verify whether the offline gains observed here translate into measurable improvements in user satisfaction and engagement.

A further limitation specific to the transfer evaluation is that the chosen cross-catalog scenarios were too aggressively defined. Domain transfer within a single catalog produces interpretable results on all three datasets, but the dataset-transfer column is too noisy to read across backbones, methods, fusion strategies, and objectives. Less distant source–target pairs would likely have yielded clearer cross-catalog evidence.

Despite these limitations, several practical conclusions can be drawn. The choice of modality should be guided by the characteristics of the dataset, although the experiments show that adding multimodal signals tends to improve in-domain performance in most cases. When fine-tuning is not feasible, strong text encoders such as BGE-M3 provide a reliable baseline. When adaptation is possible, the most effective strategies in our setup proved to be LoRA-style tuning, trainable interaction-conditioned fusion, and item–item interaction-derived supervision, with ELSA as the most reliable single objective. Domain transfer within a single catalog can be expected to follow the same orderings as in-domain ranking, just at a smaller absolute scale, cross-catalog transfer is harder to predict from offline experiments alone and should be repeated in the target catalog whenever possible. Finally, because fine-tuning preserves the internal attention patterns of the encoders, content-based interpretability remains meaningful after adaptation.

Conclusion

This thesis studied how foundation-model backbones can be evaluated, adapted, and understood in recommender systems. The central motivation was the backbone isolation problem: when an encoder is evaluated inside a full recommendation pipeline, its contribution becomes difficult to separate from the effects of the downstream model, optimization procedure, data split, candidate construction, and evaluation setup. To address this problem, we introduced REEF, a controlled evaluation framework for comparing backbone representations under identical experimental conditions.

The zero-shot study showed that pretrained foundation-model embeddings already contain useful signal for recommendation. The quality of each modality depends on the dataset: text encoders performed best where textual metadata captures most of the preference signal, image encoders became more competitive where visual appearance is more important, and multimodal representations were strong across most evaluated datasets and scenarios. This shows that modality choice matters, and that multimodal representations are most useful when text and image provide complementary information.

The fine-tuning study showed that multimodal foundation-model representations can be successfully adapted to recommendation data. The main conclusion is that fixed multimodal combinations are useful, but trainable fusion was stronger, since it allows the model to learn how to combine text and image information. Conditioning the fusion module on co-interaction context improved the results further, showing that user behavior provides useful signal for deciding how modalities should be weighted. Objectives based on item–item interaction structure were more effective than sparse pairwise ranking, and ELSA was the most reliable single objective. Domain transfer followed the same patterns as the in-domain scenarios at smaller absolute scales, subsequent representation analysis confirmed this geometrically, showing that the H&M and Amazon target domains are disjoint from their source domains in the embedding space, while MovieLens domains remain intermixed. The dataset-transfer scenarios were too aggressively defined and did not produce usable signals.

The representation analysis explained why fine-tuning improves recommendation performance. Adaptation changed the embedding space locally rather than globally as the broad semantic structure of the pretrained representation remained recognizable, while nearest-neighbor relationships became more aligned with user interactions. Attention patterns also remained largely stable after fine-tuning.

Overall, the thesis shows that foundation-model backbones are useful for recommender systems, but their value depends on how they are evaluated, which modalities are used, and how they are adapted to interaction data. REEF makes backbone comparison more controlled, the zero-shot experiments show that pretrained representations already provide useful recommendation signal, and the fine-tuning study shows that multimodal representations can be improved

through trainable, interaction-aware adaptation. Future work should extend this evaluation to additional downstream tasks, larger and more diverse backbone families, online evaluation, more multilingual and culturally diverse catalogs, as well as less aggressively defined cross-catalog transfer scenarios.

Bibliography

1. ZHANG, Aston; LIPTON, Zachary C.; LI, Mu; SMOLA, Alexander J. *Overview of Recommender Systems* [Chapter in *Dive into Deep Learning*]. 2023. [visited on 2026-04-25]. Available from: https://d2l.ai/chapter_recommender-systems/recsys-intro.html. Fig. 21.1.1, Illustration of the recommendation process.
2. RENDLE, Steffen; FREUDENTHALER, Christoph; GANTNER, Zeno; SCHMIDT-THIEME, Lars. BPR: Bayesian Personalized Ranking from Implicit Feedback. In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 2009, pp. 452–461. Available from DOI: 10.48550/arXiv.1205.2618.
3. HE, Xiangnan; LIAO, Lizi; ZHANG, Hanwang; NIE, Liqiang; HU, Xia; CHUA, Tat-Seng. Neural Collaborative Filtering. In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 173–182. Available from DOI: 10.1145/3038912.3052569.
4. ZHANG, Shuai; YAO, Lina; SUN, Aixin; TAY, Yi. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Computing Surveys*. 2019, vol. 52, no. 1, pp. 1–38. Available from DOI: 10.1145/3285029.
5. DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186. Available from DOI: 10.18653/v1/N19-1423.
6. XIAO, Shitao; LIU, Zheng; ZHANG, Peitian; MUENNIGHOFF, Niklas; LIAN, Defu; NIE, Jian-Yun. C-Pack: Packed Resources For General Chinese Embeddings. In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 641–649. Available from DOI: 10.1145/3626772.3657878.
7. OQUAB, Maxime; DAR CET, Timothée; MOUTAKANNI, Théo; VO, Huy V.; SZA FRANIEC, Marc; KHALIDOV, Vasil; FERNANDEZ, Pierre; HAZIZA, Daniel; MASSA, Francisco; EL-NOUBY, Alaaeldin; ASSRAN, Mahmoud; BALLAS, Nicolas; GALUBA, Wojciech; HOWES, Russell; HUANG, Po-Yao; LI, Shang-Wen; MISRA, Ishan; RAB BAT, Michael; SHARMA, Vasu; SYNNAEVE, Gabriel; XU, Hu; JEGOU, Hervé; MAIRAL, Julien; LABATUT, Patrick; JOULIN, Armand; BOJANOWSKI, Piotr. DINOv2: Learning Robust Visual Features without Supervision. *Transactions on Machine Learning Research*. 2024. Available from DOI: 10.48550/arXiv.2304.07193.
8. RADFORD, Alec; KIM, Jong Wook; HALLACY, Chris; RAMESH, Aditya; GOH, Gabriel; AGARWAL, Sandhini; SASTRY, Girish; ASKELL, Amanda; MISHKIN, Pamela; CLARK, Jack; KRUEGER, Gretchen; SUTSKEVER, Ilya. Learning Transferable Visual Models From Natural Language Supervision. In: *Proceedings of the 38th International Conference*

- on *Machine Learning*. 2021, pp. 8748–8763. Available from DOI: 10.48550/arXiv.2103.00020.
9. ZHAI, Xiaohua; MUSTAFA, Basil; KOLESNIKOV, Alexander; BEYER, Lucas. Sigmoid Loss for Language Image Pre-Training. In: *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023, pp. 11941–11952. Available from DOI: 10.1109/ICCV51070.2023.01100.
 10. RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha; KANTOR, Paul B. (eds.). *Recommender Systems Handbook*. Springer, 2011. Available from DOI: 10.1007/978-0-387-85820-3.
 11. AGGARWAL, Charu C. *Recommender Systems: The Textbook*. Springer, 2016. Available from DOI: 10.1007/978-3-319-29659-3.
 12. HU, Yifan; KOREN, Yehuda; VOLINSKY, Chris. Collaborative Filtering for Implicit Feedback Datasets. In: *Proceedings of the 2008 IEEE International Conference on Data Mining*. Pisa, Italy, 2008, pp. 263–272. Available from DOI: 10.1109/ICDM.2008.22.
 13. SAITO, Yuta; YAGINUMA, Suguru; NISHINO, Yuta; SAKATA, Hayato; NAKATA, Kazuhide. Unbiased Recommender Learning from Missing-Not-At-Random Implicit Feedback. In: *Proceedings of the Thirteenth ACM International Conference on Web Search and Data Mining*. 2020, pp. 501–509. Available from DOI: 10.1145/3336191.3371783.
 14. KOREN, Yehuda; BELL, Robert; VOLINSKY, Chris. Matrix Factorization Techniques for Recommender Systems. *Computer*. 2009, vol. 42, no. 8, pp. 30–37. Available from DOI: 10.1109/MC.2009.263.
 15. SCHEIN, Andrew I.; POPESCU, Alexandrin; UNGAR, Lyle H.; PENNOCK, David M. Methods and metrics for cold-start recommendations. In: *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2002, pp. 253–260. Available from DOI: 10.1145/564376.564421.
 16. SHANI, Guy; GUNAWARDANA, Asela. Evaluating Recommendation Systems. In: *Recommender Systems Handbook*. Springer, 2011, pp. 257–297. Available from DOI: 10.1007/978-0-387-85820-3_8.
 17. JÄRVELIN, Kalervo; KEKÄLÄINEN, Jaana. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Transactions on Information Systems*. 2002, vol. 20, no. 4, pp. 422–446. Available from DOI: 10.1145/582415.582418.
 18. MOGHADDAM, Farshad Bakhshandegan; ELAHI, Mehdi. Cold-start solutions for recommendation systems. In: *Big Data Recommender Systems - Volume 2: Application Paradigms*. Institution of Engineering and Technology, 2019, pp. 35–56. Available from DOI: 10.1049/PBPC035G_ch3.
 19. ABDOLLAHPOURI, Himan; BURKE, Robin; MOBASHER, Bamshad. Managing Popularity Bias in Recommender Systems with Personalized Re-Ranking. *arXiv preprint arXiv:1901.07555*. 2019. Available from DOI: 10.48550/arXiv.1901.07555.
 20. PAN, Sinno Jialin; YANG, Qiang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*. 2010, vol. 22, no. 10, pp. 1345–1359. Available from DOI: 10.1109/TKDE.2009.191.
 21. BOMMASANI, Rishi; HUDSON, Drew A.; ADELI, Ehsan; ALTMAN, Russ; ARORA, Simran; ARX, Sydney von; BERNSTEIN, Michael S.; BOHG, Jeannette; BOSSELUT, Antoine; BRUNSKILL, Emma, et al. On the Opportunities and Risks of Foundation Models. *arXiv preprint arXiv:2108.07258*. 2021. Available from DOI: 10.48550/arXiv.2108.07258.
 22. YOSINSKI, Jason; CLUNE, Jeff; BENGIO, Yoshua; LIPSON, Hod. How transferable are features in deep neural networks? In: *Advances in Neural Information Processing Systems*. 2014, vol. 27, pp. 3320–3328. Available from DOI: 10.48550/arXiv.1411.1792.

23. HU, Edward J.; SHEN, Yelong; WALLIS, Phillip; ALLEN-ZHU, Zeyuan; LI, Yuanzhi; WANG, Shean; WANG, Lu; CHEN, Weizhu. LoRA: Low-Rank Adaptation of Large Language Models. In: *International Conference on Learning Representations*. 2022. Available from DOI: 10.48550/arXiv.2106.09685.
24. KIRKPATRICK, James; PASCANU, Razvan; RABINOWITZ, Neil; VENESS, Joel; DESJARDINS, Guillaume; RUSU, Andrei A.; MILAN, Kieran; QUAN, John; RAMALHO, Tiago; GRABSKA-BARWINSKA, Agnieszka, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*. 2017, vol. 114, no. 13, pp. 3521–3526. Available from DOI: 10.1073/pnas.1611835114.
25. HOULSBY, Neil; GIURGIU, Andrei; JASTRZEBSKI, Stanislaw; MORRONE, Bruna; LARO USSILHE, Quentin de; GESMUNDO, Andrea; ATTARIYAN, Mona; GELLY, Sylvain. Parameter-Efficient Transfer Learning for NLP. In: *Proceedings of the 36th International Conference on Machine Learning*. 2019, pp. 2790–2799. Available from DOI: 10.48550/arXiv.1902.00751.
26. FINN, Chelsea; ABBEEL, Pieter; LEVINE, Sergey. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In: *Proceedings of the 34th International Conference on Machine Learning*. PMLR, 2017, vol. 70, pp. 1126–1135. Proceedings of Machine Learning Research. Available from DOI: 10.48550/arXiv.1703.03400.
27. SNELL, Jake; SWERSKY, Kevin; ZEMEL, Richard S. Prototypical Networks for Few-shot Learning. In: *Advances in Neural Information Processing Systems*. 2017, vol. 30. Available from DOI: 10.48550/arXiv.1703.05175.
28. BALTRUŠAITIS, Tadas; AHUJA, Chaitanya; MORENCY, Louis-Philippe. Multimodal Machine Learning: A Survey and Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2019, vol. 41, no. 2, pp. 423–443. Available from DOI: 10.1109/TPAMI.2018.2798607.
29. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. Attention Is All You Need. In: *Advances in Neural Information Processing Systems*. 2017. Available from DOI: 10.48550/arXiv.1706.03762.
30. AREVALO, John; SOLORIO, Thamar; MONTES-Y-GÓMEZ, Manuel; GONZÁLEZ, Fabio A. Gated Multimodal Units for Information Fusion. In: *ICLR Workshop*. 2017. Available from DOI: 10.48550/arXiv.1702.01992.
31. LIU, Tie-Yan. *Learning to Rank for Information Retrieval*. Springer, 2011. Available from DOI: 10.1007/978-3-642-14267-3.
32. OORD, Aaron van den; LI, Yazhe; VINYALS, Oriol. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*. 2018. Available from DOI: 10.48550/arXiv.1807.03748.
33. VANČURA, Vojtěch; ALVES, Rodrigo; KASALICKÝ, Petr; KORDÍK, Pavel. Scalable Linear Shallow Autoencoder for Collaborative Filtering. In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 604–609. Available from DOI: 10.1145/3523227.3551482.
34. VANČURA, Vojtěch; KORDÍK, Pavel; STRAKA, Milan. beeFormer: Bridging the Gap Between Semantic and Interaction Similarity in Recommender Systems. In: *Proceedings of the 18th ACM Conference on Recommender Systems*. 2024, pp. 1102–1107. Available from DOI: 10.1145/3640457.3691707.

35. DOSOVITSKIY, Alexey; BEYER, Lucas; KOLESNIKOV, Alexander; WEISSENBORN, Dirk; ZHAI, Xiaohua; UNTERTHINER, Thomas; DEHGHANI, Mostafa; MINDERER, Matthias; HEIGOLD, Georg; GELLY, Sylvain; USZKOREIT, Jakob; HOULSBY, Neil. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: *International Conference on Learning Representations*. 2021. Available from DOI: 10.48550/arXiv.2010.11929.
36. MCINNES, Leland; HEALY, John; SAUL, Nathaniel; GROSSBERGER, Lukas. UMAP: Uniform Manifold Approximation and Projection. *Journal of Open Source Software*. 2018, vol. 3, no. 29, p. 861. Available from DOI: 10.21105/joss.00861.
37. LLOYD, Stuart P. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*. 1982, vol. 28, no. 2, pp. 129–137. Available from DOI: 10.1109/TIT.1982.1056489.
38. STECK, Harald; BALTRUNAS, Linas; ELAHI, Ehtsham; LIANG, Dawen; RAIMOND, Yves; BASILICO, Justin. Deep Learning for Recommender Systems: A Netflix Case Study. *AI Magazine*. 2021, vol. 42, no. 3, pp. 7–18. Available from DOI: 10.1609/aimag.v42i3.18140.
39. HSIAO, Ko-Jen; FENG, Yesu; LAMKHEDE, Sudarshan. *Foundation Model for Personalized Recommendation* [online]. 2025. [visited on 2026-04-25]. Available from: <https://netflixtechblog.com/foundation-model-for-personalized-recommendation-1a0bd8e02d39>.
40. HANSEN, Casper; HANSEN, Christian; MAYSTRE, Lucas; MEHROTRA, Rishabh; BROST, Brian; TOMASI, Federico; LALMAS, Mounia. Contextual and Sequential User Embeddings for Large-Scale Music Recommendation. In: *Proceedings of the 14th ACM Conference on Recommender Systems*. 2020, pp. 53–62. Available from DOI: 10.1145/3383313.3412248.
41. REIMERS, Nils; GUREVYCH, Iryna. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. 2019, pp. 3982–3992. Available from DOI: 10.18653/v1/D19-1410.
42. KARPUKHIN, Vladimir; OĞUZ, Barlas; MIN, Sewon; LEWIS, Patrick; WU, Ledell; EDUNOV, Sergey; CHEN, Danqi; YIH, Wen-tau. Dense Passage Retrieval for Open-Domain Question Answering. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 2020, pp. 6769–6781. Available from DOI: 10.18653/v1/2020.emnlp-main.550.
43. IZACARD, Gautier; CARON, Mathilde; HOSSEINI, Lucas; RIEDEL, Sebastian; BOJANOWSKI, Piotr; JOULIN, Armand; GRAVE, Edouard. Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research*. 2022. Available from DOI: 10.48550/arXiv.2112.09118.
44. WANG, Liang; YANG, Nan; HUANG, Xiaolong; JIAO, Binxing; YANG, Linjun; JIANG, Daxin; MAJUMDER, Rangan; WEI, Furu. Text Embeddings by Weakly-Supervised Contrastive Pre-training. *arXiv preprint arXiv:2212.03533*. 2022. Available from DOI: 10.48550/arXiv.2212.03533.
45. CHEN, Jianlv; XIAO, Shitao; ZHANG, Peitian; LUO, Kun; LIAN, Defu; LIU, Zheng. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. *arXiv preprint arXiv:2402.03216*. 2024. Available from DOI: 10.48550/arXiv.2402.03216.
46. LI, Zehan; ZHANG, Xin; ZHANG, Yanzhao; LONG, Dingkun; XIE, Pengjun; ZHANG, Meishan. Towards General Text Embeddings with Multi-stage Contrastive Learning. *arXiv preprint arXiv:2308.03281*. 2023. Available from DOI: 10.48550/arXiv.2308.03281.

47. WU, Chuhan; WU, Fangzhao; HUANG, Yongfeng; XIE, Xing. Personalized News Recommendation: Methods and Challenges. *ACM Transactions on Information Systems*. 2023, vol. 41, no. 1, pp. 1–50. Available from DOI: 10.1145/3530257.
48. WANG, Hongwei; ZHANG, Fuzheng; XIE, Xing; GUO, Minyi. DKN: Deep Knowledge-Aware Network for News Recommendation. In: *Proceedings of the 2018 World Wide Web Conference*. 2018, pp. 1835–1844. Available from DOI: 10.1145/3178876.3186175.
49. WU, Chuhan; WU, Fangzhao; AN, Mingxiao; HUANG, Jianqiang; HUANG, Yongfeng; XIE, Xing. Neural News Recommendation with Attentive Multi-View Learning. In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. 2019, pp. 3863–3869. Available from DOI: 10.24963/ijcai.2019/536.
50. WU, Chuhan; WU, Fangzhao; GE, Suyu; QI, Tao; HUANG, Yongfeng; XIE, Xing. Neural News Recommendation with Multi-Head Self-Attention. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 2019, pp. 6389–6394. Available from DOI: 10.18653/v1/D19-1671.
51. WU, Chuhan; WU, Fangzhao; QI, Tao; HUANG, Yongfeng. Empowering News Recommendation with Pre-trained Language Models. *arXiv preprint arXiv:2104.07413*. 2021. Available from DOI: 10.48550/arXiv.2104.07413.
52. DING, Hao; MA, Yifei; DEORAS, Anoop; WANG, Yuyang; WANG, Hao. Zero-Shot Recommender Systems. *arXiv preprint arXiv:2105.08318*. 2021. Available from DOI: 10.48550/arXiv.2105.08318.
53. HOU, Yupeng; MU, Shanlei; ZHAO, Wayne Xin; LI, Yaliang; DING, Bolin; WEN, Ji-Rong. Towards Universal Sequence Representation Learning for Recommender Systems. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 585–593. Available from DOI: 10.1145/3534678.3539381.
54. HOU, Yupeng; HE, Zhankui; MCAULEY, Julian; ZHAO, Wayne Xin. Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders. In: *Proceedings of the ACM Web Conference 2023*. 2023, pp. 1162–1171. Available from DOI: 10.1145/3543507.3583434.
55. LI, Jiacheng; WANG, Ming; LI, Jin; FU, Jinmiao; SHEN, Xin; SHANG, Jingbo; MCAULEY, Julian. Text Is All You Need: Learning Language Representations for Sequential Recommendation. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 1258–1267. Available from DOI: 10.1145/3580305.3599519.
56. HE, Ruining; MCAULEY, Julian. VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback. In: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. 2016, pp. 144–150. Available from DOI: 10.1609/aaai.v30i1.9973.
57. KANG, Wang-Cheng; FANG, Chen; WANG, Zhaowen; MCAULEY, Julian. Visually-Aware Fashion Recommendation and Design with Generative Image Models. In: *Proceedings of the 2017 IEEE International Conference on Data Mining*. 2017, pp. 207–216. Available from DOI: 10.1109/ICDM.2017.30.
58. YI, Zixuan; LONG, Zijun; OUNIS, Iadh; MACDONALD, Craig; MCCREADIE, Richard. Large Multi-modal Encoders for Recommendation. *arXiv preprint arXiv:2310.20343*. 2023. Available from DOI: 10.48550/arXiv.2310.20343.
59. ZHOU, Xin; ZHANG, Xiaoxiong; NIYATO, Dusit; SHEN, Zhiqi. Learning Item Representations Directly from Multimodal Features for Effective Recommendation. *arXiv preprint arXiv:2505.04960*. 2025. Available from DOI: 10.48550/arXiv.2505.04960.

60. VOLKOV, Maksims; YU, Guangwei; POUTANEN, Tomi. DropoutNet: Addressing Cold Start in Recommender Systems. In: *Advances in Neural Information Processing Systems*. 2017. Available also from: <https://papers.nips.cc/paper/7081-dropoutnet-addressing-cold-start-in-recommender-systems>.
61. ZHU, Ziwei; SEFATI, Shahin; SAADATPANAH, Parsa; CAVERLEE, James. Recommendation for New Users and New Items via Randomized Training and Mixture-of-Experts Transformation. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 1121–1130. Available from DOI: 10.1145/3397271.3401178.
62. WEI, Yinwei; WANG, Xiang; LI, Qi; NIE, Liqiang; LI, Yan; LI, Xuanping; CHUA, Tat-Seng. Contrastive Learning for Cold-Start Recommendation. In: *Proceedings of the 29th ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 5382–5390. MM '21. ISBN 9781450386517. Available from DOI: 10.1145/3474085.3475665.
63. WANG, Jie; YUAN, Fajie; CHENG, Mingyue; JOSE, Joemon M.; YU, Chenyun; KONG, Beibei; HE, Xiangnan; WANG, Zhijin; HU, Bo; LI, Zang. TransRec: Learning Transferable Recommendation from Mixture-of-Modality Feedback. *arXiv preprint arXiv:2206.06190*. 2022. Available from DOI: 10.48550/arXiv.2206.06190.
64. LI, Youhua; DU, Hanwen; NI, Yongxin; ZHAO, Pengpeng; GUO, Qi; YUAN, Fajie; ZHOU, Xiaofang. Multi-Modality is All You Need for Transferable Recommender Systems. In: *Proceedings - 2024 IEEE 40th International Conference on Data Engineering, ICDE 2024*. IEEE Computer Society, 2024, pp. 5008–5021. Available from DOI: 10.1109/ICDE60146.2024.00380.
65. LEE, Hoyeop; IM, Jinbae; JANG, Seongwon; CHO, Hyunsouk; CHUNG, Sehee. MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1073–1082. Available from DOI: 10.1145/3292500.3330859.
66. DONG, Manqing; YUAN, Feng; YAO, Lina; XU, Xiwei; ZHU, Liming. MAMO: Memory-Augmented Meta-Optimization for Cold-Start Recommendation. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2020, pp. 688–697. Available from DOI: 10.1145/3394486.3403113.
67. ZHU, Yongchun; TANG, Zhenwei; LIU, Yudan; ZHUANG, Fuzhen; XIE, Ruobing; ZHANG, Xu; LIN, Leyu; HE, Qing. Personalized Transfer of User Preferences for Cross-Domain Recommendation. In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. ACM, 2022, pp. 1507–1515. Available from DOI: 10.1145/3488560.3498392.
68. GARNELO, Marta; ROSENBAUM, Dan; MADDISON, Christopher J.; RAMALHO, Tiago; SAXTON, David; SHANAHAN, Murray; TEH, Yee Whye; REZENDE, Danilo J.; ESLAMI, S. M. Ali. Conditional Neural Processes. In: *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 2018, vol. 80, pp. 1704–1713. Proceedings of Machine Learning Research. Available from DOI: 10.48550/arXiv.1807.01613.
69. YING, Rex; HE, Ruining; CHEN, Kaifeng; EKSOMBATCHAI, Pong; HAMILTON, William L.; LESKOVEC, Jure. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2018, pp. 974–983. Available from DOI: 10.1145/3219819.3219890.
70. WANG, Xiang; HE, Xiangnan; WANG, Meng; FENG, Fuli; CHUA, Tat-Seng. Neural Graph Collaborative Filtering. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2019, pp. 165–174. Available from DOI: 10.1145/3331184.3331267.

71. HE, Xiangnan; DENG, Kuan; WANG, Xiang; LI, Yan; ZHANG, Yongdong; WANG, Meng. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 639–648. Available from DOI: 10.1145/3397271.3401063.
72. WU, Jiancan; WANG, Xiang; FENG, Fuli; HE, Xiangnan; CHEN, Liang; LIAN, Jianxun; XIE, Xing. Self-supervised Graph Learning for Recommendation. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 726–735. Available from DOI: 10.1145/3404835.3462862.
73. YU, Junliang; YIN, Hongzhi; XIA, Xin; CHEN, Tong; CUI, Lizhen; NGUYEN, Quoc Viet Hung. Are Graph Augmentations Necessary? Simple Graph Contrastive Learning for Recommendation. In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pp. 1294–1303. Available from DOI: 10.1145/3477495.3531937.
74. WEI, Yinwei; WANG, Xiang; NIE, Liqiang; HE, Xiangnan; HONG, Richang; CHUA, Tat-Seng. MMGCN: Multi-modal Graph Convolution Network for Personalized Recommendation of Micro-video. In: *Proceedings of the 27th ACM International Conference on Multimedia*. 2019, pp. 1437–1445. Available from DOI: 10.1145/3343031.3351034.
75. ZHOU, Xin; SHEN, Zhiqi. A Tale of Two Graphs: Freezing and Denoising Graph Structures for Multimodal Recommendation. In: *Proceedings of the 31st ACM International Conference on Multimedia*. 2023, pp. 935–943. Available from DOI: 10.1145/3581783.3611943.
76. XU, Jinfeng; CHEN, Zheyu; YANG, Shuo; LI, Jinze; WANG, Hewei; NGAI, Edith C. H. MENTOR: Multi-level Self-supervised Learning for Multimodal Recommendation. *arXiv preprint arXiv:2402.19407*. 2024. Available from DOI: 10.48550/arXiv.2402.19407.
77. GENG, Shijie; LIU, Shuchang; FU, Zuohui; GE, Yingqiang; ZHANG, Yongfeng. Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 299–315. Available from DOI: 10.1145/3523227.3546767.
78. BAO, Keqin; ZHANG, Jizhi; ZHANG, Yang; WANG, Wenjie; FENG, Fuli; HE, Xiangnan. TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation. *arXiv preprint arXiv:2305.00447*. 2023. Available from DOI: 10.48550/arXiv.2305.00447.
79. ZHANG, Junjie; XIE, Ruobing; HOU, Yupeng; ZHAO, Wayne Xin; LIN, Leyu; WEN, Ji-Rong. Recommendation as Instruction Following: A Large Language Model Empowered Recommendation Approach. *arXiv preprint arXiv:2305.07001*. 2023. Available from DOI: 10.48550/arXiv.2305.07001.
80. HOU, Yupeng; LI, Jiacheng; FU, Xiangjun; HE, Zhankui; YAN, An; CHEN, Xiusi; MCAULEY, Julian. *Bridging Language and Items for Retrieval and Recommendation: Benchmarking LLMs as Semantic Encoders*. 2024. Available from arXiv: 2403.03952 [cs.IR].
81. ZHANG, Jiaqi; CHENG, Yu; NI, Yongxin; PAN, Yunzhu; YUAN, Zheng; FU, Junchen; LI, Youhua; WANG, Jie; YUAN, Fajie. NineRec: A Benchmark Dataset Suite for Evaluating Transferable Recommendation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2024. Available from DOI: 10.1109/TPAMI.2024.3373868.
82. H&M GROUP. *H&M Personalized Fashion Recommendations* [online]. 2022. [visited on 2026-04-25]. Available from: <https://www.kaggle.com/competitions/h-and-m-personalized-fashion-recommendations>.
83. GROUPLENS RESEARCH. *MovieLens 20M Dataset* [<https://grouplens.org/datasets/movielens/20m/>]. 2016. Released April 2015; updated October 2016.

84. GHRZAREA. *MovieLens 20M Posters for Machine Learning* [<https://www.kaggle.com/datasets/ghrzarea/movielens-20m-posters-for-machine-learning>]. 2018. Kaggle dataset containing movie poster images for MovieLens 20M.
85. WU, Fangzhao; QIAO, Ying; CHEN, Jiun-Hung; WU, Chuhan; QI, Tao; LIAN, Jianxun; LIU, Danyang; XIE, Xing; GAO, Jianfeng; WU, Winnie; ZHOU, Ming. MIND: A Large-scale Dataset for News Recommendation. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 3597–3606. Available from DOI: 10.18653/v1/2020.acl-main.331.
86. GAO, Chongming; LI, Shijun; LEI, Wenqiang; CHEN, Jiawei; LI, Biao; JIANG, Peng; HE, Xiangnan; MAO, Jiabin; CHUA, Tat-Seng. KuaiRec: A Fully-observed Dataset and Insights for Evaluating Recommender Systems. In: *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*. 2022, pp. 540–550. Available from DOI: 10.1145/3511808.3557220.
87. SUN, Zhongxiang; SI, Zihua; ZANG, Xiaoxue; LENG, Dewei; NIU, Yanan; SONG, Yang; ZHANG, Xiao; XU, Jun. KuaiSAR: A Unified Search And Recommendation Dataset. In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2023, pp. 5407–5411. Available from DOI: 10.1145/3583780.3615123.
88. YELP. *Yelp Open Dataset* [online]. [visited on 2026-04-25]. Available from: <https://www.yelp.com/dataset>.
89. RETAILROCKET. *RetailRocket Recommender System Dataset* [online]. 2017. [visited on 2026-04-25]. Available from: <https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset>.
90. ZIEGLER, Cai-Nicolas; MCNEE, Sean M.; KONSTAN, Joseph A.; LAUSEN, Georg. Improving Recommendation Lists Through Topic Diversification. In: *Proceedings of the 14th International Conference on World Wide Web*. 2005, pp. 22–32. Available from DOI: 10.1145/1060745.1060754.
91. ZAJAC, Zygmunt. *Goodbooks-10k: A New Dataset for Book Recommendations* [online]. 2017. [visited on 2026-04-25]. Available from: <https://github.com/zygmuntz/goodbooks-10k>.
92. SCHEDL, Markus. The LFM-1b Dataset for Music Retrieval and Recommendation. In: *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. 2016, pp. 103–110. Available from DOI: 10.1145/2911996.2912004.
93. EKSTRAND, Michael D.; LUDWIG, Michael; KOLB, Jack; RIEDL, John T. LensKit: a modular recommender framework. In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. 2011, pp. 349–350. Available from DOI: 10.1145/2043932.2044001.
94. HUG, Nicolas. Surprise: A Python Library for Recommender Systems. *Journal of Open Source Software*. 2020, vol. 5, no. 52, p. 2174. Available from DOI: 10.21105/joss.02174.
95. ZHAO, Wayne Xin; MU, Shanlei; HOU, Yupeng; LIN, Zihan; CHEN, Yushuo; PAN, Xingyu; LI, Kaiyuan; LU, Yujie; WANG, Hui; TIAN, Changxin; MIN, Yingqian; FENG, Zhichao; FAN, Xinyan; CHEN, Xu; WANG, Pengfei; JI, Wendi; LI, Yaliang; WANG, Xiaoling; WEN, Ji-Rong. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. In: *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 2021, pp. 4653–4664. Available from DOI: 10.1145/3459637.3482016.

96. ZHAO, Wayne Xin; HOU, Yupeng; PAN, Xingyu; YANG, Chen; ZHANG, Zeyu; LIN, Zihan; ZHANG, Jingsen; BIAN, Shuqing; TANG, Jiakai; SUN, Wenqi; CHEN, Yushuo; XU, Lanling; ZHANG, Gaowei; TIAN, Zhen; TIAN, Changxin; MU, Shanlei; FAN, Xinyan; CHEN, Xu; WEN, Ji-Rong. RecBole 2.0: Towards a More Up-to-Date Recommendation Library. In: *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*. 2022, pp. 4722–4726. Available from DOI: 10.1145/3511808.3557680.
97. ANELLI, Vito Walter; BELLOGÍN, Alejandro; FERRARA, Antonio; MALITESTA, Daniele; MERRA, Felice Antonio; POMO, Claudio; DONINI, Francesco Maria; DI NOIA, Tommaso. Elliot: A Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation. In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 2405–2414. Available from DOI: 10.1145/3404835.3463245.
98. TENSORFLOW TEAM. *TensorFlow Recommenders* [online]. 2020. [visited on 2026-04-25]. Available from: <https://www.tensorflow.org/recommenders>.
99. IVCHENKO, Dmytro; VAN DER STAAY, Dennis; TAYLOR, Colin; LIU, Xing; FENG, Will; KINDI, Rahul; SUDARSHAN, Anirudh; SEFATI, Shahin. TorchRec: a PyTorch Domain Library for Recommendation Systems. In: *Proceedings of the 16th ACM Conference on Recommender Systems*. 2022, pp. 482–483. Available from DOI: 10.1145/3523227.3547387.
100. SALAH, Aghiles; TRUONG, Quoc-Tuan; LAUW, Hady W. Cornac: A Comparative Framework for Multimodal Recommender Systems. *Journal of Machine Learning Research*. 2020, vol. 21, no. 95, pp. 1–5. Available also from: <https://www.jmlr.org/papers/v21/19-805.html>.
101. ZHOU, Xin. MMRec: Simplifying Multimodal Recommendation. In: *Proceedings of the 5th ACM International Conference on Multimedia in Asia Workshops*. 2023, 6:1–6:2. Available from DOI: 10.1145/3611380.3628561.
102. XU, Shuyuan; HUA, Wenyue; ZHANG, Yongfeng. OpenP5: An Open-Source Platform for Developing, Training, and Evaluating LLM-based Recommender Systems. In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 386–394. Available from DOI: 10.1145/3626772.3657883.
103. JU, Clark Mingxuan; COLLINS, Liam; NEVES, Leonardo; KUMAR, Bhuvish; WANG, Louis Yufeng; ZHAO, Tong; SHAH, Neil. Generative Recommendation with Semantic IDs: A Practitioner’s Handbook. In: *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*. 2025, pp. 6420–6425. Available from DOI: 10.1145/3746252.3761612.
104. TSCHANNEN, Michael; GRITSENKO, Alexey; WANG, Xiao; NAEEM, Muhammad Ferjad; ALABDULMOHSIN, Ibrahim; PARTHASARATHY, Nikhil; EVANS, Talfan; BEYER, Lucas; XIA, Ye; MUSTAFA, Basil; HÉNAFF, Olivier; HARMSSEN, Jeremiah; STEINER, Andreas; ZHAI, Xiaohua. SigLIP 2: Multilingual Vision-Language Encoders with Improved Semantic Understanding, Localization, and Dense Features. *arXiv preprint arXiv:2502.14786*. 2025. Available from DOI: 10.48550/arXiv.2502.14786.
105. KANG, Wang-Cheng; MCAULEY, Julian. Self-Attentive Sequential Recommendation. In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 197–206. Available from DOI: 10.1109/ICDM.2018.00035.

Sample REEF Configuration

Listing A.1 shows an example REEF configuration file corresponding to the tuning-method comparison reported in Section 5.4.1. It loads the Amazon Reviews `Toys_and_Games` category and declares two backbone-trainer pairs that share the same multimodal BGE-M3 + SigLIP2 encoder but differ in adaptation: one is kept frozen, the other is fine-tuned with LoRA adapters under the ELSA objective with concatenation fusion. Both are evaluated on the standard ranking task.

■ **Code listing A.1** Sample REEF configuration with two backbone-trainer pairs (frozen vs. LoRA-adapted) on the Amazon `Toys_and_Games` standard ranking task.

```
seeds: [42]
bundle:
  loader: reef.loaders.amazon:AmazonReviewsLoader
  params:
    categories: [Toys_and_Games]
    min_interactions_per_user: 10
    min_interactions_per_item: 10
backbones:
  bge_siglip_frozen:
    model:
      target: reef.backbones.multimodal:ConcatBackbone
      params:
        text:
          name: BAAI/bge-m3
          dim: 1024
        vision:
          name: google/siglip2-base-patch16-224
          dim: 768
        fusion: concat
        projection_dim: 256
        modalities: [text, image]
    trainer:
      target: reef.trainers.frozen:FrozenTrainer
      params: {}
  bge_siglip_lora:
    model:
      target: reef.backbones.multimodal:ConcatBackbone
      params:
        text:
          name: BAAI/bge-m3
          dim: 1024
        vision:
          name: google/siglip2-base-patch16-224
          dim: 768
        fusion: concat
```

```

        projection_dim: 256
        modalities: [text, image]
    trainer:
        target: reef.trainers.elsa:ELSATrainer
        params:
            adapter: lora
            lora_rank: 16
            lora_targets: [q_proj, v_proj]
            learning_rate: 2.0e-4
            weight_decay: 1.0e-2
            batch_size: 2048
            max_epochs: 100
            warmup_ratio: 0.1
            lr_schedule: cosine
            early_stopping:
                monitor: val_ndcg_at_10
                patience: 5
            gradient_checkpointing: true
    tasks:
        standard_toys:
            task:
                target: reef.tasks.standard:StandardRankingTask
                params:
                    task_id: standard_toys
                    kind: ranking
                    dataset_id: Toys_and_Games
                    split: chronological
                    train_fraction: 0.8
                    val_fraction: 0.1
                    test_fraction: 0.1
                    num_negatives: 999
            downstream:
                target: reef.downstreams.ranker:AttentionPoolingRanker
                params:
                    kind: ranking
                    item_tower_hidden: 256
                    item_tower_out: 128
                    dropout: 0.1
                    learning_rate: 1.0e-3
                    batch_size: 1024
                    max_epochs: 100
                    early_stopping:
                        monitor: val_ndcg_at_10
                        patience: 3
            evaluator:
                target: reef.evaluators.ranking:RankingEvaluator
                params:
                    supported_kinds: [ranking]
                    metrics: [recall, ndcg]
                    k_values: [10]

```