WARSAW UNIVERSITY OF TECHNOLOGY

Ph.D. Thesis

Discipline of Science: Information and Communications Technology
Field of Science: Engineering and Technology

M.Sc. Eng.
Michał Daniluk

Multimodal recommendation system for e-commerce

Supervisor
D.Sc. PW Professor **Grzegorz Pastuszak**

WARSZAWA 2026

Abstract

Recommender systems in e-commerce have largely adopted sequential modeling techniques from natural language processing, treating user interaction histories as ordered token sequences. This paradigm rests on a flawed analogy: unlike words in a sentence, the order of purchases and clicks reflects page layout, promotional timing, and logistical convenience rather than meaningful preference dynamics. Point-based representations compound the problem by averaging item embeddings into a single vector, collapsing genuinely multimodal preferences into uninformative centroids. The result is a systematic loss of behavioral structure at the representational level, before any model architecture can compensate.

This thesis develops a distributional alternative grounded in density estimation over item embedding manifolds. The first contribution, the Efficient Manifold Density Estimator (EMDE), uses locality-sensitive hashing with density-dependent partitioning to represent behavioral histories as probability distributions, producing fixed-size sketches that preserve multimodal preference structure and integrate heterogeneous data modalities through concatenation. The second contribution, BaseModel, is a self-supervised foundation model that predicts future behavioral distributions from past ones. A single pretrained backbone supports recommendation, classification, and regression without task-specific labels, shifting the computational cost from per-task training to a one-time pretraining step.

To validate these contributions, research hypotheses were formulated and tested, and research objectives were defined and achieved across three complementary settings. On academic benchmarks, BaseModel achieves over 100% improvement against Meta's HSTU and state-of-the-art results on 10 of 12 RelBench tasks. In international competitions, the methods earned consistent top-three placements alongside teams from DeepMind, Baidu, and NVIDIA, including a Best Paper Award at the WSDM WebTour Challenge. In production, the system processes 42 billion events monthly across more than 200 organizations, delivering measurable gains in engagement and conversion. Taken together, the results demonstrate that representational choices dominate architectural ones: simple feed-forward networks operating on distributional sketches outperform complex sequential and graph-based architectures that discard the very structure they attempt to model.

Keywords: recommender systems, density estimation, multimodal representation, foundation models, locality-sensitive hashing, behavioral modeling

Streszczenie

Systemy rekomendacyjne w e-commerce w znacznej mierze przejęły techniki modelowania sekwencyjnego z przetwarzania języka naturalnego, traktując historie interakcji użytkowników jako uporządkowane ciągi tokenów. Analogia ta jest jednak błędna: w odróżnieniu od słów w zdaniu, kolejność zakupów i kliknięć wynika z układu strony, harmonogramu promocji czy względów logistycznych, a nie z rzeczywistej dynamiki preferencji. Reprezentacje punktowe pogłębiają ten problem: uśrednianie wektorów produktów do pojedynczego wektora sprawia, że wielomodalne preferencje zwijają się do centroidu pozbawionego użytecznej informacji. W efekcie struktura behawioralna jest systematycznie tracona już na poziomie reprezentacji, zanim jakakolwiek architektura modelu zdąży to skompensować.

Niniejsza rozprawa proponuje alternatywę dystrybucyjną opartą na estymacji gęstości na rozmaitościach reprezentacji wektorowych produktów. Pierwszym wkładem jest Efficient Manifold Density Estimator (EMDE), metoda wykorzystująca haszowanie wrażliwe na lokalność z podziałem zależnym od gęstości danych do reprezentowania historii behawioralnych jako rozkładów prawdopodobieństwa. Metoda ta generuje szkice o stałym rozmiarze, które zachowują wielomodalną strukturę preferencji i umożliwiają integrację heterogenicznych modalności danych poprzez prostą konkatenację. Drugim wkładem jest BaseModel, samonadzorowany model fundacyjny, który na podstawie przeszłych rozkładów behawioralnych przewiduje przyszłe. Pojedynczy wstępnie wytrenowany rdzeń obsługuje zadania rekomendacji, klasyfikacji i regresji bez etykiet specyficznych dla danego zadania, co przenosi koszt obliczeniowy z trenowania specyficznego dla każdego zadania na jednorazowy etap pretrenowania.

Aby zweryfikować te wkłady, sformułowano i przetestowano hipotezy badawcze, a cele badawcze zdefiniowano i zrealizowano w trzech uzupełniających się kontekstach. Na benchmarkach akademickich BaseModel uzyskuje ponad 100% poprawy względem HSTU firmy Meta oraz najlepsze wyniki w 10 z 12 zadań RelBench. W międzynarodowych konkursach metody konsekwentnie plasowały się w czołowej trójce, obok zespołów z DeepMind, Baidu i NVIDIA, a rozwiązanie zaprezentowane na WSDM WebTour Challenge zdobyło nagrodę za najlepszą pracę (Best Paper Award). W środowisku produkcyjnym system przetwarza 42 miliardy zdarzeń miesięcznie w ponad 200 organizacjach, dostarczając mierzalnych wzrostów zaangażowania i konwersji. Uzyskane wyniki pokazują, że wybór reprezentacji ma większe znaczenie niż wybór architektury: proste sieci jednokierunkowe operujące na szkicach dystrybucyjnych przewyższają złożone architektury sekwencyjne i grafowe, które odrzucają tę samą strukturę, którą próbują modelować.

Słowa kluczowe: systemy rekomendacyjne, estymacja gęstości, reprezentacja wielomodalna, modele fundacyjne, haszowanie wrażliwe na lokalność, modelowanie behawioralne

Contents

1 Introduction

Recommender systems have become infrastructure we no longer notice. They decide which products appear when you open Amazon, which videos YouTube plays next, which posts surface in your social media feed. The scale of their influence is remarkable: YouTube's recommendation algorithm drives over 70% of the platform's total watch time [44], while Netflix reports that roughly 80% of hours streamed originate from personalized suggestions rather than direct search [22]. A recent study projects that two trillion dollars in revenue will shift toward companies that excel at personalization over the next five years [6]. The commercial stakes have driven decades of research into predicting user preferences from observed behavior.

Yet despite this maturity, a fundamental question remains unresolved: how should we represent what a user wants?

The dominant answer borrows from natural language processing. If predicting the next word in a sentence works brilliantly for language, perhaps predicting the next item in a browsing session will work for behavior. This analogy has shaped modern recommendation research. Recurrent networks process interaction sequences step by step [30]. Transformers apply self-attention across browsing histories [33]. The largest industrial systems, including Meta's trillion-parameter HSTU [88], treat user actions as tokens in a new modality, importing the entire machinery of generative language modeling.

The analogy has merit because sentences have grammar: word order encodes syntactic and semantic structure that profoundly affects meaning. "Dog bites man" differs from "man bites dog." And for some behavioral domains, order matters similarly. When users progress through educational content, follow narrative arcs in media consumption, or engage in tightly coupled multi-step tasks, the order of actions carries genuine information. A student must learn calculus before differential equations. A viewer watching a television series experiences episodes in intended sequence. In these domains, sequential modeling captures real structure.

But many high-value commercial settings sit in a different regime. Consider a user who adds shampoo, a novel, and a phone charger to their shopping cart. The order typically reflects website layout, search ranking, or simple chance. The user did not intend a meaningful progression from personal care to literature to electronics. Purchase timestamps often reflect logistics, promotions, or inventory availability rather than preference evolution. Users batch unrelated items into single sessions. Ratings are entered weeks after consumption. In these common scenarios, what matters is knowing *which* items a user interacted with and how recent those interactions were, not the precise sequence of transitions from one item to the next. Yet sequential models treat all orderings as informative, learning patterns from what may be noise.

This thesis presents an alternative perspective grounded in density estimation. Rather than asking "what sequence of items did this user interact with," I ask "what regions of item space has this user occu-

pied." A user who purchased running shoes, fitness trackers, and protein supplements has not traced a trajectory through product space; they have indicated which regions of that space match their preferences. Representing this history as a probability distribution, rather than as an ordered list or a single averaged point, preserves structure that other representations destroy.

My prior work on attention mechanisms provided early motivation for this perspective [14]. When investigating memory-augmented neural language models with access to extended context windows, I found that models predominantly attended to only the five most recent positions, regardless of how much context was available. A simple concatenation of recent outputs matched sophisticated attention mechanisms. Even when architectures are explicitly designed for long-range dependencies, learned models collapse to local context. This observation suggested that the problem might not be architectural. If attention mechanisms struggle to leverage extended context even when designed to do so, perhaps the meaningful signal in much behavioral data does not reside in sequential structure at all.

1.1 The Representational Problem

To understand why representation matters, consider what happens when a user's interaction history must become input to a neural network. The user has engaged with some set of items, perhaps dozens, perhaps thousands. Each item has an embedding vector positioning it in a learned space where similar items cluster together. The network needs a fixed-size input. How should variable-length histories be converted?

The standard solution is averaging. Take all item embeddings from the user's history and compute their mean. This produces a single vector regardless of history length, which is computationally convenient. But consider a user who purchased both a laptop and a romance novel. The laptop embedding sits in one region of the space, among electronics. The novel embedding sits elsewhere, among books. Their average falls somewhere between, perhaps in a region corresponding to kitchen appliances or sporting goods, categories the user never expressed interest in. The averaged embedding represents a phantom preference that exists nowhere in the user's actual history.

This problem worsens with diverse histories. A photography enthusiast who buys only camera equipment and a casual shopper who bought one camera among many unrelated items might produce nearly identical mean embeddings if their purchases happen to balance to the same centroid. Yet these users will behave quite differently, and a recommender should treat them differently.

Sequential models offer a different answer: process the history recurrently or with attention, letting the architecture learn what matters. This avoids the averaging problem but introduces the ordering assumptions discussed above. More fundamentally, as my attention span research demonstrated, even sophisticated architectures may not actually leverage the sequential structure they are designed to capture.

Graph neural networks provide yet another perspective, representing users and items as nodes in an interaction graph and propagating information through neighborhood aggregation [77, 81]. But compressing a user's neighborhood into a fixed-size embedding through message passing can lose the very distributional information needed for personalization. A user connected to both photography equipment and romance novels gets an embedding somewhere between these categories, echoing the averaging problem in a different form.

The problem compounds when we consider the full richness of real-world data. A product is not merely an identifier in a transaction log; it has a title, description, images, category hierarchy, and price. Each signal captures something different, and each lives in a different representational space. Current multimodal approaches treat fusion as an architectural challenge, designing specialized encoders and learned combination mechanisms for each modality. The resulting systems grow complex, and adding a new signal source requires revisiting architectural decisions. An ideal representation would handle heterogeneous modalities as naturally as it handles interaction histories.

1.2 Density Estimation as an Alternative

The alternative developed in this thesis treats behavioral histories as samples from preference distributions. To build intuition, consider an analogy with geographic data. Suppose you want to represent a person's travel history. You could compute the average latitude and longitude of all places they visited, but this would be nearly useless. Someone who splits time between Tokyo and Paris would average to somewhere in Central Asia, a place they have never been.

A heatmap works better. Mark each visited location on a globe, with intensity proportional to visit frequency. The result shows which regions the person actually frequents. Tokyo and Paris both appear; Central Asia stays dark. The representation preserves distributional structure that averaging destroys.

The same logic applies to item embeddings. A user interested in both electronics and books should have a bimodal profile over product space, not a single point awkwardly positioned between unrelated categories. Representing histories as distributions over the item manifold preserves the multimodal structure of real preferences.

The challenge is computational. Geographic heatmaps work because the Earth's surface is two-dimensional with natural divisions like countries and cities. Embedding spaces have hundreds of dimensions with no pre-existing regions. Naive histogram approaches require exponentially many buckets as dimensionality grows. Kernel density estimation becomes computationally expensive in high dimensions and produces function outputs rather than fixed-size vectors suitable for neural network input.

EMDE (Efficient Manifold Density Estimator), developed collaboratively at Synerise [12] and extended in this thesis, combines ideas from locality-sensitive hashing and streaming algorithms. Random hyperplanes partition the embedding space into regions, with thresholds positioned at data quantiles rather than at zero. This density-dependent positioning ensures that partitions cut through regions where items actually exist, allocating representational capacity where it matters rather than wasting it on empty space. Multiple independent partitionings provide resolution that grows multiplicatively while storage grows only linearly. The result is a fixed-size sketch that approximates the density of items across the embedding manifold.

These sketches have properties that matter for practical systems. They are additive: when a user interacts with a new item, its contribution can be added to the existing sketch without reprocessing the entire history. They have fixed size regardless of history length: a user with ten purchases and a user with ten thousand produce profiles of identical dimension. They integrate multiple modalities naturally: collaborative signals, text embeddings, and image embeddings each produce their own sketches, which combine through simple concatenation without requiring learned fusion architectures. And they support

cold-start items: any product with content features can be positioned in the sketch immediately, without waiting for interaction data to accumulate.

Most importantly for multimodal settings, the same sketching procedure applies uniformly across modalities: collaborative embeddings, text embeddings, and image embeddings each produce density sketches that combine through simple concatenation, without learned fusion architectures or modality-specific design decisions.

1.3 From Representation to Foundation Model

Representing behavioral histories as density sketches addresses the aggregation problem but raises another question. The sketches capture where a user has been in item space, but recommendation requires predicting where they will go. Preferences evolve. A user browsing baby products today will need toddler supplies in two years. Someone who just purchased a camera may soon want lenses and accessories. The trajectory through preference space, not just the current position, determines what a user will want next.

BaseModel, the framework I introduce in this thesis, learns these trajectories through self-supervised training. The objective is conceptually straightforward: given a user's past behavioral distribution encoded as a density sketch, predict their future behavioral distribution, also encoded as a sketch. No task-specific labels are required. The supervision comes entirely from the temporal structure of behavioral data itself.

This framing treats preference evolution as distributional shift rather than as a sequence of discrete item transitions. The model learns how probability mass tends to migrate across item space as time passes. A user concentrated in the running-shoe region might expand toward general fitness equipment. A user exploring multiple categories might consolidate toward a few favorites. These patterns of distributional change, learned from millions of users, transfer to predicting behavior for any individual.

Once trained, the same backbone supports diverse downstream tasks through lightweight task-specific heads. For recommendation, the predicted future sketch is decoded into item scores through efficient bucket lookups. For classification tasks like churn prediction, a linear layer maps the behavioral profile to class probabilities. For regression tasks like lifetime value estimation, a similar head produces continuous outputs. The underlying representation is shared across tasks, with only the decoder differing between objectives.

I should clarify what I mean by "foundation model" in this context, as the term has become overloaded. In natural language processing, foundation models evoke massive transformer architectures with emergent capabilities and broad generalization across arbitrary tasks. I use the term more specifically: BaseModel employs self-supervised pretraining on behavioral data, learns a reusable backbone representation, and transfers to multiple downstream tasks with minimal task-specific engineering. I do not claim universal cross-domain generalization without adaptation, nor capabilities analogous to large language models. The contribution is demonstrating that behavioral data can support a foundation model paradigm at all, with the associated benefits of reduced task-specific engineering and improved sample efficiency on downstream tasks.

The methods developed in this thesis have moved from research into commercial deployment. EMDE and the BaseModel framework now serve as the technical foundation for Synerise's personalization infras-

tructure, processing over 42 billion behavioral events monthly for enterprise clients including BNP Paribas, mBank, Orange Polska, Empik, Modivo, and Nike across more than 150 markets [18]. In January 2023, the methodology was released commercially as BaseModel.ai, with subsequent deployments spanning fashion e-commerce, banking, and telecommunications sectors. This cross-industry adoption provides validation of deployment requirements that academic benchmarks cannot address: infrastructure flexibility, production-scale latency, and the stringent regulatory compliance demanded by financial services. This trajectory from doctoral research to production systems provides validation that controlled benchmarks cannot: evidence that the methods work under real-world constraints where latency budgets are measured in milliseconds, data arrives continuously, and business outcomes determine success.

1.4 Research Objectives

This dissertation pursues four interconnected research objectives that collectively establish density estimation as a foundation for behavioral data modeling.

RO1: Develop a representational framework that captures distributional properties of behavioral data. Standard approaches represent user histories either as sequences of item identifiers or as averaged embedding vectors. Both lose information: sequences impose ordering assumptions that may not reflect actual preference structure, while averaging collapses multimodal preferences into meaningless centroids. This objective investigates whether representing behavioral histories as probability distributions over learned embedding manifolds preserves information that point-based and sequential representations discard.

RO2: Enable seamless integration of heterogeneous data modalities within a unified representation. Real-world behavioral systems involve diverse signals: interaction graphs capturing collaborative patterns, textual descriptions encoding semantic content, images conveying visual attributes, and categorical metadata providing structured features. Current multimodal recommendation approaches require complex fusion architectures with modality-specific components. This objective explores whether density-based representations can accommodate multiple modalities through simple concatenation, without requiring learned fusion mechanisms.

RO3: Establish whether self-supervised learning on behavioral dynamics enables transfer across diverse prediction tasks. Foundation models in language and vision achieve broad applicability by learning general-purpose representations that transfer to downstream tasks. This objective investigates whether a self-supervised objective based on predicting future behavioral distributions from past distributions can learn transferable structure, enabling a single pretrained model to support recommendation, classification, and regression tasks.

RO4: Validate the practical applicability of density-based behavioral modeling through competitive benchmarks and production deployment. Academic benchmarks provide controlled evaluation but may not reflect real-world conditions. This objective seeks validation through

machine learning competitions against diverse methods and through production deployment measuring actual business impact. Strong performance across both settings would demonstrate that density-based approaches offer not only theoretical advantages but practical utility.

1.5 Research Hypotheses

The research objectives give rise to testable hypotheses, each predicting measurable outcomes and specifying how evaluation will proceed.

H1: Density-based representations outperform point-based aggregation for behavioral modeling. User behavioral histories are better characterized as distributions over item embedding space than as single point vectors. Representing histories as density sketches preserves structural information that mean-pooling destroys. This hypothesis predicts that EMDE sketches will achieve higher recommendation accuracy than averaged embeddings, and that performance will degrade significantly when geometric structure is removed from input embeddings through randomization.

Evaluation: Chapter 5 tests this through ablation studies comparing EMDE against mean-pooled baselines with identical input features, and by examining performance degradation when structured embeddings are replaced with random vectors lacking geometric properties.

H2: Density sketching enables effective multimodal integration through simple concatenation. Different data modalities capture complementary aspects of user-item relationships in geometrically distinct embedding spaces. Density-based sketching allows these signals to be integrated through concatenation of modality-specific sketches, without learned fusion architectures. This hypothesis predicts consistent performance improvements when modalities are added, with largest gains when modalities capture genuinely complementary information.

Evaluation: Chapter 5 tests this on session-based and top-k recommendation benchmarks by comparing unimodal against multimodal EMDE configurations. Chapter 6 extends evaluation to relational prediction tasks, ablating contributions of metadata, text, and visual features across multiple databases.

H3: Density-based input representations reduce the benefits of architectural complexity. When input representations already capture distributional structure, complex neural architectures yield diminishing returns. This hypothesis predicts that shallow feed-forward networks operating on EMDE sketches will achieve performance competitive with deep sequential models and graph neural networks while requiring substantially less training time.

Evaluation: Chapters 5 and 6 compare EMDE with simple feed-forward architectures against recurrent models (GRU4Rec, NARM), attention-based models (SASRec, BERT4Rec), and graph neural networks (SR-GNN, TAGNN) under comparable evaluation protocols. Chapter 7 provides additional evidence through competition results where shallow architectures achieved top placements against sophisticated alternatives.

H4: Self-supervised pretraining on behavioral dynamics enables transfer across task types. Behavioral data exhibits regularities in how preference distributions evolve that generalize across users and prediction tasks. A self-supervised objective predicting future distributions from past distributions captures these regularities. This hypothesis predicts that models initialized from a pretrained backbone will converge faster and achieve better final performance than identical architectures trained from scratch, even for classification and regression tasks that differ structurally from the recommendation-style pretraining objective.

Evaluation: Chapter 6 tests this through learning curve analysis on recommendation tasks (Amazon Beauty) and classification tasks (rel-hm churn prediction), comparing pretrained initialization against random initialization. The hypothesis is further evaluated by measuring performance across recommendation, classification, and regression tasks on RelBench using a single pretrained backbone with task-specific output heads.

H5: Density-based representations provide disproportionate advantages in cold-start scenarios. Content features such as text descriptions, images, and categorical metadata can position new items in the embedding manifold immediately, without requiring interaction history. Density-based representations leverage this positioning through the same inference mechanism used for established items, rather than requiring separate cold-start handling. This hypothesis predicts that adding content modalities will enable non-trivial recommendation performance on previously unseen items, whereas interaction-only approaches will achieve zero cold-start recall.

Evaluation: Chapter 6 tests this on RelBench datasets with documented cold-start rates ranging from 18% to 71% of test items unseen during training, comparing full multimodal profiles against interaction-only baselines and measuring rates at which new items appear in top-k recommendations.

1.6 Contributions

This dissertation makes six contributions:

1. **A distributional representation of behavioral histories.** EMDE [12] transforms variable-length sets of item embeddings into fixed-size density representations through locality-sensitive hashing with density-dependent partitioning. Hyperplane thresholds positioned at data quantiles ensure balanced partitions that allocate representational capacity where items actually exist. This thesis contributes the theoretical grounding for why this approach preserves distributional structure, and comprehensive experimental validation on session-based and top-k recommendation benchmarks.

2. **A self-supervised framework for learning behavioral dynamics.** BaseModel predicts future behavioral distributions from past distributions, learning patterns of preference evolution without task-specific supervision. The network uses factorized linear layers inspired by factorization machines [53] to capture pairwise cross-modal interactions within the concatenated sketch. The same pretrained backbone supports recommendation, classification, and regression through lightweight task-specific heads, demonstrating that behavioral representations can generalize across prediction problems.

3. **A unified approach to multimodal integration.** Different data modalities, including interaction graphs, text descriptions, images, and categorical metadata, integrate through concatenation of modality-specific density sketches without requiring learned fusion architectures. The network learns implicitly how to weight modalities through the prediction objective.

4. **Analysis of failure modes in temporal graph networks.** I demonstrate that existing temporal graph architectures produce saturated predictions under global temporal dynamics, collapsing to popularity-based ranking that cannot discriminate among plausible candidates. A trivial popularity-tracking baseline outperforms sophisticated neural methods on recent benchmarks. This analysis explains why density-based approaches with non-contrastive training avoid these failure modes.

5. **Validation across complementary settings, including leadership of three international competitions.** Evidence spans controlled academic benchmarks, production deployment with substantial engagement improvements, and three international machine learning competitions where I led the research and served as first author on all resulting publications. The solutions placed 2nd at the Booking.com WSDM Data Challenge (with Best Paper Award), 3rd at the KDD Cup behind only Baidu and Google DeepMind among approximately 150 teams, and 2nd at the ACM RecSys Challenge behind only NVIDIA. Achieving consistent top-three placements against the strongest industrial research labs, across domains as different as travel, academic citation networks, and social media, without domain-specific architectural changes, provides unusually strong evidence for the generality of density-based methods.

6. **Open implementation and commercial deployment.** The methods are available as open-source software and have been integrated into production systems processing 42 billion events monthly for over 200 organizations across 150 markets. Production results include fourfold improvement in customer retention prediction at a major European bank and consistent engagement gains across retail and telecommunications deployments, providing evidence that the approach meets accuracy, latency, and regulatory constraints that benchmarks cannot test.

1.7 Validation

Claims about representation require empirical support across settings that test different aspects of generalization. This thesis provides validation through three complementary channels.

Academic benchmarks. On standard recommendation datasets, BaseModel improves over state-of-the-art methods by substantial margins. Against TIGER, Google DeepMind's generative retrieval model, improvements range from 54% to 82% on sparse Amazon datasets. Against HSTU, Meta's trillion-parameter sequential architecture, BaseModel achieves over 100% higher hit rate on Amazon Books despite HSTU's architectural innovations specifically designed for behavioral sequences. On RelBench relational prediction tasks spanning recommendation, classification, and regression, BaseModel matches or exceeds specialized graph neural networks on 10 of 12 tasks using a single pretrained backbone.

Machine learning competitions. Competitions provide harsher tests than academic benchmarks: hidden test sets, strict deadlines, and comparison against teams from leading research organizations. During this doctoral research, the methods developed here achieved consistent top-three placements:

- **2nd place** in the Booking.com WSDM Data Challenge 2021 for multi-destination travel prediction, with Best Paper Award at the WebTour workshop
- **3rd place** in the KDD Cup 2021 among approximately 150 teams, behind only Baidu and DeepMind, on node classification in a 244-million node graph
- **2nd place** in the ACM RecSys Challenge 2021, behind only NVIDIA, on Twitter engagement prediction under strict latency constraints

These results span domains from travel to academic citations to social media, achieved without domain-specific architectural innovations.

Production deployment. The methods were integrated into a commercial personalization platform where they now power predictive analytics for enterprise clients, processing over 42 billion behavioral events monthly across more than 200 organizations in 150 markets [18]. Controlled A/B testing across multiple deployments demonstrated consistent improvements: a fashion e-commerce platform achieved 21% higher open rates, 18% higher click-through rates, and 7% higher conversion rates; a major European bank reported fourfold improvement in customer retention prediction accuracy compared to legacy heuristic models [66]; and a telecommunications provider increased digital channel sales by 2.1% annually through AI-driven personalization [65].

In January 2023, the methodology was released as a commercial product, BaseModel.ai, available through Microsoft Azure Marketplace, Snowflake Marketplace, and as containerized deployments for on-premises installation in regulated industries [67]. The platform has since been adopted by organizations including BNP Paribas, mBank, Orange Polska, Empik, Modivo, and Nike, spanning banking, retail, telecommunications, and fashion sectors [4, 64]. This cross-industry adoption demonstrates that the methods satisfy requirements academic evaluation cannot address: deployment flexibility across infrastructure environments, production-scale latency, and operation within the regulatory constraints of financial services. The trajectory from doctoral research to production systems serving millions of users provides evidence that the methods work under real-world constraints where latency budgets are measured in milliseconds and business outcomes determine success.

1.8 Publications

The research in this dissertation has produced peer-reviewed publications at major venues.

Conference Publications

1. **M. Daniluk**, T. Rocktäschel, J. Welbl, S. Riedel. *Frustratingly Short Attention Spans in Neural Language Modeling*. International Conference on Learning Representations (ICLR), 2017. **Citations: 162 |**

MEiN: 200 points Foundational work on attention mechanism limitations that motivated the non-sequential approach developed in this thesis.

2. J. Dąbrowski, B. Rychalska, **M. Daniluk**, D. Basaj, K. Gołuchowski, P. Bąbel, A. Michałowski, A. Jakubowski. *An Efficient Manifold Density Estimator for All Recommendation Systems*. International Conference on Neural Information Processing (ICONIP), Springer LNCS vol. 13111, 2021. **Citations: 23 | MEiN: 140 points**

3. A. Wróblewska, J. Dąbrowski, M. Pastuszak, A. Michałowski, **M. Daniluk**, B. Rychalska, M. Wieczorek, S. Sysko-Romańczuk. *Designing Multi-Modal Embedding Fusion-Based Recommender*. Electronics 11(9), 2022. **Citations: 14 | MEiN: 100 points**

4. **M. Daniluk**, M. Spytek, D. Matlak, J. Dąbrowski. *BaseModel: A Foundation Model for Behavioral Data*. Under review at ACM RecSys 2026. **MEiN: 140 points**

Workshop Publications and Competition Papers

All workshop papers listed below underwent full peer review processes at their respective venues.

1. **M. Daniluk**, B. Rychalska, K. Gołuchowski, J. Dąbrowski. *Modeling Multi-Destination Trips with Sketch-Based Model*. ACM WSDM WebTour Workshop, 2021. **2nd place, Booking.com Data Challenge. Best Paper Award, Citations: 5**

2. **M. Daniluk**, J. Dąbrowski, B. Rychalska, K. Gołuchowski. *Synerise at KDD Cup 2021: Node Classification in Massive Heterogeneous Graphs*. KDD Cup Workshop, 2021. **3rd place among approximately 150 teams, Citations: 3**

3. **M. Daniluk**, J. Dąbrowski, B. Rychalska, K. Gołuchowski. *Synerise at RecSys 2021: Twitter User Engagement Prediction with a Fast Neural Model*. ACM RecSys Challenge Workshop, 2021. **2nd place, behind NVIDIA, Citations: 10**

4. **M. Daniluk**, J. Dąbrowski. *Temporal Graph Models Fail to Capture Global Temporal Dynamics*. Temporal Graph Learning Workshop at NeurIPS, 2023. **Citations: 7.**

5. B. Rychalska, D. Basaj, J. Dąbrowski, **M. Daniluk**. *I Know Why You Like This Movie: Interpretable Efficient Multimodal Recommender*. ML4MD Workshop at ICML, 2020. **Citations: 4.**

The publications have accumulated over 220 citations. I served as first author on six works.

1.9 Statement of Contributions

This dissertation presents research conducted between 2020 and 2026 as part of an industrial doctorate program at Synerise S.A. under academic supervision from Warsaw University of Technology. This section clarifies my individual contributions.

Primary contributions. **BaseModel** (Chapter 6) represents the primary original contribution. I conceptualized the foundation model approach for behavioral data, including the formulation of self-supervised pretraining objectives based on temporal distribution prediction. This objective is meaningfully distinct from contrastive and masked-item alternatives: by requiring the model to predict full future distributions over B buckets per partitioning, it provides dense gradients across the entire behavioral manifold at every training step, learning global preference dynamics rather than item co-occurrence statistics. I also designed FMLinear, a factorized projection layer that augments standard linear projections with low-rank pairwise interaction terms, enabling the network to capture joint activations across sketch components from different modalities and partitionings. I designed the unified architecture supporting recommendation, classification, and regression through a shared pretrained backbone, implemented and conducted all experiments on sequential recommendation benchmarks and RelBench relational database tasks, and led production deployment at a major European retailer achieving 21% improvement in email engagement metrics.

Competition solutions (Chapter 7) were led entirely by me across all three challenges. I designed the system architectures, implemented the core algorithms, and was first author on all resulting publications. Each solution adapted density-based methods to a fundamentally different domain: multi-destination travel prediction (Booking.com), node classification in a 244-million node academic graph (KDD Cup), and real-time social media engagement prediction under strict latency constraints (RecSys Challenge). The teams placed immediately above us were Baidu, Google DeepMind, and NVIDIA, respectively; the Booking.com solution also received the Best Paper Award. The breadth of these results, spanning three top-tier venues and three unrelated application domains, demonstrates that the density-based framework generalizes beyond any single benchmark setting.

Contributions to EMDE. **EMDE** (Chapter 5) was developed collaboratively at Synerise. The overall CMS+LSH combination framework was conceived by Dąbrowski [12]. My contributions were: proposing density-dependent partitioning, the key modification that positions hyperplane thresholds at data quantiles rather than at zero, which is what makes EMDE a manifold density estimator rather than a standard LSH hash table; developing the theoretical grounding connecting this to manifold density estimation; designing and executing the experimental benchmarks on session-based and top-k recommendation datasets; and developing the analysis explaining why density-based representations succeed where temporal graph networks fail, including the PopTrack diagnostic baseline (published as first author at the NeurIPS 2023 Temporal Graph Learning Workshop).

Background. **Cleora** (Chapter 4) was developed by colleagues at Synerise. My contribution was limited to applying Cleora embeddings as input features for EMDE and BaseModel. The chapter provides background necessary for understanding subsequent chapters but does not represent original work by me.

Prior foundational work. The ICLR 2017 work on attention mechanisms [14] predates this doctoral research but informed its direction. I designed and implemented the key-value-predict attention architecture, conducted experiments demonstrating that models attend predominantly to recent positions,

and showed that simple concatenation matches sophisticated attention mechanisms. While addressing language modeling rather than recommendation, this finding motivated the shift from sequential to distributional modeling that underlies this thesis.

1.10 Industrial Cooperation

This thesis was conducted through Poland's Implementation Doctorate programme (Doktorat Wdrożeniowy), a Ministry of Science and Higher Education initiative launched in 2017 to support doctoral research with direct industrial applications. The programme enables researchers to pursue a doctoral degree while working at a company, with supervision from both an academic advisor and an industrial mentor.

The industrial partner, Synerise, is a Polish deep-tech company founded in 2012 and headquartered in Kraków. Synerise develops an AI-driven behavioral data platform that has grown to process over 42 billion customer events monthly, serving more than 200 enterprise organizations across 150 markets worldwide. The platform's technical infrastructure includes the Cleora embedding system, developed by colleagues, and the EMDE density estimator, developed collaboratively as described in Section 1.9. Both systems serve as foundations for the work presented in this thesis.

The industrial setting shaped this research in important ways. Access to production-scale data with millions of users and billions of interactions enabled development and validation of methods that academic benchmarks alone cannot test. The requirement to deploy models in real-time systems with strict latency constraints, typically under 6 milliseconds for inference, forced attention to computational efficiency throughout. The opportunity to measure impact through business metrics such as click-through rates and conversion provided evidence that offline experiments cannot establish.

The research has led to commercial outcomes. The methods are integrated into Synerise's platform as the foundation for personalization capabilities. The BaseModel framework was commercialized as a standalone product, BaseModel.ai, launched in January 2023 and available through cloud marketplaces and as containerized deployments for on-premises installation in regulated industries.

The dual validation through academic benchmarks and production deployment at scale provides stronger evidence for the methods' effectiveness than either channel alone. Academic experiments demonstrate controlled comparison against baselines; production deployment demonstrates that the methods work under real-world constraints with measurable business impact.

1.11 Thesis Organization

The remainder of this dissertation is organized as follows.

Chapter 2: Background. This chapter provides technical foundations necessary for understanding subsequent contributions. It covers recommender system paradigms and evaluation metrics, embedding methods for representing items and users, locality-sensitive hashing and sketching algorithms, and the basics of graph neural networks and temporal modeling. Readers familiar with these topics may skim or skip this chapter.

Chapter 3: Related Work. This chapter surveys existing approaches to behavioral modeling and recommendation. It reviews sequential recommendation methods, graph-based collaborative filtering,

multimodal fusion approaches, self-supervised and contrastive learning methods, and efforts to develop foundation models for recommendation. The chapter identifies a common failure mode across paradigms, the point representation bottleneck, and positions the density-based approach of this thesis as a direct response to it.

Chapter 4: Graph Embeddings with Cleora. This chapter introduces Cleora, a scalable method for computing embeddings from hypergraph-structured data. While Cleora was developed by colleagues at Synerise, understanding its properties is essential for subsequent chapters, as Cleora embeddings serve as input features for both EMDE and BaseModel.

Chapter 5: Efficient Manifold Density Estimator. This chapter presents EMDE, a framework for representing behavioral histories as probability distributions over embedding manifolds. It describes the density-dependent locality-sensitive hashing scheme, the sketch construction process, and the approach to multimodal fusion through concatenation. Experimental results demonstrate state-of-the-art performance on session-based and top-k recommendation benchmarks, and analysis reveals why temporal graph networks fail where density estimation succeeds.

Chapter 6: BaseModel: A Foundation Model for Behavioral Data. This chapter presents the primary contribution of this dissertation. It describes the BaseModel architecture, the self-supervised pretraining objective based on temporal distribution prediction, and the fine-tuning procedure for downstream tasks. Experiments on sequential recommendation benchmarks and RelBench relational database tasks demonstrate that a single pretrained model can match or exceed specialized baselines across recommendation, classification, and regression tasks.

Chapter 7: Validation Through Competitions. This chapter describes the application of EMDE and related methods to three machine learning competitions in 2021: the Booking.com WSDM WebTour Challenge for multi-destination trip prediction, the KDD Cup MAG240M-LSC task for node classification in a 244-million node graph, and the ACM RecSys Challenge for Twitter engagement prediction. The chapter details the technical solutions and discusses what the competition results reveal about the generality of density-based approaches.

Chapter 8: Conclusions. The final chapter summarizes the contributions of this dissertation, evaluates hypotheses against accumulated evidence, discusses limitations of the proposed methods, and outlines directions for future research.

2 Background

The methods developed in this thesis are a response to specific assumptions embedded in the current mainstream of recommendation research: that user behavior is best modeled as a sequence, and that interaction histories are best represented as single points in embedding space. Both assumptions are convenient. Neither holds up under scrutiny. This chapter lays out what the field has built on those assumptions, where the foundations crack, and why a distributional alternative is worth pursuing. Readers already familiar with collaborative filtering, content-based methods, and representation learning may skip ahead to Chapter 3.

2.1 Recommender Systems: An Overview

A recommender system answers a simple question: out of millions of available items, which ones should be shown to this particular user right now? Manual browsing does not scale to catalogs of that size, so systems must filter and rank automatically, drawing on whatever signals they have about user preferences and item characteristics.

Formally, the recommendation problem can be stated as follows. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ denote a set of users and $\mathcal{I} = \{i_1, i_2, \ldots, i_n\}$ a set of items. Users interact with items in various ways: they may purchase products, rate movies, click on articles, or listen to songs. These interactions are recorded in an interaction matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$, where entry r_{ui} represents the observed interaction between user u and item i. In explicit feedback settings, r_{ui} might be a numerical rating on a scale from 1 to 5 stars. In implicit feedback settings, which are more common in practice, r_{ui} might be binary (1 if the user interacted with the item, 0 otherwise) or a count of interactions.

The core challenge is that $\mathbf{R}$ is extremely sparse. A typical user interacts with a tiny fraction of available items, often less than 0.01% of the catalog. The goal of a recommender system is to predict the missing entries of $\mathbf{R}$: given the sparse observations, estimate which unobserved user-item pairs would yield positive interactions if presented to the user. These predictions then determine which items to recommend.

Two dominant paradigms have emerged for making such predictions: collaborative filtering, which exploits patterns in collective user behavior, and content-based filtering, which leverages item attributes. The following subsections review each approach and their combination in hybrid systems.

2.1.1 Collaborative Filtering

Collaborative filtering exploits a simple pattern: users who agreed in the past tend to agree in the future. If Anna and Bob both loved the same five films, and Anna enjoyed a sixth that Bob has not seen, Bob will

probably like it too. Notice that this reasoning requires no knowledge of what the films are actually about. We are exploiting regularities in collective behavior, not properties of the items themselves.

Early systems implemented this directly through neighborhood methods [54, 58]. Find users similar to the target user, see what they liked, recommend accordingly. Similarity was typically cosine distance or Pearson correlation on rating vectors. This works and has the virtue of interpretability: you can tell a user "people like you enjoyed X." But it scales poorly. Computing similarities against all other users for every recommendation becomes expensive as the user base grows.

The Netflix Prize in 2006 pushed the field toward matrix factorization [36]. The key insight is that user-item interactions form a matrix, mostly empty, and we can try to fill in the blanks by assuming low-rank structure. If preferences are driven by a modest number of latent factors, perhaps a few dozen, then we can represent each user and each item as a short vector and predict ratings as dot products. Formally, we seek matrices $\mathbf{P} \in \mathbb{R}^{m \times d}$ and $\mathbf{Q} \in \mathbb{R}^{n \times d}$ such that $\mathbf{R} \approx \mathbf{P}\mathbf{Q}^\top$, where m is the number of users, n the number of items, and d the dimensionality of the latent space. Each row $\mathbf{p}_u \in \mathbb{R}^d$ of $\mathbf{P}$ is the latent representation of user u, and each row $\mathbf{q}_i \in \mathbb{R}^d$ of $\mathbf{Q}$ is the latent representation of item i. Training minimizes reconstruction error on the set of observed entries $\mathcal{O}$:

$$\min_{\mathbf{P},\mathbf{Q}} \sum_{(u,i)\in\mathcal{O}} (r_{ui} - \mathbf{p}_u^\top \mathbf{q}_i)^2 + \lambda(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2) \tag{2.1}$$

where r_{ui} is the observed rating of user u for item i, and $\|\cdot\|_F$ denotes the Frobenius norm (the square root of the sum of squared entries of a matrix). The regularization term, controlled by the hyperparameter λ, prevents overfitting to the sparse observations.

Matrix factorization won Netflix and remains surprisingly competitive today. More importantly for this thesis, it established the embedding paradigm that now dominates the field. Users and items become vectors in a shared latent space. Compatibility is geometric: dot products, cosine similarities, learned distance functions. Nearly all modern recommendation research builds on this foundation, including the methods I develop in later chapters.

2.1.2 Content-Based Filtering

Collaborative filtering has an obvious weakness: it knows nothing about items except who interacted with them. A new product with no purchase history cannot be recommended, no matter how well it matches a user's tastes. Content-based filtering takes the opposite approach. Instead of relying on other users' behavior, it recommends items similar to those a user has liked before, where similarity is defined through item attributes [48].

Consider a user who has purchased several books on machine learning. A content-based system would examine these books, extract features like topic, author, publisher, and keywords, then search the catalog for other books with similar features. The system builds a profile of what the user likes and matches it against item descriptions. No other users are involved in this reasoning.

The approach requires some way to represent item content. Early systems relied on manual tagging or keyword extraction from text descriptions. Modern systems use learned representations: word embeddings for text [43], convolutional networks for images [26], or pretrained language models for richer semantic

understanding [16]. A user profile is then constructed by aggregating the representations of items they have engaged with, and new items are scored by similarity to this profile.

Content-based methods solve the new item problem that plagues collaborative filtering. A book entering the catalog today can be recommended immediately based on its description, without waiting for readers to discover it. But this strength comes with a cost. Content-based systems tend to recommend more of the same. A user who bought machine learning books will see more machine learning books, never the novel that readers with similar taste unexpectedly loved. The surprising discoveries that collaborative filtering enables, where items are connected through behavioral patterns invisible in their descriptions, are lost.

2.1.3 Hybrid Approaches

The complementary weaknesses of collaborative and content-based filtering naturally suggest combining them. If collaborative filtering fails on new items but excels at capturing behavioral patterns, and content-based filtering handles new items but misses collaborative signals, why not use both?

Hybrid recommender systems do exactly this [7]. The simplest approach switches between methods: use content-based recommendations for new items, collaborative filtering once enough interactions accumulate. More sophisticated systems blend predictions from both paradigms, either through weighted averaging or learned combination functions. More recent neural architectures process interaction histories, text descriptions, and images jointly through unified encoders, bypassing the need to assign each signal a label [89].

Whether simple or sophisticated, hybrid systems share a common limitation: they treat the combination of content and behavior as an engineering problem to be solved through architecture design. The question is always how to fuse two different kinds of signal. This framing, while practically useful, may miss something deeper. Content and behavior are not separate signals to be merged; they are two views of the same underlying structure. A user who buys running shoes and protein bars is expressing preferences that live simultaneously in product-attribute space and in behavioral space. The challenge is not fusion but representation: finding a way to capture both views without losing either. This perspective motivates the density-based approach developed later in this thesis.

2.2 Notation

Table 2.1 summarizes core notation used throughout this thesis. Additional notation specific to graph embeddings, density sketches, and the foundation model framework is introduced in the relevant chapters.

2.3 Fundamental Challenges

The previous section presented collaborative and content-based filtering as two paradigms with complementary strengths. But both share deeper problems that no simple combination can resolve. These challenges are not implementation difficulties that better engineering might fix. They are structural, arising from the nature of behavioral data itself.

Table 2.1: Summary of core notation.

Symbol	Description
Sets and Indices	
$\mathcal{U}, \mathcal{I}$	Sets of users and items
$\lvert\mathcal{U}\rvert, \lvert\mathcal{I}\rvert$	Number of users and items
u, v	User indices
i, j	Item indices
t	Time index or timestamp
Interactions	
$\mathbf{R} \in \mathbb{R}^{\lvert\mathcal{U}\rvert \times \lvert\mathcal{I}\rvert}$	User-item interaction matrix
r_{ui}	Interaction value for user u and item i
$\mathcal{H}_u$	Interaction history for user u: set of (i, t) pairs
Embeddings	
d	Embedding dimensionality
$\mathbf{e}_i \in \mathbb{R}^d$	Embedding vector for item i
$\mathbf{e}_u \in \mathbb{R}^d$	Embedding vector for user u
Temporal Weighting	
$w(\Delta t)$	Temporal weighting function
λ	Temporal decay rate (continuous: $w(\Delta t) = \exp(-\lambda \Delta t)$)
α, γ	Temporal decay parameters (discrete: $w(\Delta t) = \alpha \cdot \gamma^{\Delta t}$)
Common Operations	
$\lVert\cdot\rVert_1, \lVert\cdot\rVert_2$	L_1 and L_2 norms
$\mathbf{1}[\cdot]$	Indicator function (1 if condition true, 0 otherwise)
$[\,;\,]$	Vertical concatenation of vectors or matrices

2.3.1 The Cold-Start Problem

The cold-start problem refers to the difficulty of making predictions when insufficient information is available [59]. It appears in three related forms: new users, new items, and new systems.

When a new user joins a platform, the system knows nothing about their preferences. Collaborative filtering requires comparing users based on shared interaction history, but a new user has no history to compare. The system faces a bootstrapping dilemma: it needs to observe behavior to make good recommendations, but the user may leave if initial recommendations are poor. Common workarounds include asking users to rate a few items during onboarding, inferring preferences from demographic information, or falling back to popularity-based recommendations. None of these solutions is satisfying. Explicit preference elicitation creates friction. Demographic inference relies on stereotypes that may not apply to individuals. Popularity-based recommendations ignore personalization entirely.

New items present the symmetric challenge. A product entering the catalog has no interaction history, so collaborative filtering cannot determine which users might want it. This creates a visibility problem: established items with rich interaction histories keep getting recommended, while new items struggle to attract initial attention [1]. For businesses with rapidly changing inventory, like fashion retailers with

seasonal collections or news platforms with time-sensitive content, this limitation has direct revenue impact.

Content-based methods partially address item cold-start. A new product can be matched to users based on its description, category, or image, without waiting for interactions. But content features alone miss the collaborative patterns that often matter most. Two products may have similar descriptions yet appeal to entirely different audiences. The behavioral signal that collaborative filtering captures is real and valuable; content-based methods simply cannot access it for new items.

The cold-start problem is not a temporary inconvenience that disappears as a system matures. Real platforms continuously onboard new users and add new items. Studies of production systems suggest that cold-start scenarios account for 20–40% of recommendation requests even on established platforms [75]. Any practical recommendation approach must handle cold-start as a core capability, not an edge case.

2.3.2 Data Sparsity

Even for users and items with interaction history, the available data is thin. Consider what a typical user actually does on an e-commerce platform. Over several years, they might purchase a few hundred items from a catalog containing millions. The ratio of observed interactions to possible interactions is vanishingly small, often less than 0.01%. Standard benchmark datasets, despite being curated for research, still have 95–99% of entries missing. Production systems are sparser still.

This sparsity creates problems at multiple levels. Most directly, it means there are no observations for most user-item pairs. Matrix factorization addresses this by learning low-rank structure that generalizes to unobserved entries, but the learned factors become unreliable as observations grow scarce. With fewer ratings per user and per item, there is simply less signal to extract.

Sparsity also undermines similarity computation. Neighborhood-based methods require meaningful overlap between users to compute similarity. If two users have each rated fifty items from a catalog of millions, the chance that they rated any item in common is tiny. Without overlap, similarity is undefined. Various heuristics address this, including default voting schemes and significance weighting, but they introduce assumptions that may not hold.

The sparsity problem interacts with cold-start in subtle ways. Popular items accumulate interactions faster than niche items, so their representations become more reliable. This creates a feedback loop: items with better representations get recommended more often, accumulating yet more interactions, further improving their representations [9]. Niche items that might perfectly match certain users never gain enough visibility to prove their value.

2.3.3 Scalability

The challenges above would be difficult enough on small datasets. Real systems operate at scales that stress even simple algorithms. A major e-commerce platform might serve hundreds of millions of users browsing tens of millions of products. Recommendations must be generated in milliseconds, refreshed as new interactions arrive, and personalized for each user.

Different algorithmic approaches hit different scaling walls. Neighborhood methods require computing similarities between the target user and all other users, or between candidate items and all items in the

user's history. Naive implementations scale quadratically, which becomes prohibitive at platform scale. Approximate nearest neighbor techniques and careful indexing help, but they add engineering complexity and introduce approximation errors [15].

Matrix factorization scales better in some respects. Once trained, the factor matrices are compact, and scoring an item requires only a dot product. But training on billions of interactions demands distributed computation. Updating the model as new data arrives presents additional challenges: full retraining is expensive, while incremental updates can drift from optimality.

Deep learning approaches introduce their own concerns. Embedding tables that map user and item identifiers to dense vectors can consume enormous memory. A system with a hundred million items using 256-dimensional embeddings requires tens of gigabytes just for item representations, before accounting for model parameters or user embeddings. Industrial systems at companies like Meta have grown to trillions of parameters, requiring infrastructure that few organizations can replicate [45].

For this thesis, scalability matters primarily as a design constraint. The approaches I develop must remain practical at production scale, which rules out methods requiring expensive inference or per-user model updates. Computational considerations are revisited when presenting the architecture in Chapter 5.

2.4 The Embedding Paradigm

The matrix factorization approach described in Section 2.1.1 introduced an idea that now dominates recommendation research: representing users and items as dense vectors in a shared latent space. This embedding paradigm has proven so successful that it is easy to forget it represents a choice, one with consequences that shape what recommendation systems can and cannot do.

2.4.1 From Sparse Features to Dense Representations

Before embeddings became dominant, recommendation systems operated on sparse, high-dimensional representations. A user might be described by a binary vector indicating which items they had purchased, with dimensionality equal to the catalog size. An item might be represented by a bag-of-words vector over product descriptions. These representations are interpretable but unwieldy. Computing similarity between million-dimensional sparse vectors is expensive, and the representations capture only surface-level co-occurrence rather than deeper semantic structure.

Embeddings compress this sparse information into dense vectors of manageable size, typically 64 to 512 dimensions. The compression is not arbitrary: it is learned to preserve predictive signal. Users with similar preferences end up nearby in the embedding space. Items that appeal to similar audiences cluster together. The geometry of the space encodes relationships that were implicit in the raw interaction data.

Matrix factorization learns these embeddings directly from the interaction matrix. Neural approaches extend the idea by passing sparse inputs through embedding layers, essentially lookup tables that map discrete identifiers to dense vectors [28]. A user ID indexes into a user embedding table; an item ID indexes into an item embedding table. These embeddings then feed into neural networks that learn nonlinear combinations and interactions.

The appeal is clear. Dense vectors are computationally convenient, enabling fast similarity computation through optimized linear algebra routines. The continuous nature of embeddings allows gradient-based optimization. Pretrained embeddings can transfer knowledge across tasks. And the geometric structure of embedding spaces can, at least in low dimensions, be visualized and interpreted [43].

2.4.2 The Aggregation Problem

Embeddings work naturally for individual items. But recommendation requires representing users, and users are defined by their interaction histories. A user who has purchased fifty items has fifty item embeddings. How should these be combined into a single user representation?

The standard solution is to average them. Take all item embeddings from a user's history, sum them up, and normalize. This gives a fixed-size vector regardless of how many items the user has interacted with, which makes it easy to feed into downstream models. The approach is simple and fast.

But averaging throws away information that matters. Consider a user who bought both a laptop and a novel. The laptop embedding sits in one region of the space, clustered with other electronics. The novel embedding sits in a different region, among books. When these two vectors are averaged, the result lands somewhere in between. That middle ground might correspond to kitchen appliances, or sporting goods, or nothing in particular. The user never expressed interest in any of these categories. The averaged embedding represents a phantom preference.

This problem becomes severe as user histories grow more diverse. Someone with broad interests, buying across many categories over several years, ends up with an average embedding that accurately represents none of those interests. The mean of a multimodal distribution tells you little about where the actual mass lies. Yet averaging implicitly assumes that preferences concentrate around a single point.

Researchers have proposed various fixes. One approach maintains multiple embeddings per user, trying to capture different facets of their interests [79]. Another uses attention mechanisms to weight items differently depending on what is being predicted [33]. These help, but they do not solve the underlying problem: a small set of point vectors struggles to faithfully represent how preferences are actually distributed across the item space.

2.4.3 Cold-Start Revisited

The embedding paradigm also creates a structural cold-start problem. An embedding is learned from data: the system observes how a user interacts with items and adjusts their vector to reflect those patterns. A new user with no interaction history provides nothing to learn from.

The typical fix is to assign new users a default embedding. This might be the average of all existing user embeddings, or a specially learned initialization vector. Either way, it provides a starting point that the system refines as interactions accumulate. But until those interactions arrive, the new user receives generic recommendations indistinguishable from any other newcomer. Personalization requires patience.

For new items, the situation looks symmetric but has an important difference. A product entering the catalog has no interaction-based embedding, just like a new user. However, products usually come with descriptions, images, categories, and other metadata. These content features can be processed through

neural encoders to produce an embedding, positioning the item in the space based on what it is rather than who has bought it.

This content-based fallback is valuable, but it solves only half the problem. Content embeddings and collaborative embeddings capture different things. A book's description might place it near other books with similar themes, but readers of those books might have completely different tastes. The collaborative signal, which products appeal to which audiences, is exactly what content cannot provide. Two embedding spaces with different semantics do not automatically align.

Some systems try to bridge this gap by learning a mapping from content space to collaborative space, or by training hybrid models that process both signals jointly [75]. These approaches work to varying degrees, but they treat the problem as one of alignment or fusion: two different representations that must somehow be combined. An alternative perspective, which I pursue in this thesis, is that content and behavior are not separate signals requiring fusion. They are two observations of the same underlying structure. The challenge is finding a representation that captures both naturally, without forcing them through separate pipelines that must later be reconciled.

2.5 Multimodal Data in Recommendation

The previous sections focused primarily on interaction data and content features as two distinct sources of information. Real recommendation systems operate with far richer inputs. A product in an e-commerce catalog has a title, a description, images, a category hierarchy, a price, customer reviews, and a history of purchases. A user has demographic attributes, browsing patterns, search queries, and perhaps social connections. Each of these signals captures something different about what the product is or what the user wants. The challenge lies not in scarcity of information but in its heterogeneity.

2.5.1 What Modalities Exist

The signals available in recommendation systems can be grouped into several broad categories.

Behavioral signals capture how users interact with items. Purchases, clicks, ratings, time spent, scroll depth, add-to-cart events, wishlists. These are the primary currency of collaborative filtering. They are also noisy: a user might click an item by accident, view a product with no intention of buying, or leave a tab open without reading. Distinguishing genuine interest from incidental interaction is not straightforward.

Textual content includes product titles, descriptions, reviews, and search queries. Text is semantically rich but requires processing to be useful. A product description might mention features, intended use cases, and comparisons to alternatives. Reviews aggregate opinions and highlight aspects that matter to different types of buyers. Search queries reveal explicit intent in ways that browsing behavior does not.

Visual content matters enormously in domains like fashion, furniture, and food. A user's attraction to a particular style, color, or aesthetic cannot be inferred from text alone. Product images, and increasingly video, carry information that other modalities miss entirely.

Categorical attributes provide structured metadata: brand, category, size, color, price range. These are typically well-organized in product catalogs and offer reliable signal for certain recommendation

scenarios. A user who consistently buys a particular brand or shops within a price range has preferences that categorical features capture directly.

Relational structure captures connections beyond user-item interactions. Users may follow other users. Items may belong to collections or be frequently purchased together. Knowledge graphs link items to external entities like directors, authors, or ingredients. These relationships form networks that contain patterns not visible in any single modality.

2.5.2 Why Fusion Is Hard

Given all these signals, the natural instinct is to use everything. More information should mean better predictions. In practice, combining modalities turns out to be surprisingly difficult.

Different modalities have different shapes. A text encoder might produce 768-dimensional vectors. An image encoder might output 2048 dimensions. Categorical features could be one-hot encoded into thousands of sparse dimensions or embedded into a few dozen dense ones. Behavioral histories vary in length from user to user. Simply concatenating these representations creates an unbalanced input where some modalities dominate by virtue of their dimensionality rather than their informativeness.

Compounding this, not every item has every modality. A new product might lack reviews. An older product might lack high-quality images. Some items have rich descriptions while others have only a title. A multimodal system must handle these gaps gracefully rather than failing when any input is absent.

Perhaps most troublesome is semantic misalignment. Each modality has its own notion of similarity. Two products might be similar in text, both described as "lightweight running shoes for beginners," but look completely different in images. They might be similar visually but appeal to different audiences based on behavioral patterns. Which notion of similarity matters depends on the user and the context. A fixed weighting of modalities cannot capture this variability.

These difficulties explain why most production systems still rely heavily on a single primary modality, usually behavioral data, with other signals added carefully and incrementally. The multimodal fusion problem is not solved; it is managed.

2.5.3 Current Approaches to Multimodal Recommendation

Researchers have explored several strategies for combining modalities [3].

Early fusion concatenates features from all modalities before any processing. Raw or lightly transformed features are joined into a single vector that feeds into a model. This is simple but requires all modalities to be present and places the entire burden of learning cross-modal relationships on the downstream model.

Late fusion processes each modality independently through specialized encoders, then combines the resulting representations. A text encoder handles descriptions, an image encoder handles photos, and a collaborative model handles interactions. Their outputs are merged through concatenation, averaging, or learned attention. This approach handles missing modalities more naturally since each encoder operates independently. But it may miss interactions between modalities that early fusion could capture.

Graph-based fusion represents items, users, and attributes as nodes in a heterogeneous graph, with different edge types for different relationships [78]. Graph neural networks then propagate information

across the structure, allowing modalities to influence each other through connectivity patterns. This is flexible but adds complexity and computational cost.

Despite the variety of approaches, a common thread runs through them: multimodal fusion is treated as an architectural problem. The question is always how to design networks that combine different inputs. Adding a new modality means choosing an encoder, a fusion strategy, and a policy for missing data. Each decision multiplies the number of hyperparameters and failure modes.

The approach I develop in this thesis takes a different perspective. Rather than designing fusion architectures, I seek a representation that treats all modalities uniformly from the start. If behavioral patterns, text, and images can all be encoded into the same kind of structure using the same machinery, then fusion becomes simple concatenation rather than a design challenge. Whether this perspective is viable is an empirical question that later chapters address.

2.6 Sequential vs. Distributional Views of Behavior

Over the past decade, the dominant way to model user behavior has been as sequence prediction. A user's interaction history is treated as an ordered list of items, and the task is to predict what comes next. This framing aligns recommendation with the successes of natural language processing, where predicting the next word given previous words has proven powerful. The analogy between sentences and shopping sessions has shaped much of modern recommendation research.

But analogies can mislead. User behavior is not language, and the assumptions that make sequence modeling effective for text may not transfer to behavioral data.

2.6.1 The Sequential Paradigm

Sequential recommendation treats a user's history as an ordered list. If a user has interacted with items $i_1, i_2, \ldots, i_t$ in that temporal order, the goal is to predict i_{t+1}. The problem becomes: given the sequence so far, what comes next?

This framing opened the door to architectures developed for language. The earliest methods used Markov chains [60], but these struggled with longer-range dependencies. Recurrent neural networks addressed this limitation by processing the sequence step by step, maintaining a hidden state that accumulates information from previous items [30]. The final hidden state summarizes the session and predicts the next item. GRU4Rec, introduced in 2015, demonstrated that this approach could substantially outperform non-sequential baselines on session-based recommendation tasks.

Attention mechanisms offered an alternative to recurrence [33]. Rather than compressing the entire history into a single hidden state, self-attention allows the model to look directly at any previous item when making a prediction. SASRec adapted the Transformer architecture to recommendation, achieving strong results by letting each position attend to all earlier positions in the sequence. BERT4Rec went further, training bidirectionally by masking random items and predicting them from context on both sides [62].

The paradigm has been scaled aggressively. Meta's HSTU architecture extends sequential modeling to 1.5 trillion parameters, treating user actions as a generative modeling problem analogous to language [88].

At that scale, the implicit assumption is that sequential patterns in behavior contain deep structure, and that a sufficiently large architecture will eventually learn to exploit it.

2.6.2 When Sequences Mislead

The sequential framing rests on an assumption: that the order in which a user interacts with items carries meaningful information. For language, this is obviously true. "Dog bites man" differs from "man bites dog." Grammar and meaning depend on word order.

For behavioral data, the assumption is less solid. A user who purchases shampoo, a novel, and a phone charger in one session probably added them in whatever order the site surfaced them, not because personal care products lead narratively to literature and then to electronics. Training a model to predict phone chargers given the preceding two items means fitting to noise rather than learning preferences.

The problem runs deeper than random orderings within sessions. User behavior is often fragmented across devices, interrupted by days or weeks, and driven by multiple concurrent needs. A user might browse laptops on Monday, forget about it, then return on Friday after reading reviews elsewhere. The recorded sequence shows laptop views followed by a gap followed by more laptop views, but the actual decision process was continuous, just not visible to the system. Sequential models see gaps and context switches that reflect logging artifacts rather than genuine behavioral transitions.

Even when temporal order is meaningful, it may not be the most informative signal. Knowing that a user visited pages A, B, and C tells you something. But knowing that they visited A, B, and C regardless of order might tell you almost as much. The set of items a user has engaged with, the regions of product space they have explored, can be more stable and predictive than the particular path they took through that space.

2.6.3 A Distributional Alternative

An alternative to sequential modeling is to treat user behavior as samples from an underlying distribution. Rather than asking "what sequence of items did this user interact with," ask "what regions of the item space has this user occupied."

This shift in perspective changes what the model must learn. A sequential model learns transition dynamics: given that a user just viewed item A, what is the probability of viewing item B next? A distributional model learns preference structure: given that a user has shown interest in certain types of items, what other types are they likely to prefer?

The distinction is not between "order matters" and "order is ignored." Temporal information can enter a distributional model through weighting: recent interactions contribute more heavily than older ones, capturing the intuition that current preferences matter more than historical ones. What the distributional view avoids is modeling specific item-to-item transitions. It does not try to learn that viewing running shoes predicts viewing running socks. Instead, it learns that a user who has recently concentrated their attention in the athletic footwear region of product space is likely to remain interested in that region.

This framing has practical advantages. It is robust to the arbitrary orderings that arise from interface design and browsing accidents. It handles variable-length histories naturally, since a distribution can be estimated from any number of samples. And it separates two concerns that sequential models conflate:

what a user is interested in, and how recent that interest is. Recency can be captured through simple decay weighting rather than through complex architectural choices about hidden states and attention patterns.

The question is empirical: for which recommendation tasks does explicit sequential structure help beyond what recency weighting provides? The experiments in later chapters suggest that for many practical settings, the distributional approach with temporal decay matches or exceeds sequential models. Representing these weighted distributions efficiently is a technical challenge. Chapter 5 introduces EMDE, a sketch-based density estimator designed to address this challenge at scale.

2.7 Attention Mechanisms and Their Limitations

The attention mechanism has become ubiquitous in neural sequence modeling since its introduction for machine translation [2]. By allowing models to directly query previous representations rather than compressing all history into a fixed-size hidden state, attention promised to overcome the information bottleneck that limits recurrent architectures. This promise motivated extensive investigation into memory-augmented neural language models.

In prior work, I investigated this question directly [14]. I proposed a key-value-predict attention mechanism that separates the output representation at each time step into three components: a key for memory addressing, a value for memory content, and a predict vector for the next-word distribution. This separation addresses the problem that standard attention forces a single output vector to serve multiple incompatible purposes simultaneously.

The key-value-predict model outperformed existing memory-augmented language models on a Wikipedia corpus (75.8 perplexity versus 80.1 for the Recurrent Memory model of Tran et al. [71]) and the Children's Book Test. However, analysis of learned attention patterns revealed a surprising finding: regardless of the available memory window size, the model predominantly attended to only the five most recent positions.

This observation led to an unexpected result. A simple model that concatenates output representations from the previous three time steps, which I called the 4-gram RNN, achieved 75.9 test perplexity. This matched the sophisticated Key-Value-Predict architecture without any attention mechanism at all. The architectural complexity designed to capture long-range dependencies provided no benefit over direct concatenation of recent context.

These results carry implications beyond language modeling. If attention mechanisms struggle to use extended context even when explicitly designed to do so, the limitation may lie in the sequential framing itself rather than in any particular architectural choice. Sequences impose an ordering that makes distant positions harder to access, regardless of whether the pathway runs through recurrent state updates or attention weights.

This experience informed the direction of this thesis. Rather than designing better attention mechanisms to capture long-range sequential dependencies, the methods developed in subsequent chapters abandon the sequential framing entirely. EMDE represents interaction histories as densities over embedding space, preserving information about *which* items were visited without imposing structure on *when* they were visited. The question of attention span becomes irrelevant when there is no sequence over which to attend. The success of the simple 4-gram RNN, which aggregates recent context through concatenation

rather than learned attention, foreshadows EMDE's approach of aggregating embeddings into fixed-size sketches without sequential structure.

Implications for large-scale sequential recommendation. These findings from 2017 foreshadowed limitations that persist even in the most sophisticated sequential architectures deployed today. Meta's HSTU [88], representing the current industrial frontier with 1.5 trillion parameters, addresses the attention span limitation through several architectural innovations. Pointwise aggregated attention preserves intensity information that softmax normalization destroys. Stochastic length training artificially increases sequence sparsity. Relative attention biases incorporate both positional and temporal information. These modifications improve efficiency and enable scaling, but they do not resolve the fundamental tension identified in my earlier work: attention mechanisms struggle to use extended context even when explicitly designed to do so.

HSTU's own ablation studies reveal this tension. Their stochastic length technique, which randomly truncates sequences during training, removes over 80% of tokens at aggressive settings while maintaining model quality within 0.2% normalized entropy. This finding is notable. If most of a user's history can be discarded without meaningful degradation, the sequential structure that HSTU models so carefully may contain less information than the architecture assumes. The density-based representations developed in this thesis take this insight to its logical conclusion: rather than modeling sequences and then discovering that most positions can be ignored, we represent behavioral histories as distributions from the outset, preserving the information that matters (which items were visited, with what recency) while discarding the information that does not (precise temporal ordering).

2.8 Evaluation Methodology

Measuring the quality of recommendations is subtle. A model that accurately predicts held-out ratings can still produce lists that users find useless; offline metrics computed on historical data do not always predict what happens when a system is deployed. This section establishes the evaluation protocols and metrics used throughout subsequent chapters.

2.8.1 Offline Evaluation

Offline evaluation uses historical data to assess model quality without deploying to users. The available interactions are split into training and test sets. The split strategy significantly affects what is measured.

Random splitting assigns interactions to training and test sets uniformly at random. This is simple but problematic: the model might see a user's recent interactions during training and be asked to predict their earlier ones during testing.

Temporal splitting respects the order of interactions. All interactions before a cutoff time enter training; interactions after enter testing. This simulates the realistic setting where models predict future behavior from past observations. The experiments in this thesis use temporal splits unless otherwise noted.

Leave-one-out is a user-level protocol where for each user, a single interaction (typically the most recent) is held out for testing.

Ranking Metrics

Recommendation output is a ranked list; metrics should reward both finding relevant items and placing them near the top.

Hit Rate (HR@k) measures whether the relevant item appears in the top-k recommendations:

$$\text{HR@}k = \frac{1}{|\mathcal{U}_{\text{test}}|} \sum_{u \in \mathcal{U}_{\text{test}}} \mathbf{1}[i_u \in \mathcal{L}_u^{(k)}] \tag{2.2}$$

where i_u is the test item for user u and $\mathcal{L}_u^{(k)}$ is the top-k recommendation list.

Mean Reciprocal Rank (MRR@k) accounts for position within the list:

$$\text{MRR@}k = \frac{1}{|\mathcal{U}_{\text{test}}|} \sum_{u \in \mathcal{U}_{\text{test}}} \frac{1}{\text{rank}_u} \cdot \mathbf{1}[\text{rank}_u \leq k] \tag{2.3}$$

where rank_u is the position of the relevant item in user u's recommendation list. Items ranked beyond position k contribute zero to the sum.

Normalized Discounted Cumulative Gain (NDCG@k) rewards placing relevant items higher:

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}, \quad \text{DCG@}k = \sum_{j=1}^{k} \frac{2^{r_j} - 1}{\log_2(j+1)} \tag{2.4}$$

where r_j is the relevance of the item at position j and IDCG is the DCG of the ideal ranking.

Mean Average Precision (mAP@k) averages precision across positions where relevant items appear:

$$\text{mAP@}k = \frac{1}{|\mathcal{U}_{\text{test}}|} \sum_{u \in \mathcal{U}_{\text{test}}} \text{AP}_u@k \tag{2.5}$$

where $\text{AP}_u@k$ is the average precision for user u, computed as the mean of precision values at each position containing a relevant item.

Recall@k measures the fraction of relevant items appearing in the top-k:

$$\text{Recall@}k = \frac{|\mathcal{L}_u^{(k)} \cap \mathcal{R}_u|}{|\mathcal{R}_u|} \tag{2.6}$$

where $\mathcal{R}_u$ is the set of relevant items for user u.

Precision@k measures the fraction of recommended items that are relevant:

$$\text{Precision@}k = \frac{|\mathcal{L}_u^{(k)} \cap \mathcal{R}_u|}{k} \tag{2.7}$$

Classification Metrics

Some downstream tasks are binary: churn prediction, click-through prediction, purchase probability estimation. These use classification metrics.

Area Under the ROC Curve (AUC) measures the probability that a randomly chosen positive example is ranked higher than a randomly chosen negative:

$$\text{AUC} = \frac{\sum_{i \in \mathcal{P}} \sum_{j \in \mathcal{N}} \mathbf{1}[\hat{y}_i > \hat{y}_j]}{|\mathcal{P}| \cdot |\mathcal{N}|} \tag{2.8}$$

where $\mathcal{P}$ and $\mathcal{N}$ are sets of positive and negative examples, and $\hat{y}$ denotes predicted scores. AUC is threshold-independent and tolerant of class imbalance, making it suitable for recommendation scenarios where negative examples vastly outnumber positives.

2.8.2 Online Evaluation

Offline metrics measure prediction accuracy on historical data, but users experience recommendations, not accuracy scores. Online evaluation assesses recommendations through experiments with real users.

A/B Testing

In an A/B test, users are randomly assigned to control or treatment groups. The control group receives recommendations from the existing system; the treatment group receives recommendations from a candidate system. Differences in user behavior are attributed to the recommender change.

Key design considerations include the randomization unit (typically account-level for consistency), sample size (determined by power calculations), duration (multiple weeks to capture time-varying effects), and outcome metrics (click-through rates, conversion rates, revenue). The production deployment results in Chapter 6 use A/B testing to validate that offline improvements translate to real engagement gains.

2.8.3 Challenges in Evaluation

Several challenges complicate recommendation evaluation.

Offline-online gaps arise because offline evaluation observes only what happened; it cannot observe what would have happened under different recommendations [55]. A model optimized for offline metrics does not always perform best online.

Selection bias occurs because users interact only with items they encounter. A product never shown to a user cannot be rated by that user, regardless of whether they would have loved it. Training on observed interactions inherits this bias.

Feedback loops emerge when recommendations influence the data used to train future recommendations. If a system recommends item A, item A accumulates more interactions, and future systems learn that item A is popular. This can amplify initial biases [9].

Finally, users value more than accuracy. Diversity, novelty, and fairness all matter for user experience but are not captured by standard accuracy metrics.

These challenges do not invalidate offline evaluation, which remains essential for efficient method comparison, but they counsel appropriate humility about what offline metrics reveal.

2.9 Summary and Road Ahead

This chapter covered the technical foundations needed to understand the contributions of this thesis. The discussion began with the core paradigms of recommender systems (collaborative filtering, content-based methods, and their hybrids) and examined the fundamental challenges they face: cold-start problems, data sparsity, and scalability constraints. The embedding paradigm that dominates modern recommendation research was introduced, along with its key limitation: the aggregation problem that arises when compressing variable-length interaction histories into fixed-size representations.

The discussion of multimodal data pointed to a recurring issue: real-world behavioral systems involve heterogeneous signals that current approaches struggle to integrate cleanly. The contrast between sequential and distributional views of behavior raised a central question that this thesis addresses: when does explicit sequential modeling help, and when does a distributional representation suffice or even excel? The examination of attention mechanisms, including findings from my prior work showing that learned attention collapses to local context, motivated the non-sequential approach developed in later chapters.

Finally, the evaluation section established the metrics and experimental protocols used throughout the dissertation, covering both offline benchmarks and online A/B testing that validates whether offline improvements translate to real user impact.

These foundations set the stage for the next chapter, which surveys specific methods that constitute the current state of the art. Where this chapter asked what the fundamental challenges are, Chapter 3 asks how researchers have attempted to address them and where current solutions fall short.

3 Related Work

This chapter examines methods that constitute the current state of the art in behavioral modeling and recommendation, organized around four paradigms: sequential models that treat behavior as ordered token streams, graph neural networks that propagate information through interaction structure, multimodal systems that fuse heterogeneous signals, and foundation model approaches that attempt to bring transfer learning to recommendation. Each has achieved strong results on specific benchmarks. Each also rests on assumptions that constrain its applicability when behavioral data is sparse, weakly ordered, or drawn from heterogeneous item catalogs.

The goal is not to catalog all existing work but to examine where current approaches succeed, where they fail, and why these failures point toward the density-based alternative developed in this thesis.

3.1 Sequential Recommendation Models

Sequential recommendation has become the dominant paradigm in both academic research and industrial practice. The core idea treats user behavior as a sequence and predicts what comes next, borrowing directly from language modeling. This borrowing has been remarkably productive, enabling recommendation systems to leverage architectural innovations developed for natural language processing (NLP). It has also imported assumptions that frequently fail to hold for behavioral data.

3.1.1 The Sequential Paradigm

The earliest neural approaches to sequential recommendation used recurrent architectures. GRU4Rec introduced gated recurrent units to session-based recommendation, processing items sequentially and using the final hidden state to predict the next item [30]. The model demonstrated substantial improvements over non-neural baselines and established the viability of deep learning for sequential recommendation.

The Transformer architecture subsequently replaced recurrence with self-attention, allowing direct connections between any two positions in a sequence [74]. SASRec adapted self-attention for recommendation [33], embedding each item in the user's history and applying multiple layers of causal attention. BERT4Rec took a different approach, adapting bidirectional training by masking random items within the sequence and training the model to reconstruct them [62]. Careful empirical comparison later revealed that much of BERT4Rec's reported advantage stemmed from differences in loss functions rather than the bidirectional architecture itself [35].

More recent work has explored alternatives to standard Transformers. State space models like Mamba offer linear complexity while maintaining the ability to capture long-range dependencies [24]. Mamba4Rec

applied these ideas to recommendation with competitive performance at lower computational cost [39]. Whether these efficiency improvements enable qualitatively different applications or merely reduce costs remains to be seen.

3.1.2 Industrial Scale: Meta's HSTU

The largest sequential models have been deployed by major technology companies operating at scales far beyond typical academic settings. Meta's Hierarchical Sequential Transduction Units (HSTU) represent the current industrial frontier [88]. The system reformulates recommendation as a sequential transduction task within a generative modeling framework, treating user actions as tokens in a new modality.

HSTU's architectural innovations address specific challenges of recommendation at scale. Pointwise aggregated attention replaces softmax normalization with pointwise normalization followed by layer norm, preserving information about preference intensity that softmax destroys. Stochastic length training randomly truncates sequences during training, reducing computational cost while maintaining quality. The hierarchical design reduces activation memory usage, enabling deeper networks within the same memory budget.

HSTU achieved 12.4% improvement in online A/B tests at Meta, with deployed models reaching 1.5 trillion parameters. Most significantly, HSTU demonstrated that scaling laws from language modeling transfer to recommendations: model quality improves as a power-law of training compute across three orders of magnitude.

3.1.3 The Limits of Sequential Assumptions

Despite these achievements, sequential models rest on an assumption that frequently fails: that temporal order carries essential information. For language, this is obviously true—"dog bites man" differs from "man bites dog." For behavioral data, the assumption is far less solid.

Consider a user who purchases shampoo, a novel, and a phone charger in a single shopping session. The order in which these items were added to the cart likely reflects website layout, search ranking, or simple chance. It does not reflect a coherent narrative where shampoo naturally leads to novels which naturally lead to phone chargers. Treating this sequence as meaningful means fitting to noise rather than learning preferences.

The problem extends beyond random orderings within sessions. User behavior fragments across devices, interrupts over days or weeks, and reflects multiple concurrent needs. A user browsing laptops on Monday may return on Friday after reading reviews elsewhere. The recorded sequence shows laptop views followed by a gap followed by more laptop views, but the actual decision process was continuous, just not visible to the system.

HSTU's own ablation studies inadvertently reveal this limitation. Their stochastic length technique removes over 80% of sequence tokens while maintaining model quality within 0.2% normalized entropy. If four-fifths of a user's interaction history can be discarded without meaningful degradation, the sequential structure that HSTU models so carefully may contain far less information than the architecture assumes.

Reproducibility presents another concern. The same model can produce substantially different results depending on implementation details, hyperparameter choices, and evaluation protocols [50, 63]. Claims

of state-of-the-art performance often fail to replicate when other researchers attempt to reproduce them, making it difficult to assess genuine progress.

These observations connect to findings from my earlier work on attention in language models [14], discussed in Section 2.7. Even when attention mechanisms are explicitly designed to access extended context, learned models concentrate on recent positions. The sequential recommendation methods reviewed here inherit this limitation: architectural sophistication does not guarantee that long-range dependencies will actually be learned.

3.2 Graph Neural Networks for Recommendation

Graph neural networks offer a different perspective. Rather than viewing user behavior as a sequence, graph-based methods represent the problem as a network of entities and relationships. Users, items, and their interactions form a graph whose structure encodes collaborative signals. Graph neural networks learn representations by propagating information through this structure, allowing each node to aggregate information from its neighbors.

3.2.1 From Matrix Factorization to Message Passing

The user-item interaction matrix can be viewed as the adjacency matrix of a bipartite graph. Matrix factorization learns embeddings such that connected pairs have similar representations. Graph neural networks extend this by explicitly modeling multi-hop structure.

Neural Graph Collaborative Filtering (NGCF) made this connection explicit [77]. NGCF represents users and items as nodes and applies graph convolution operations that propagate embeddings along edges. Each layer allows information to flow one hop further. A user's representation is influenced not only by items they interacted with but also by other users who interacted with those items, and by items those users liked.

LightGCN challenged the complexity of early graph neural networks for recommendation [29]. Through careful ablation, the authors showed that feature transformation layers and nonlinear activations actually hurt performance. The essential operation is neighborhood aggregation: averaging the embeddings of neighboring nodes. Removing transformation weights and activations simplified the architecture while improving accuracy by 16% on average. This finding suggests that the collaborative signal encoded in graph structure is the primary source of information, and that complex neural transformations add noise rather than useful inductive bias.

3.2.2 Session Graphs and Knowledge Graphs

Graph structures can encode more than user-item interactions. Session-based methods construct graphs within individual sessions, with edges connecting consecutively clicked items [82]. SR-GNN models sessions as graphs and applies gated graph neural networks to learn item representations. TAGNN adds target-aware attention to weight different items based on what is being predicted [85].

Knowledge graph methods incorporate external information about items, connecting products to their brands, categories, and attributes [76]. The promise is that richer structure should yield better recommendations.

The practical value of these extensions is debated. Recent evaluation of knowledge graph methods found that randomly corrupting or removing knowledge graph facts did not always hurt performance, and sometimes improved it [83]. This suggests either that current methods fail to leverage the available information effectively, or that the information itself is less valuable than assumed.

3.2.3 The Aggregation Bottleneck

Most graph neural network (GNN) based recommendation systems follow a two-tower design [29, 77]. One tower computes user embeddings, the other computes item embeddings, and the prediction is their inner product. This enables efficient inference through approximate nearest neighbor search.

But the two-tower architecture has a fundamental limitation. The user representation is computed without knowledge of which item is being considered, and vice versa. Neither captures information specific to the user-item pair. A user who regularly restocks certain products has a different relationship with those items than with products they have never seen. The two-tower model represents both situations identically.

More fundamentally, compressing a user's neighborhood into a fixed-size embedding through message passing can lose exactly the distributional information needed for personalization. A user connected to both photography equipment and romance novels gets an embedding somewhere between these categories, echoing the averaging problem discussed in Chapter 2. Graph neural networks promised to escape this problem by capturing richer structure, but the final representation still collapses to a point.

3.2.4 Recent Advances

Recent work has attempted to address these limitations. ContextGNN combines pair-wise and two-tower representations [87]. Items within a user's local interaction neighborhood receive pair-wise representations that capture fine-grained context, while items outside this neighborhood use two-tower representations. A learned fusion score determines how to weight the two components.

RelGNN addresses challenges specific to relational databases [10]. Many-to-many relationships create characteristic bridge and hub structures that standard message passing handles poorly. RelGNN introduces atomic routes derived from primary-foreign key relationships, enabling direct information exchange between source and destination nodes without the inefficiencies of standard two-hop propagation.

Both methods achieve strong results on their respective benchmarks. These results demonstrate that careful attention to graph structure matters. They also demonstrate the complexity required: each method introduces specialized mechanisms for particular structural patterns.

3.2.5 Scalability Constraints

Graph neural networks face fundamental scalability challenges. The neighborhood explosion problem is severe: aggregating multi-hop information requires accessing exponentially many nodes as depth

increases. A three-layer GNN on a graph with average degree 100 might need to access millions of nodes to compute a single embedding.

PinSage addressed this for Pinterest's graph of over 3 billion nodes through random walk sampling, importance pooling, and curriculum training [84]. These engineering advances enabled deployment but added substantial complexity. Sampling-based approaches trade exactness for efficiency, introducing variance that different samples produce different representations for the same node.

The computational burden of graph neural networks matters for this thesis because it motivates the alternative approach developed in subsequent chapters. If graph structure can be captured during preprocessing through efficient embedding algorithms like Cleora [57] (Chapter 4), and if these embeddings can be aggregated through density sketches that preserve distributional information, then the expensive message passing at training and inference time becomes unnecessary.

3.3 Multimodal Recommendation

Products are not mere identifiers in a transaction log. They have images, text descriptions, categorical hierarchies, prices, and relationships to other products. Each signal captures something about what makes an item appealing, and each lives in a different representation space. Multimodal recommendation methods attempt to exploit this richness, but the dominant architectural patterns introduce rigidities that limit their practical utility.

3.3.1 Visual and Textual Fusion

VBPR [27] was among the first to demonstrate that visual features improve recommendation beyond what collaborative signals alone provide. It extends Bayesian personalized ranking by incorporating CNN-extracted image features into the latent factor model, so that the predicted affinity between a user and an item reflects both learned collaborative preferences and visual appearance. On fashion datasets where aesthetic appeal drives purchase decisions, the improvement over purely collaborative methods is substantial.

Subsequent work extended this pattern to richer modalities. MMGCN [78] applies graph convolution on user-item interaction graphs for each modality separately, then combines the resulting representations. The method demonstrated that propagating visual and acoustic signals through graph structure captures cross-modal interaction effects that simple feature concatenation misses, particularly for micro-video recommendation where content and behavior are tightly coupled. Knowledge graphs provided another avenue: connecting items to their brands, attributes, and semantic categories creates paths through which preferences can generalize beyond directly observed interactions [76].

The common pattern across these approaches is that each modality receives its own processing pathway, typically a dedicated encoder, followed by a learned fusion stage that combines the modality-specific representations. Early fusion concatenates raw features before the model; late fusion combines predictions from modality-specific models; mid-level fusion merges intermediate representations. Surveys of the field document dozens of variants on this template [90].

3.3.2 The Fusion Bottleneck

Architecturally mediated fusion creates several practical problems. Adding a new data modality requires redesigning the fusion component. Different downstream tasks often require different fusion strategies. When one modality is missing for a subset of items, the architecture must accommodate sparse inputs through specialized mechanisms. And the learned fusion weights reflect the distribution of the training data; when that distribution shifts, the fusion may degrade silently.

More fundamentally, these approaches treat modality combination as an architectural problem rather than a representational one. The question they ask is: given separate representations of behavior, text, and image, how should they be combined? The question this thesis asks is different: given a common representational framework that operates on embedding spaces, why should different modalities require different combination logic at all?

Density sketches sidestep the fusion bottleneck entirely. Each modality produces its own embedding space, and the density of a user's interaction history is estimated independently in each. The resulting sketches are then concatenated. The prediction model learns which regions of which modality-specific sketch contain useful signal through the standard training objective, without any modality-specific fusion architecture. This means adding a new modality is as simple as appending its sketch to the input vector, with no architectural changes required. Chapter 5 demonstrates that this approach consistently improves performance as modalities are added, with the largest gains when modalities capture genuinely complementary structure.

3.4 Foundation Models for Recommendation

The success of foundation models in natural language processing has prompted significant effort to develop analogous systems for recommendation. The goal is appealing: a single pretrained model that handles diverse tasks through fine-tuning or prompting, trained once and deployed broadly. This cross-platform vision remains unachieved. The discussion below concerns this ambitious sense of the term; Chapter 1 (Section 1.3) defines the more precise, narrower sense in which this thesis uses it for BaseModel.

3.4.1 The Vocabulary Problem

Foundation models in NLP work because language has universal vocabulary. The word "cat" means approximately the same thing regardless of which corpus it appears in. Models can be pretrained on text from anywhere and applied to text from anywhere else.

Item identifiers have no such universality. Product ID 12345 on Amazon refers to a completely different entity than product ID 12345 on eBay. User identifiers are even more platform-specific. The collaborative signals that make recommendation work, specifically the patterns of which users interact with which items, do not transfer across platforms: a model trained on Amazon purchase histories has learned nothing applicable to Spotify listening behavior, because the item spaces share no common vocabulary.

This vocabulary problem is not a minor technical obstacle. It strikes at the core of what makes foundation models powerful.

3.4.2 Language Models as Recommenders

One approach sidesteps the vocabulary problem by converting recommendation into a language task. P5 (Pretrain, Personalized Prompt & Predict Paradigm) [21] unified five recommendation task families (rating prediction, sequential recommendation, explanation generation, review summarization, and direct recommendation) as text-to-text problems using a T5 [51] backbone. User histories and item metadata are rendered as natural language through personalized prompt templates; the model generates textual outputs that are parsed into recommendations.

But converting to text loses information. Behavioral patterns obvious in interaction data become obscured when serialized as sentences. The models struggle to capture collaborative signals that emerge from interaction structure rather than content. Language models may complement dedicated recommendation methods; they do not replace them.

3.4.3 Semantic IDs and Generative Retrieval

A more technical approach replaces arbitrary item identifiers with learned semantic IDs that encode item content [52]. If items can be represented as token sequences from a learned codebook, recommendation becomes sequence generation: given a user's history as semantic IDs, generate the semantic IDs of items they will like.

TIGER (Transformer Index for GEnerative Recommenders) [52] implements this idea. Items are encoded using a Residual-Quantized Variational Autoencoder that maps content features to tuples of discrete codes. A Transformer learns to generate these sequences autoregressively, conditioned on the user's history. TIGER achieves strong results, improving NDCG by 10–29% over SASRec and BERT4Rec on Amazon datasets.

The approach has limitations. A small fraction of generated IDs do not map to valid items, requiring fallback mechanisms. Beam search decoding is slower than the approximate nearest neighbor retrieval used by embedding methods. Cold-start handling improves because new items receive semantic IDs from content features alone, but the semantic ID assignment depends on content features that may not capture what actually makes items appealing.

3.4.4 Why No Universal Foundation Model Exists

Despite significant effort, no equivalent of GPT has emerged for recommendation. Several obstacles explain this.

Behavioral data lacks universal vocabulary. Words have consistent meanings across corpora; item and user identifiers are platform-specific. Semantic IDs partially address this for items with content features, but user representations remain tied to specific platforms.

Collaborative signals do not transfer. The insight of collaborative filtering is that users with similar behavior have similar preferences. But similarity is defined relative to a specific catalog and population. Users similar on Amazon may not be similar on Spotify because the behaviors being compared are entirely different.

Scale mismatch limits pretraining. Language models train on trillions of tokens aggregated from the entire internet. Behavioral data is fragmented across platforms with proprietary data that is not shared. Even the largest behavioral datasets are orders of magnitude smaller than text corpora.

The task structure differs as well. Language modeling has a natural self-supervised objective: predict the next token. Recommendation has multiple possible objectives (rating prediction, ranking, next-item prediction) with no consensus on which produces the most transferable representations.

Finally, feature heterogeneity complicates unified architectures. Recommendation involves categorical attributes, numerical features, text, images, timestamps, and graph structure. Language models operate on uniform token representations. Handling heterogeneous inputs makes it harder to design architectures that generalize.

Platform-specific foundation models show that scale helps within a domain. Meta's HSTU demonstrates this. But a single pretrained model that works across recommendation domains, like GPT works across language tasks, remains distant.

3.4.5 Self-Supervised and Contrastive Approaches

A parallel literature sidesteps the labeled-data problem not by pretraining on text but by designing self-supervised objectives directly over interaction graphs. Contrastive methods generate multiple views of the same user's interaction data through augmentation, then train the model to produce similar representations for different views of the same entity and dissimilar representations for different entities [20]. SGL (Self-supervised Graph Learning) augments the interaction graph by randomly dropping nodes or edges and maximizes agreement between representations from different augmented views alongside the standard recommendation objective [80]. SimGCL removes the graph augmentation step entirely, applying random noise directly to embeddings and showing that the contrastive objective itself, rather than the particular augmentation, provides the benefit [86]. Both report substantial improvements over LightGCN on standard benchmarks.

These methods demonstrate that self-supervised signals in behavioral data are real and exploitable. They do not, however, resolve the point representation bottleneck: the final user embedding is still a single vector, produced by aggregating over a graph, and the contrastive objective encourages compact representations rather than preserving distributional structure. The augmentation-based view generation also makes an implicit assumption: that the modified interaction graph is a plausible alternate view of the same underlying preferences. When interaction order is noisy and histories are sparse, aggressive augmentation may destroy the signal along with the noise.

BaseModel's pretraining objective differs in kind. Rather than contrasting augmented views of the same user, it learns to predict the future behavioral distribution from the past distribution. The supervision comes from the temporal structure of real data, not from artificial augmentation. The result is a model trained to understand preference dynamics rather than simply to produce stable embeddings under perturbation.

3.5 Positioning This Thesis

The preceding sections reveal a field that has made remarkable progress on specific benchmarks while leaving fundamental challenges unresolved. Sequential models achieve strong results when sufficient history is available but impose ordering assumptions that often do not hold. Graph neural networks capture collaborative structure but compress it through aggregation bottlenecks that lose distributional information. Foundation models promise generalization but remain constrained by platform-specific vocabularies and fragmented data.

3.5.1 A Common Thread: The Point Representation Bottleneck

A limitation recurs across paradigms: representing users as point vectors loses information that matters for prediction. This is the aggregation problem described in Chapter 2, restated here in terms of its representational outcome: the point representation bottleneck.

Sequential models compress interaction histories into recurrent hidden states or attention-weighted combinations. The final representation is a single vector. A user interested in both electronics and books produces one vector that must somehow encode both interests, typically ending up somewhere between them.

Graph neural networks aggregate neighborhood information through message passing. However many hops they propagate, the final user embedding is still a point. LightGCN showed that the essential operation is averaging, which is precisely the operation that creates phantom preferences by collapsing multimodal distributions to unimodal centroids.

Even foundation models typically produce point embeddings. TIGER's semantic IDs represent items as discrete codes, but the user representation feeding the decoder is still a single vector.

This observation motivates the central hypothesis of this thesis: representing behavioral histories as probability distributions over embedding manifolds, rather than as point vectors, preserves structure that current approaches destroy.

3.5.2 An Alternative: Behavioral Density Estimation

The approach developed in this thesis treats user behavior as samples from an underlying preference distribution. Rather than asking "what sequence of items did this user interact with," it asks "what regions of the item space has this user occupied."

A user who purchased running shoes, fitness trackers, and protein supplements has not traced a trajectory through product space; they have indicated which regions match their preferences. Representing this history as a distribution over item features preserves the multimodal structure that point representations collapse. The user interested in both electronics and books should have a bimodal profile, not a single vector awkwardly positioned between unrelated categories.

This perspective addresses the limitations identified above. The distributional view treats histories as weighted sets rather than sequences. Temporal information enters through recency weighting, not through architectural choices about hidden states and attention patterns. This avoids fitting to spurious orderings while still capturing that recent behavior matters more than distant past.

Density sketches preserve distributional structure that averaging destroys. Two users with identical mean embeddings but different preference distributions produce different sketches. The representation maintains identity-level detail about which specific items were engaged with.

Because the representation operates on feature space rather than item identities, new items with content features can be positioned in the distribution immediately. Different modalities such as collaborative signals, text, and images each define their own feature spaces; density sketches can be computed independently and concatenated without learned fusion mechanisms.

The representation has fixed size regardless of catalog size or history length. There are no per-item embedding tables that grow with the catalog. Profile construction involves summing precomputed bucket codes, completing in microseconds.

3.5.3 Summary

The methods developed in subsequent chapters constitute a concrete realization of this perspective. Cleora captures collaborative structure from interaction graphs. EMDE converts those embeddings into distributional sketches that preserve multimodal preference structure and accommodate new modalities through concatenation rather than architectural redesign. BaseModel learns how these distributions evolve through self-supervised temporal prediction, supporting recommendation, classification, and regression from a single pretrained backbone. Validation across three international competitions, with consistent top-three placements against teams from leading research organizations, demonstrates that the approach generalizes beyond individual benchmarks.

This framing is complementary to existing work rather than opposed to it. Sequential models remain appropriate when interaction order carries genuine meaning. Graph neural networks expose collaborative structure that density sketches exploit. The claim is narrower: density-based representations provide measurable advantages precisely when ordering is unreliable, cold-start rates are high, and the multimodal structure of preferences matters. That describes a large share of practical recommendation in e-commerce.

4 Graph Embeddings for Behavioral Data

Behavioral data possesses inherent relational structure that extends beyond the properties of individual entities. When a customer purchases a product, this event creates connections not only between customer and product, but also between the product and its brand, its category, and the context of the transaction. These connections form networks whose topology encodes information that no single entity's features can capture. A product's position in a co-purchase network reveals which user segments find it appealing, information that product descriptions and images cannot provide on their own.

This chapter develops the mathematical framework for extracting such structural information. The methods presented here provide the foundational entity representations on which EMDE and BaseModel, the original contributions of this thesis, are built. Section 4.1 introduces graphs and hypergraphs as formal models of relational data, emphasizing why the hypergraph formulation naturally captures the multi-way relationships present in behavioral events. Section 4.2 defines the node embedding problem and surveys existing approaches, identifying the requirements that practical behavioral systems impose. Section 4.3 presents Cleora, a scalable embedding algorithm developed by colleagues at Synerise [57], which serves as the primary method for computing entity embeddings throughout the experimental work in this thesis. While I did not develop Cleora, understanding its mechanics is essential for the chapters that follow. Section 4.4 discusses the gap between graph embeddings and the aggregation mechanisms needed for behavioral modeling, motivating the density-based approach developed in Chapter 5.

4.1 Graphs and Hypergraphs

Behavioral data is everywhere. Every click, purchase, search query, and page visit generates a trace of human intent and preference. Yet beneath the apparent chaos of user interactions lies structure. When a customer purchases a product, this single event creates a connection between the customer and the product, but also between the product and its brand, its category, and the context of the purchase itself. These connections form networks whose topology encodes information that no single entity's features can capture. A product's position in a co-purchase network reveals which user segments find it appealing, which is information that product descriptions and images cannot provide on their own.

The mathematical language of graphs provides a natural formalism for representing such relational data. This section introduces the necessary definitions and explains why hypergraphs, rather than ordinary graphs, are the appropriate abstraction for behavioral data.

4.1.1 Basic Definitions

A *graph* $G = (V, E)$ consists of a finite set V of nodes (or vertices) together with a set E of edges, where each edge connects a pair of nodes. In an undirected graph, edges are unordered pairs $\{u, v\}$ with $u, v \in V$; in a directed graph, edges are ordered pairs (u, v) representing a connection from u to v. A *weighted graph* associates a weight $w : E \to \mathbb{R}^+$ with each edge, capturing the strength or frequency of the connection.

The structure of a graph can be represented algebraically through its *adjacency matrix* $\mathbf{A} \in \mathbb{R}^{|V| \times |V|}$, defined by

$$A_{ij} = \begin{cases} w(\{i, j\}) & \text{if } \{i, j\} \in E \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

for undirected graphs, where $w(\{i, j\}) = 1$ in the unweighted case. For directed graphs, A_{ij} records the weight of the edge from node i to node j. The adjacency matrix completely specifies the graph's connectivity, but its size grows quadratically with the number of nodes. For graphs with millions of nodes, explicit storage of $\mathbf{A}$ becomes impractical. Real-world behavioral graphs are typically sparse, however: each user interacts with a tiny fraction of available items, so most entries of $\mathbf{A}$ are zero. Sparse matrix representations exploit this structure by storing only the non-zero entries.

The *degree* of a node v, denoted $\deg(v)$, counts the number of edges incident to v. In weighted graphs, the *weighted degree* (or strength) sums the weights of incident edges:

$$\deg_w(v) = \sum_{u : \{u,v\} \in E} w(\{u, v\}). \tag{4.2}$$

The degree distribution of a graph, describing the frequency of nodes with each degree, often reveals important structural properties. Behavioral graphs typically exhibit heavy-tailed degree distributions: a small number of popular items accumulate many interactions while most items have few.

4.1.2 From Pairwise to Multi-way Relations

The standard graph formulation restricts edges to connecting exactly two nodes. This limitation proves problematic for behavioral data, where events naturally involve multiple entities simultaneously.

Consider a purchase transaction in which customer c buys products p_1, p_2, and p_3 together in a single basket at store s. Representing this event as a collection of pairwise edges loses essential information. Creating edges (c, p_1), (c, p_2), (c, p_3), and (c, s) captures that the customer interacted with each entity, but loses the fact that the three products were purchased together. The co-occurrence of p_1, p_2, and p_3 in a single basket carries meaning that their separate connections to c cannot convey: these products may be complements, or they may appeal to a particular shopping occasion. A pairwise representation cannot distinguish a basket containing all three products from three separate transactions each containing one product.

Hypergraphs generalize graphs by allowing edges to connect arbitrary subsets of nodes. Formally, a hypergraph $H = (V, \mathcal{E})$ consists of a node set V and a set $\mathcal{E}$ of *hyperedges*, where each hyperedge $e \in \mathcal{E}$ is a subset of V with no restriction on cardinality. The shopping transaction described above becomes a single hyperedge $e = \{c, p_1, p_2, p_3, s\}$, preserving the co-occurrence structure.

The *incidence matrix* $\mathbf{H} \in \{0,1\}^{|V| \times |\mathcal{E}|}$ provides a matrix representation of hypergraph structure:

$$H_{ve} = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{otherwise.} \end{cases} \tag{4.3}$$

Unlike the adjacency matrix, which is square and grows quadratically with $|V|$, the incidence matrix has dimensions determined by both the number of nodes and the number of hyperedges. For behavioral data where each transaction creates one hyperedge, the incidence matrix grows linearly with the number of transactions.

From the incidence matrix, node degrees and hyperedge cardinalities follow directly. The degree of node v equals the number of hyperedges containing it:

$$\deg(v) = \sum_{e \in \mathcal{E}} H_{ve} = \|\mathbf{H}_{v,:}\|_1. \tag{4.4}$$

The cardinality (or width) of hyperedge e equals the number of nodes it contains:

$$|e| = \sum_{v \in V} H_{ve} = \|\mathbf{H}_{:,e}\|_1. \tag{4.5}$$

4.1.3 Hypergraphs in Behavioral Modeling

The hypergraph formulation aligns naturally with how behavioral events are recorded. A single row in a transaction log typically contains multiple fields: user identifier, item identifier, timestamp, device type, location, and various contextual attributes. Each field value can be treated as a node, and each transaction becomes a hyperedge linking all participating entities.

This perspective reveals structure that pairwise analysis obscures. Two products that frequently appear in the same baskets share many hyperedges, even if no transaction contains only those two products. A brand node becomes connected to user nodes through the products that mediate the relationship. Category hierarchies emerge from the products they contain. These higher-order patterns reflect the compositional nature of behavioral events and provide signal for downstream prediction tasks.

Throughout this thesis, hypergraphs are constructed from behavioral logs by treating each interaction event as a hyperedge. The specific entity types depend on the application domain. In the e-commerce experiments of Chapters 5 and 6, hyperedges connect users, products, brands, and categories. In the Booking.com competition (Chapter 7), hyperedges connect travelers to the cities they visit. In the KDD Cup citation network, hyperedges connect papers to their authors and institutions. The principle remains constant: the hypergraph captures the full relational structure of each event, preserving co-occurrence patterns that pairwise graphs would lose.

4.1.4 Graph Notation Summary

Table 4.1 summarizes notation for graphs and hypergraphs used in this chapter and throughout the thesis.

Table 4.1: Graph and hypergraph notation.

Symbol	Description
Graph Structure	
$G = (V, E)$	Graph with node set V and edge set E
$\|V\|, \|E\|$	Number of nodes and edges
$\mathbf{A} \in \mathbb{R}^{\|V\| \times \|V\|}$	Adjacency matrix
w_{ij}	Weight of edge (i, j)
$\deg(v)$	Degree of node v
$\mathcal{N}(v)$	Neighborhood of node v: set of adjacent nodes
Hypergraph Structure	
$H = (V, \mathcal{E})$	Hypergraph with node set V and hyperedge set $\mathcal{E}$
$e \in \mathcal{E}$	A hyperedge (subset of V)
$\|e\|$	Cardinality of hyperedge e
$\mathbf{H} \in \{0, 1\}^{\|V\| \times \|\mathcal{E}\|}$	Incidence matrix
Cleora Embeddings	
$\mathbf{P} \in \mathbb{R}^{\|V\| \times \|V\|}$	Random walk transition matrix
$\mathbf{T}^{(\ell)} \in \mathbb{R}^{\|V\| \times d}$	Node embedding matrix at iteration ℓ
$\mathbf{t}_v \in \mathbb{R}^d$	Embedding vector for node v
I	Number of Cleora iterations

4.2 The Node Embedding Problem

Graphs and hypergraphs encode relational structure, but machine learning algorithms require numerical input. Neural networks operate on continuous vectors. Gradient-based optimization demands differentiable representations. Similarity computations require a metric space where distances are meaningful. The challenge is to transform discrete graph structure into continuous representations that preserve meaningful relationships.

4.2.1 Problem Formulation

The *node embedding problem* seeks a mapping $\phi : V \to \mathbb{R}^d$ that assigns to each node v a d-dimensional vector $\mathbf{z}_v = \phi(v)$ such that the geometry of the embedding space reflects structural relationships in the graph. The embedding dimension d is typically much smaller than $|V|$, compressing the high-dimensional adjacency structure into a more compact representation.

What should it mean for embeddings to reflect graph structure? Different applications demand different answers, and this choice fundamentally shapes the embedding method. Common notions of structural similarity include the following.

First-order proximity requires that nodes connected by an edge have similar embeddings. If $(u, v) \in E$, then $\mathbf{z}_u$ and $\mathbf{z}_v$ should be close in the embedding space. This notion is local, considering only direct connections.

Second-order proximity requires that nodes with similar neighborhoods have similar embeddings. Even if u and v are not directly connected, they should have similar representations if they connect to

similar sets of other nodes. This captures the intuition that nodes playing similar structural roles should be represented similarly.

Higher-order proximity extends this principle to longer paths. Nodes reachable through short paths should be more similar than those connected only through long paths. This notion relates to community structure: densely connected regions of the graph should map to compact regions of the embedding space.

Formally, given a similarity function $s : V \times V \to \mathbb{R}$ defined over node pairs based on graph structure, the goal is to find embeddings such that

$$s(u, v) \approx g(\mathbf{z}_u, \mathbf{z}_v) \tag{4.6}$$

for some function g in the embedding space, typically the inner product $\mathbf{z}_u^\top \mathbf{z}_v$ or a decreasing function of Euclidean distance. The choice of s encodes which structural relationships the embedding should preserve.

4.2.2 Existing Approaches

The literature contains numerous node embedding methods, which can be organized into several families based on their underlying principles.

Spectral methods. Classical approaches exploit spectral properties of graph matrices. Laplacian Eigenmaps [5] compute embeddings from the eigenvectors of the graph Laplacian $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal degree matrix. The smallest non-trivial eigenvectors provide coordinates that minimize $\sum_{(i,j)\in E} \|\mathbf{z}_i - \mathbf{z}_j\|^2$, encouraging connected nodes to have similar embeddings. These methods possess elegant theoretical properties but require eigendecomposition, which scales poorly to large graphs. Computing eigenvectors of a matrix with millions of rows is computationally prohibitive for the datasets considered in this thesis.

Random walk methods. DeepWalk [49] introduced a connection between graph structure and natural language processing. The algorithm performs random walks on the graph, treating the resulting node sequences as sentences and applying the skip-gram objective from Word2Vec [43]. The intuition is that nodes frequently co-occurring in random walks should have similar embeddings, just as words frequently co-occurring in text windows should have similar word vectors. Node2Vec [23] extends this approach with parameters controlling the trade-off between breadth-first (local) and depth-first (global) exploration.

Random walk methods have proven effective across many benchmarks, but they require extensive computation. Generating sufficient random walks to cover the graph demands time proportional to the number of walks, their length, and the number of nodes. The subsequent skip-gram training adds further computational burden. On graphs with millions of nodes, these methods can require hours or days to converge.

Matrix factorization methods. Several methods frame node embedding as factorization of a matrix derived from the adjacency structure. LINE [70] optimizes embeddings to preserve both first-order and second-order proximity through separate objective functions. GraRep [8] factorizes powers of the transition matrix to capture different scales of proximity. These methods often reduce to eigendecomposition of particular matrices, connecting them to spectral approaches while sometimes offering computational advantages through approximation techniques.

Graph neural networks. A more recent line of work learns embeddings through neural network architectures that operate directly on graph structure. Graph Convolutional Networks [34] and GraphSAGE [25] compute node representations by aggregating features from local neighborhoods, stacking multiple layers to capture multi-hop information. These methods can incorporate node features beyond graph structure and support inductive learning on unseen nodes. However, they typically require labeled data for training, turning embedding into a supervised learning problem. When labels are scarce or when the goal is general-purpose representations for multiple downstream tasks, this requirement limits applicability.

4.2.3 Requirements for Behavioral Systems

The methods surveyed above were developed primarily for academic benchmarks with moderate-scale graphs. Production behavioral systems impose additional requirements that disqualify many approaches.

Scalability. Real-world behavioral graphs contain millions of nodes and billions of edges. Methods requiring eigendecomposition or quadratic-time operations cannot handle these volumes. Even methods with theoretically favorable complexity may prove impractical if constant factors are large or if the implementation does not parallelize effectively.

Efficiency. Behavioral patterns evolve over time. User preferences shift, new products enter the catalog, and seasonal trends modulate interaction patterns. Embeddings must be recomputed periodically to remain current. A method requiring days of computation for a single embedding update cannot support the rapid iteration cycles that production systems demand.

Determinism. Non-deterministic methods produce different embeddings on different runs, complicating debugging, testing, and reproducibility. When embeddings change unexpectedly, determining whether the cause is a bug, a data change, or random variation consumes engineering effort. Deterministic methods, which produce identical output given identical input, eliminate this source of uncertainty.

Inductive capability. New nodes appear continuously as users join the platform and products enter the catalog. Methods that require recomputing all embeddings to accommodate new nodes impose unacceptable latency. Inductive methods, which can compute embeddings for new nodes from their connections to existing nodes, enable real-time incorporation of new entities.

These requirements guided the selection of Cleora as the embedding method for this thesis. The next section describes Cleora in detail, explaining how it satisfies these requirements while producing embeddings of competitive quality.

4.3 Cleora: Scalable Hypergraph Embeddings

Cleora is a graph embedding algorithm developed by Rychalska et al. at Synerise [57]. I did not participate in its development, but I have used it extensively throughout the research presented in this thesis. Cleora embeddings serve as input to EMDE in Chapter 5, provide the collaborative signal component of BaseModel profiles in Chapter 6, and formed the foundation of all three competition solutions in Chapter 7. Understanding how Cleora operates is therefore essential for interpreting the experimental results that follow.

The defining characteristic of Cleora is its simplicity. There is no objective function to optimize, no sampling of positive and negative examples, and no gradient descent. The algorithm consists of repeated matrix multiplication followed by normalization. This simplicity yields several practical benefits: deterministic output, linear time complexity, straightforward parallelization, and natural support for inductive embedding of new nodes. These properties make Cleora well-suited to production environments where reliability and efficiency matter as much as embedding quality.

4.3.1 Algorithm Description

The core principle underlying Cleora is that a node's embedding should resemble the embeddings of its neighbors. Rather than formulating this as an optimization objective and solving it through gradient descent, Cleora achieves it directly through averaging. Each node's embedding is replaced by the weighted average of its neighbors' embeddings, then normalized to maintain numerical stability. After several iterations, nodes with similar neighborhoods converge to similar representations.

To state this precisely, let $G = (V, E)$ be a graph with adjacency weights w_{ij} for each edge $(i, j) \in E$. The random walk transition matrix $\mathbf{P} \in \mathbb{R}^{|V|\times|V|}$ is defined by

$$P_{ij} = \frac{w_{ij}}{\sum_{k:(i,k)\in E} w_{ik}}, \tag{4.7}$$

with $P_{ij} = 0$ when $(i, j) \notin E$. Each row of $\mathbf{P}$ sums to one, and entry P_{ij} represents the probability of transitioning from node i to node j in a random walk. Nodes with stronger connections to i receive higher transition probabilities.

The algorithm initializes an embedding matrix $\mathbf{T}^{(0)} \in \mathbb{R}^{|V|\times d}$ with entries drawn uniformly from $[-1, 1]$. Random initialization ensures that all nodes begin with distinct embeddings of similar pairwise distances, avoiding both collapse to identical representations and spurious structure from accidentally similar initializations.

The iterative update rule takes the form

$$\mathbf{T}^{(\ell+1)} = \text{normalize}\left(\mathbf{P} \cdot \mathbf{T}^{(\ell)}\right), \tag{4.8}$$

where normalize$(\cdot)$ applies L_2 normalization to each row, projecting all embedding vectors onto the unit hypersphere. After I iterations, the final embedding matrix $\mathbf{T}^{(I)}$ is returned.

Algorithm 1 Cleora embedding algorithm

Require: Graph $G = (V, E)$, iteration count I, embedding dimension d
Ensure: Embedding matrix $\mathbf{T} \in \mathbb{R}^{|V| \times d}$
1: Compute transition matrix $\mathbf{P}$ where $P_{ij} = w_{ij} / \sum_{k:(i,k)\in E} w_{ik}$
2: Initialize $\mathbf{T}^{(0)} \sim \text{Uniform}(-1, 1)^{|V| \times d}$
3: **for** $\ell = 0$ to $I - 1$ **do**
4: $\quad \mathbf{T}^{(\ell+1)} \leftarrow \mathbf{P} \cdot \mathbf{T}^{(\ell)}$
5: $\quad \mathbf{T}^{(\ell+1)}_{i,:} \leftarrow \mathbf{T}^{(\ell+1)}_{i,:} / \|\mathbf{T}^{(\ell+1)}_{i,:}\|_2$ for all $i \in V$
6: **end for**
7: **return** $\mathbf{T}^{(I)}$

The matrix multiplication $\mathbf{P} \cdot \mathbf{T}^{(\ell)}$ replaces each node's embedding with a weighted average of its neighbors' embeddings from the previous iteration. For a single node i, this operation computes

$$\tilde{\mathbf{t}}_i^{(\ell+1)} = \sum_{j:(i,j)\in E} P_{ij} \cdot \mathbf{t}_j^{(\ell)}. \tag{4.9}$$

The subsequent normalization ensures that $\|\mathbf{t}_i^{(\ell+1)}\|_2 = 1$, preventing embeddings from collapsing toward zero through repeated multiplication by $\mathbf{P}$, whose spectral radius is at most one.

Algorithm 1 presents the complete procedure, including support for processing graphs in chunks when memory constraints preclude loading the entire graph at once.

4.3.2 Hypergraph Expansion

Cleora operates on pairwise edges, but behavioral data naturally forms hypergraphs. Before applying the algorithm, hyperedges must be converted into ordinary edges. Cleora supports two expansion strategies.

Clique expansion transforms each hyperedge $e = \{v_1, v_2, \ldots, v_k\}$ into a complete subgraph connecting all $\binom{k}{2}$ pairs of participating nodes. This preserves the intuition that all nodes in a hyperedge are mutually related: if three products were purchased together, clique expansion creates edges between each pair. The cost is quadratic growth in the number of edges with respect to hyperedge cardinality.

Star expansion introduces an auxiliary node n_e for each hyperedge e and creates k edges connecting n_e to each original node in e. This grows linearly with hyperedge cardinality but increases the total node count by $|\mathcal{E}|$. Nodes that participated in the same hyperedge become connected through a path of length two via the auxiliary node, rather than directly.

The choice between strategies depends on hyperedge size distributions. Clique expansion is appropriate when hyperedges are small, as in many e-commerce datasets where basket sizes are moderate. Star expansion becomes necessary when hyperedges vary widely in size or when some hyperedges contain many nodes, since direct clique expansion would create an intractable number of edges.

In the experiments throughout this thesis, clique expansion is used for datasets with bounded hyperedge sizes (shopping baskets, trip itineraries) and star expansion for datasets with large or highly variable hyperedges.

4.3.3 Convergence Behavior

Cleora does not optimize an explicit objective function, so there is no loss value to monitor for convergence. Instead, the embeddings evolve through iterations, capturing progressively more global structure.

After one iteration, nodes with identical immediate neighborhoods obtain identical embeddings. After two iterations, similarity extends to two-hop neighborhoods. Each additional iteration incorporates information from one hop further in the graph. The number of iterations thus controls the scale of structural similarity that embeddings capture.

This behavior connects to the concept of average path length. If the average shortest path between pairs of nodes is ℓ, then approximately ℓ iterations suffice to propagate information across typical node pairs. For most behavioral graphs encountered in this thesis, average path lengths fall between 3 and 6, suggesting that 4 to 5 iterations should capture relevant structure.

Too few iterations produce embeddings that reflect only local neighborhoods, potentially missing global patterns. Too many iterations cause embeddings to converge toward uniformity: as information propagates through the entire connected component, all nodes receive influence from all other nodes, and distinctions wash out. In the limit, if the graph is connected, all embeddings converge to the same vector.

The original Cleora paper [57] demonstrates this trade-off empirically, showing that downstream task performance typically peaks around 4 to 5 iterations and then degrades. In the experiments reported here, the iteration count is treated as a hyperparameter, selected based on validation performance for each downstream task. Table 6.4 in Chapter 6 reports the iteration counts used across all experimental settings, which range from 3 to 6.

4.3.4 Computational Complexity

The efficiency of Cleora stems from the simplicity of its operations. Each iteration requires one sparse matrix multiplication and one normalization pass.

Sparse matrix multiplication $\mathbf{P} \cdot \mathbf{T}^{(\ell)}$ costs $O(|E| \cdot d)$, where $|E|$ is the number of edges after hypergraph expansion. For each non-zero entry of $\mathbf{P}$, the algorithm performs d multiply-add operations. Row normalization costs $O(|V| \cdot d)$. With I iterations, the total time complexity is

$$O\big(I \cdot (|E| + |V|) \cdot d\big), \tag{4.10}$$

which is linear in the size of the graph.

Memory consumption is similarly modest. The transition matrix $\mathbf{P}$ is stored in sparse format, requiring $O(|E|)$ space. The embedding matrix $\mathbf{T}$ requires $O(|V| \cdot d)$ space. No auxiliary data structures grow with graph size beyond these requirements.

Table 4.2 compares Cleora's computational requirements against other embedding methods on graphs of varying sizes. The comparison, drawn from the original Cleora paper, demonstrates order-of-magnitude speedups over methods like DeepWalk while maintaining competitive embedding quality on downstream tasks.

Table 4.2: Embedding computation time comparison across methods and datasets, adapted from Rychalska et al. [57]. Times include data loading and preprocessing. Cleora achieves substantial speedups over competing methods, with the advantage growing on larger graphs.

Method	**Facebook** (22K nodes)	**YouTube** (1.1M nodes)	**RoadNet** (2.0M nodes)	**LiveJournal** (4.8M nodes)
Cleora	0.7 min	12 min	24 min	96 min
PBG	4.5 min	55 min	38 min	638 min
DeepWalk	37 min	1716 min	3209 min	timeout

The Cleora implementation, written in Rust with attention to memory layout and parallelization, fully exploits these theoretical properties. In practice, graphs with tens of millions of nodes and hundreds of millions of edges complete embedding in under two hours on a machine with 32 CPU cores. This efficiency enabled the hyperparameter exploration reported in subsequent chapters.

4.3.5 Properties Relevant to This Thesis

Several properties of Cleora prove particularly valuable for the behavioral modeling tasks addressed in this thesis.

Determinism. Given identical input and random seed, Cleora produces identical embeddings on every execution. This reproducibility simplifies debugging and validation. When downstream performance changes unexpectedly, the cause must lie in the input data or the downstream model, not in embedding randomness.

Inductive embedding. New nodes can be embedded without recomputing the entire graph. If a new node v connects to existing nodes $\{u_1, \ldots, u_k\}$ with weights $\{w_1, \ldots, w_k\}$, its embedding is computed as

$$\mathbf{t}_v = \frac{\sum_{i=1}^{k} w_i \cdot \mathbf{t}_{u_i}}{\left\| \sum_{i=1}^{k} w_i \cdot \mathbf{t}_{u_i} \right\|_2}. \tag{4.11}$$

This is precisely what one iteration of Cleora would compute for the new node given fixed neighbor embeddings. For products entering an e-commerce catalog, this property enables immediate embedding based on connections to existing brands, categories, or co-purchased items.

Feature initialization. The algorithm can incorporate content features by initializing $\mathbf{T}^{(0)}$ with precomputed embeddings rather than random values. If nodes have associated text or images, embeddings from language models or vision transformers can serve as initialization. The iterative averaging then blends content-based similarity with structural similarity from the graph topology. In the BaseModel experiments of Chapter 6, this capability is used to combine collaborative signals from interaction graphs with semantic signals from product descriptions and images.

Interpretable similarity. The cosine similarity between Cleora embeddings has a clear interpretation: it measures neighborhood overlap at the scale determined by the iteration count. Two nodes with high similarity share similar k-hop neighborhoods, where k equals the iteration count. This interpretability aids in understanding model behavior and diagnosing unexpected results.

4.3.6 Limitations

Cleora's simplicity entails trade-offs that should be acknowledged.

The algorithm cannot be tuned for specific downstream tasks. The embeddings capture general neighborhood structure without knowing whether the goal is link prediction, classification, or recommendation. Task-specific adaptation requires additional components built on top of the Cleora embeddings, which is precisely the role that EMDE and BaseModel fulfill in subsequent chapters.

Cleora treats all edges as expressing the same type of similarity. It cannot learn that certain edge types are more informative than others for a particular task. In heterogeneous graphs with multiple relation types (purchased, viewed, returned), all relations contribute equally to the neighborhood average. More sophisticated methods might weight relation types differently, but Cleora's uniform treatment sacrifices this flexibility for simplicity and speed.

When content features are incorporated through initialization, Cleora propagates and smooths these features according to graph structure but does not learn how to optimally combine content and structure. If the initial embeddings and the graph topology encode conflicting notions of similarity, Cleora provides no mechanism to resolve the conflict. The downstream components must handle such tensions.

Finally, while the linear complexity is theoretically attractive, constant factors matter in practice. For graphs exceeding available memory, Cleora requires partitioning strategies that add engineering complexity. The experiments in this thesis did not encounter this limitation, but it would become relevant at larger scales.

4.4 From Embeddings to Behavioral Profiles

Cleora and related embedding methods solve the problem of representing individual entities as dense vectors. A product, a brand, a category, or a user can each be mapped to a point in $\mathbb{R}^d$ where geometric proximity reflects co-occurrence in behavioral events. This representation enables similarity search, clustering, and other operations that require numerical input.

However, a gap remains between entity embeddings and the representations needed for behavioral prediction. A user who has purchased fifty products has fifty product embeddings. A neural network needs a fixed-size input regardless of how many products appear in the history. How should variable-length collections of embeddings be converted into a single representation suitable for downstream models?

The standard solution is averaging: compute the mean of all item embeddings in the user's history. This produces a fixed-size vector regardless of history length, which is computationally convenient. But as discussed in Chapter 2, averaging destroys information that matters for prediction. A user who purchased both a laptop and a novel produces an average embedding that falls somewhere between electronics and

books, potentially in a region corresponding to neither interest. The averaged representation describes a phantom preference that exists nowhere in the user's actual history.

An alternative perspective, developed in the next chapter, treats behavioral histories as probability distributions over the embedding manifold rather than as single points. A user interested in both electronics and books should have a bimodal profile showing concentration in both regions, not a single point awkwardly positioned between them. This distributional view motivates EMDE, which combines ideas from locality-sensitive hashing and streaming algorithms to construct fixed-size sketches that preserve distributional structure.

The connection between Cleora and EMDE is direct: Cleora provides the embedding space over which EMDE estimates densities. The quality of the embeddings affects what structure the density estimates can capture. If Cleora positions similar items nearby and dissimilar items far apart, then EMDE's locality-sensitive partitioning will group items in meaningful ways. If the embeddings lack geometric structure, EMDE degenerates to a counting mechanism that ignores item relationships.

This dependence was verified empirically in the ablation studies of Chapter 5. When Cleora embeddings were replaced with random vectors lacking geometric structure, EMDE performance dropped substantially (MRR from 0.365 to 0.279 on the RETAIL dataset). The density-based representation succeeds precisely because it preserves distributional information over a meaningful embedding manifold.

4.5 Summary

This chapter established the graph-theoretic and algorithmic foundations for the methods developed in subsequent chapters. The key concepts introduced are as follows.

Behavioral data naturally forms hypergraphs where each interaction event connects multiple entities simultaneously. The hypergraph formulation preserves co-occurrence structure that pairwise graphs lose, capturing the compositional nature of transactions, sessions, and other behavioral events.

The node embedding problem asks for a mapping from discrete graph structure to continuous vector representations that preserve meaningful relationships. Existing approaches include spectral methods, random walk methods, matrix factorization, and graph neural networks, each with different trade-offs between quality, scalability, and flexibility.

Cleora addresses the requirements of production behavioral systems through a simple iterative averaging scheme. Its linear time complexity, deterministic output, and inductive capability make it suitable for large-scale applications where embeddings must be computed efficiently and updated frequently. The algorithm produces embeddings where nodes with similar neighborhoods receive similar representations, capturing the collaborative structure implicit in behavioral data.

The embeddings produced by Cleora position entities in a continuous space where geometric proximity reflects co-occurrence in behavioral events. Products frequently purchased by the same users cluster together. Brands associated with similar product categories occupy nearby regions. This geometric structure provides the foundation for density-based behavioral profiles.

However, individual entity embeddings do not directly solve the representation problem for behavioral prediction. Users have variable-length interaction histories that must be converted to fixed-size inputs for neural networks. The next chapter introduces EMDE, which addresses this aggregation problem by

representing histories as probability distributions over the embedding manifold rather than as single points or averaged vectors. The density-based representation preserves information about which regions of item space a user has visited, enabling downstream models to make predictions that respect the multimodal structure of real preferences.

5 Efficient Manifold Density Estimator

This chapter is based on the following publications:

Jacek Dąbrowski, Barbara Rychalska, **Michał Daniluk**, Dominika Basaj, Konrad Gołuchowski, Piotr Bąbel, Andrzej Michałowski, and Adam Jakubowski. "An Efficient Manifold Density Estimator for All Recommendation Systems." *International Conference on Neural Information Processing* (ICONIP 2021), Springer, pp. 323-337, 2021 [12].

Michał Daniluk and Jacek Dąbrowski. "Temporal Graph Models Fail to Capture Global Temporal Dynamics." *Temporal Graph Learning Workshop at NeurIPS 2023*, New Orleans, December 2023.

My contributions: For the ICONIP paper, the overall CMS+LSH combination framework was conceived by Dąbrowski. I proposed density-dependent partitioning, the key algorithmic modification that positions hyperplane thresholds at data quantiles rather than at zero, ensuring balanced bucket utilization and making EMDE a genuine manifold density estimator rather than a standard LSH hash table. I also developed the theoretical grounding connecting this scheme to manifold density estimation, designed and executed the experimental evaluation on session-based and top-k recommendation benchmarks, and developed the training and inference pipeline. For the NeurIPS workshop paper, I conceived the research direction, designed the experiments, developed the PopTrack baseline, and was first author. Code is available online.[1]

This chapter addresses three hypotheses introduced in Chapter 1:

Hypothesis H1: Density-based representations outperform point-based aggregation for behavioral modeling. This is evaluated by comparing EMDE against baselines using mean-pooled embeddings with identical input features, and by examining performance degradation when structured embeddings are replaced with random vectors.

Hypothesis H2: Density sketching enables effective multimodal integration through simple concatenation. This is evaluated by comparing unimodal against multimodal EMDE configurations across multiple datasets.

Hypothesis H3: Density-based input representations reduce the benefits of architectural complexity. This is evaluated by showing that shallow feed-forward networks operating on EMDE sketches match or exceed deep sequential models and graph neural networks while requiring substantially less training time.

[1] EMDE: `https://github.com/emde-conf/emde-session-rec`; Temporal graph analysis: `https://github.com/temporal-graphs-negative-sampling/TGB`

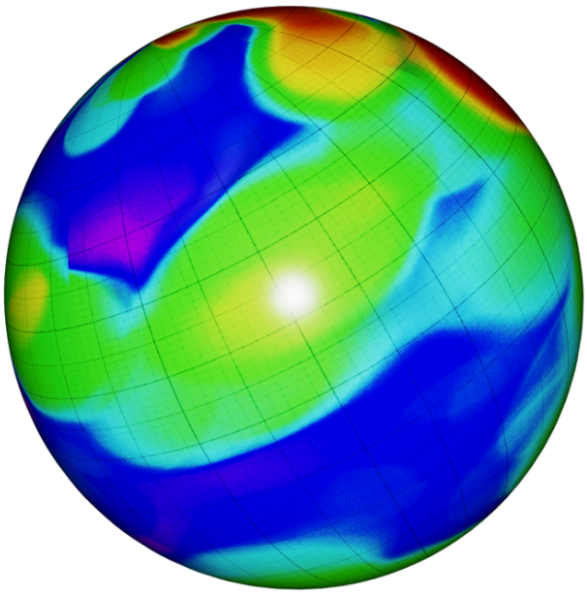


Figure 5.1: A heatmap over the Earth's surface illustrating distributional representation. Brighter regions indicate higher visit frequency. The same principle applies to embedding manifolds: instead of averaging item embeddings to a single point, EMDE represents which regions of the space a user occupies. Figure adapted from [69].

5.1 Introduction

Chapter 4 showed how Cleora transforms behavioral hypergraphs into dense entity embeddings. Every product, brand, and category receives an embedding vector, as does every user. However, recommendation requires representing a user's entire interaction history in a form suitable for neural networks. The standard approach is averaging: take all product embeddings from a user's purchase history and compute their mean. This produces a fixed-size vector regardless of history length, which is computationally convenient. But it destroys information that matters for prediction.

Consider what happens when a user has purchased both a laptop and a novel. The average of these embeddings falls somewhere between electronics and books. This point does not represent either interest; it represents a phantom preference that exists nowhere in the user's actual history.

5.1.1 From Points to Distributions

The core idea behind EMDE is to represent user histories as distributions over the embedding space rather than as single points. To build intuition, consider an analogy with geographic data. Suppose one wants to represent a person's travel history. Computing the average latitude and longitude of all places they visited would be almost useless: someone who splits time between Warsaw and Tokyo would average to somewhere in Siberia, a place they have never been.

A heatmap works better. Mark each visited location on a globe, with intensity proportional to visit frequency. The result shows which regions the person actually frequents. Warsaw and Tokyo both light up; Siberia stays dark. This representation preserves distributional structure that averaging destroys (Figure 5.1).

The same idea applies to product embeddings. If a user bought many pairs of running shoes but no swimwear, a heatmap over the product manifold would show the running shoe region illuminated and the swimwear region dark. This captures something that a mean embedding cannot: localized, structured preferences.

The challenge is making this work in practice. Geographic heatmaps succeed because the Earth's surface is two-dimensional with natural divisions. Embedding spaces have hundreds of dimensions with no pre-existing regions. A viable solution must satisfy several constraints simultaneously:

- **Fixed size regardless of history length**, since some users have thousands of interactions while others have just a few.
- **Scalability to high-dimensional embeddings** without requiring exponentially many buckets.
- **Preservation of distributional structure**, capturing which regions a user has visited rather than just the average location.
- **Computational efficiency** for production systems serving millions of users in real time.
- **Continuous vector output** suitable as input to neural networks.

Standard density estimation methods fail these constraints. Histograms require exponentially many buckets as dimensionality grows. Kernel density estimation becomes computationally expensive in high dimensions and produces function outputs rather than fixed-size vectors suitable for neural network input.

5.1.2 Proposed Approach

Before getting to the algorithm, there is a structural fact worth stating explicitly. Embeddings learned from behavioral data are not scattered uniformly through $\mathbb{R}^d$. Human purchasing and consumption patterns are organized around a finite set of categories, styles, and needs. Items therefore cluster near a low-dimensional manifold: a structured subspace whose intrinsic dimension is far smaller than $d = 512$ or $d = 4096$. This matters because classical density estimation would need exponentially many buckets to cover all of $\mathbb{R}^d$. On a lower-dimensional manifold, far fewer regions are actually populated. With K hyperplanes per partitioning creating 2^K buckets and N independent partitionings for resolution, $N \times 2^K$ entries suffice. The density is estimated over the manifold the data occupies, not the ambient space, which is precisely why the approach is tractable and why the method is named as it is.

EMDE solves this by combining ideas from two classical algorithms: locality-sensitive hashing (LSH) and Count-Min Sketch (CMS). Neither alone suffices. Standard LSH places hyperplane thresholds at zero, which works poorly when embeddings cluster far from the origin. CMS ignores geometry entirely, treating items as arbitrary identifiers.

The key modification is *density-dependent partitioning*: positioning hyperplane thresholds at quantiles of the actual data distribution rather than at zero. This ensures partitions cut through regions where items exist, not through empty space. Multiple independent partitionings provide resolution that grows multiplicatively while storage grows only linearly.

The output is a fixed-size *sketch* that approximates density over the embedding manifold. Sketches are additive (new interactions can be incorporated without reprocessing entire histories), integrate multiple

modalities through simple concatenation, and support cold-start items naturally through content-based embedding.

5.1.3 Chapter Organization

The remainder of this chapter develops these ideas formally. Section 5.2 reviews Count-Min Sketch and locality-sensitive hashing, establishing why neither alone addresses the behavioral aggregation problem. Section 5.3 presents the EMDE algorithm itself. Section 5.4 discusses properties relevant to production systems. Section 5.5 describes how EMDE enables a complete recommendation pipeline. Section 5.6 presents experimental results. Section 5.7 analyzes why density estimation succeeds where temporal graph architectures fail. Section 5.9 summarizes findings and evaluates the chapter's hypotheses.

5.2 Theoretical Foundations

EMDE draws on two classical algorithms: Count-Min Sketch for compact frequency estimation and locality-sensitive hashing for geometry-aware space partitioning. This section reviews both, focusing on properties relevant to behavioral density estimation and limitations that motivated the modifications in EMDE.

5.2.1 Count-Min Sketch

Count-Min Sketch (CMS) is a data structure for estimating frequencies in data streams [11]. Items arrive sequentially from a large universe, and the goal is to answer frequency queries without storing exact counts for every distinct item.

CMS maintains a two-dimensional array of counters with width w and depth d (here d and w denote CMS parameters, distinct from the embedding dimension d used elsewhere; the EMDE adaptation uses N and B for these quantities), along with d independent hash functions $h_1, \ldots, h_d$, each mapping items to positions in $\{1, \ldots, w\}$. When item x arrives, one counter in each row is incremented:

$$\text{count}[j, h_j(x)] \leftarrow \text{count}[j, h_j(x)] + 1 \quad \text{for } j = 1, \ldots, d \tag{5.1}$$

To estimate the count of item x, the minimum across rows is taken:

$$\hat{f}(x) = \min_{j=1}^{d} \text{count}[j, h_j(x)] \tag{5.2}$$

The minimum is important because hash collisions cause counters to be incremented by items other than x, potentially inflating individual estimates. Taking the minimum across independent rows reduces the probability that all d estimates are inflated by the same collision.

Two properties of CMS matter for EMDE. First, sketches are *additive*: given two CMS structures built from disjoint streams, their element-wise sum is a valid CMS for the combined stream. Second, sketch size depends only on accuracy parameters, not on the number of distinct items or stream length.

The limitation is that CMS is geometry-blind. Hash functions treat items as opaque identifiers. Running shoes and hiking boots, despite being similar products, hash to unrelated buckets. The geometric structure of embeddings is ignored entirely.

5.2.2 Locality-Sensitive Hashing

Locality-sensitive hashing (LSH) takes the opposite approach [32]. Where CMS ignores geometry, LSH is designed specifically to respect it: similar items should land in the same bucket with high probability, while dissimilar items are separated.

For vectors in $\mathbb{R}^d$, a common LSH scheme uses random hyperplanes. Drawing a random vector $\mathbf{r}$ from a standard Gaussian distribution, the hash function is:

$$h(\mathbf{v}) = \text{sign}(\mathbf{v} \cdot \mathbf{r}) \tag{5.3}$$

This partitions the space into two half-spaces. The probability that two vectors $\mathbf{u}$ and $\mathbf{v}$ collide depends on the angle θ between them:

$$P[h(\mathbf{u}) = h(\mathbf{v})] = 1 - \frac{\theta}{\pi} \tag{5.4}$$

Vectors pointing in similar directions collide often; those pointing in opposite directions rarely collide. This is the locality-sensitive property.

With K independent random vectors $\mathbf{r}_1, \ldots, \mathbf{r}_K$, each item receives a K-bit code:

$$c(\mathbf{v}) = \sum_{k=1}^{K} \mathbf{1}[\mathbf{v} \cdot \mathbf{r}_k \geq 0] \cdot 2^{k-1} \tag{5.5}$$

This maps each vector to one of 2^K buckets.

LSH has its own limitation, nearly the mirror of the CMS problem. It partitions space geometrically but provides no aggregation mechanism. Given a set of item embeddings, LSH assigns each to a bucket, but there is no natural way to combine these assignments into a distributional summary. Additionally, standard LSH places thresholds at zero, which can produce highly imbalanced partitions when data is not centered at the origin.

5.2.3 The Gap

Neither algorithm alone meets the requirements for behavioral aggregation. CMS provides compact, additive aggregation but ignores geometry. LSH respects geometry but provides no aggregation and suffers from partition imbalance when data is not centered.

Consider a user who bought running shoes, a yoga mat, and a protein bar. CMS would hash these items to essentially random buckets; a new item like running socks would hash to its own unrelated bucket, receiving no benefit from its similarity to running shoes. Standard LSH would assign each item to a geometric bucket, so running socks might share a bucket with running shoes, but LSH provides no counting mechanism. Moreover, if fitness products cluster far from the origin, a zero-threshold hyperplane might place them all in a single bucket, wasting resolution.

What is needed is a combination of CMS-style aggregation with LSH-style geometry awareness, while fixing the partition imbalance problem. The next section describes how EMDE achieves this.

5.3 The EMDE Algorithm

This section presents the EMDE algorithm in detail. Table 5.1 summarizes the notation introduced in this chapter.

Table 5.1: EMDE notation. Core notation from Table 2.1 applies throughout.

Symbol	Description
Sketch Structure	
N	Number of independent partitionings (sketch depth)
B	Number of buckets per partitioning, $B = 2^K$ (sketch width)
K	Number of hyperplanes per partitioning
$\mathbf{S} \in \mathbb{R}^{N \times B}$	Density sketch matrix
$\mathbf{S}[j, b]$	Count in bucket b of partitioning j
Partitioning	
$\mathbf{r}_k \in \mathbb{R}^d$	Random hyperplane normal vector ($k = 1, \ldots, K$)
$b_k \in \mathbb{R}$	Density-dependent threshold for hyperplane k
$c^{(j)}(\mathbf{e}) \in \{0, \ldots, B-1\}$	Bucket code for embedding $\mathbf{e}$ under partitioning j
$\mathbf{h}_j(\mathbf{e}) \in \{0,1\}^B$	One-hot encoding of bucket assignment
Sketch Variants	
$\tilde{\mathbf{S}}$	L_2-normalized sketch (for network input)
$\hat{\mathbf{S}}$	L_1-normalized sketch (for target distributions)
$\mathbf{S}_{\text{input}}$	Input sketch (past behavior, before time split)
$\mathbf{S}_{\text{target}}$	Target sketch (future behavior, for training)
$\mathbf{S}_{\text{output}}$	Predicted sketch (network output, for inference)
Multimodal Sketches	
M	Number of modalities
$\mathbf{S}^{(m)}$	Sketch for modality m
$\mathbf{S}^{\text{multi}}$	Concatenated multimodal sketch

The sketch structure mirrors Count-Min Sketch, where N corresponds to the depth (number of hash functions) and B to the width (number of counters per hash function). The key innovations are density-dependent partitioning, which addresses the imbalanced bucket problem from standard LSH, and multiple independent partitionings, which provide high resolution with compact storage.

5.3.1 Density-Dependent Partitioning

The problem with standard LSH is that hyperplanes pass through the origin regardless of where data actually resides. If embeddings cluster in some region far from the origin, a hyperplane through zero may put 90% of items on one side and 10% on the other, or miss the data entirely.

The solution is to use the median of projected data as the threshold rather than zero. For hyperplane k with random direction $\mathbf{r}_k$ and a catalog of item embeddings $\{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$:

$$b_k = \text{median}(\{\mathbf{r}_k \cdot \mathbf{e}_1, \mathbf{r}_k \cdot \mathbf{e}_2, \ldots, \mathbf{r}_k \cdot \mathbf{e}_n\}) \tag{5.6}$$

The hash function then becomes:

$$h_k(\mathbf{e}) = \mathbf{1}[\mathbf{r}_k \cdot \mathbf{e} \geq b_k] \tag{5.7}$$

By construction, roughly half the items fall on each side of the threshold. The hyperplane retains its random orientation, preserving the locality-sensitive property, but the partition is now balanced.

With K random directions $\mathbf{r}_1, \ldots, \mathbf{r}_K$ and corresponding thresholds $b_1, \ldots, b_K$, each item receives a K-bit code:

$$c(\mathbf{e}) = \sum_{k=1}^{K} \mathbf{1}[\mathbf{r}_k \cdot \mathbf{e} \geq b_k] \cdot 2^{k-1} \tag{5.8}$$

This maps each item to one of $B = 2^K$ buckets. Because each hyperplane independently bisects the data, buckets tend to be reasonably balanced.

I call this scheme Density-Dependent LSH. The random directions and thresholds are computed once from the item catalog during preprocessing and then fixed.

5.3.2 Multiple Independent Partitionings

A single partitioning with K hyperplanes creates $B = 2^K$ buckets. With $K = 7$, that is 128 buckets, insufficient resolution for a catalog with millions of products. Increasing K causes the number of buckets to grow exponentially.

The solution, borrowed from Count-Min Sketch, is to use N independent partitionings, each with its own random directions and thresholds. An item is characterized by a vector of N indices, one per partitioning.

The power of this approach comes from intersection. If two items collide in one partitioning, they probably will not collide in another, since random directions are independent. The probability of collision in all N partitionings decreases exponentially with N. With N partitionings of B buckets each, effective resolution approaches B^N while storage is only $N \times B$.

Figure 5.2 illustrates this principle. The experiments used N between 9 and 30 partitionings depending on the dataset and task, with $K = 7$ (128 buckets per partitioning) for session-based recommendation and larger widths for top-k recommendation.

5.3.3 Sketch Construction

Given N partitionings, constructing a sketch from a set of item embeddings is straightforward. Let $\mathcal{S}$ be the set of item embeddings in a user's history. The sketch is a matrix $\mathbf{S}$ of size $N \times B$, initialized to zeros.

For each item embedding $\mathbf{e}_i$ and each partitioning $j \in \{1, \ldots, N\}$, compute the bucket index $c^{(j)}(\mathbf{e}_i) \in \{0, \ldots, B-1\}$ and increment:

$$\mathbf{S}[j, c^{(j)}(\mathbf{e}_i)] \leftarrow \mathbf{S}[j, c^{(j)}(\mathbf{e}_i)] + 1 \tag{5.9}$$

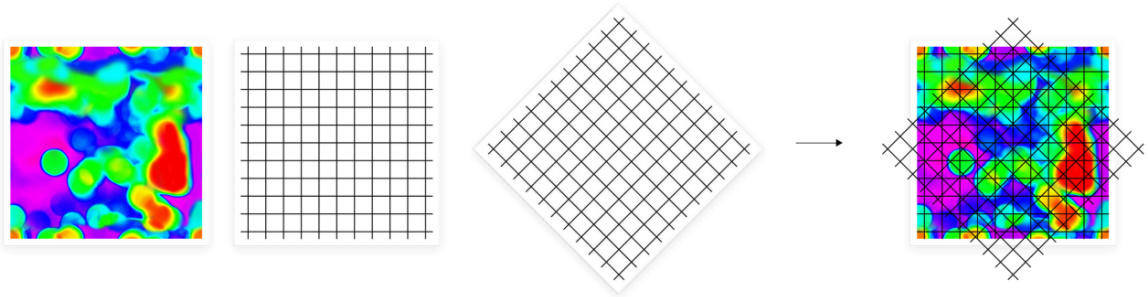


Figure 5.2: Multiple independent partitionings provide multiplicative resolution. Left: a heatmap over an embedding manifold. Middle: two different coarse grids. Right: their intersection creates many more distinct regions than either grid alone. Figure adapted from [69].

After processing all items, $\mathbf{S}[j, c]$ contains the count of items whose code under partitioning j equals c. Each row is a histogram over that partitioning's buckets; the full sketch is a stack of N such histograms, each providing an independent view of how the user's items distribute across embedding space.

An equivalent formulation uses one-hot encoding. For each item $\mathbf{e}_i$ and partitioning j, create a one-hot vector $\mathbf{h}_j(\mathbf{e}_i)$ of length 2^K with a 1 at position $c^{(j)}(\mathbf{e}_i)$:

$$\mathbf{S}[j, :] = \sum_{\mathbf{e} \in \mathcal{S}} \mathbf{h}_j(\mathbf{e}) \tag{5.10}$$

This makes additivity explicit: the sketch of a union equals the sum of individual sketches. In practice, item codes are precomputed and cached during preprocessing, reducing sketch construction to summing precomputed vectors.

5.3.4 Weighted Aggregation

Not all interactions are equally informative. A purchase signals stronger intent than a page view; a recent interaction is more relevant than one from months ago. EMDE handles this through weighted aggregation. Instead of incrementing by 1, increment by a weight w_i:

$$\mathbf{S}[j, c^{(j)}(\mathbf{e}_i)] \leftarrow \mathbf{S}[j, c^{(j)}(\mathbf{e}_i)] + w_i \tag{5.11}$$

Common weighting schemes include temporal decay ($w_i = \exp(-\lambda \Delta t_i)$), interaction type weighting (purchases weighted higher than views), and engagement strength (weighting by dwell time or rating). Weighted sketches remain additive.

5.3.5 Encoding Auxiliary Numerical Features

Behavioral profiles often benefit from auxiliary numerical features: number of purchases, days since last activity, account age. These features span orders of magnitude, making normalization unstable for heavy-tailed distributions.

A multi-scale sinusoidal encoding, termed Fourier Feature Encoding (FFE), sidesteps these issues. Each scalar value x is transformed into a $2S$-dimensional vector:

$$\text{FFE}(x) = \left[\sin\left(\frac{x}{2^{s_1}}\right), \ldots, \sin\left(\frac{x}{2^{s_S}}\right), \cos\left(\frac{x}{2^{s_1}}\right), \ldots, \cos\left(\frac{x}{2^{s_S}}\right)\right] \tag{5.12}$$

where $s_1, \ldots, s_S$ are scale parameters. The encoding uses $S = 8$ scales with $s_i \in \{-1, 0, 1, 2, 3, 4, 5, 6\}$, producing a 16-dimensional encoding per feature. The trigonometric functions bound outputs to $[-1, 1]$ regardless of input magnitude, while different scales capture variation at different granularities.

The encoded features are concatenated with the EMDE sketch before the neural network, allowing the network to learn how to combine distributional and numerical signals without specialized fusion architecture.

5.3.6 Normalization

Before feeding sketches to a neural network, normalization ensures comparable magnitudes regardless of history length.

For input sketches, L_2 normalization is applied independently to each row:

$$\tilde{\mathbf{S}}[j, :] = \frac{\mathbf{S}[j, :]}{\|\mathbf{S}[j, :]\|_2} \tag{5.13}$$

For target sketches used in training, L_1 normalization converts counts into probability distributions:

$$\hat{\mathbf{S}}[j, :] = \frac{\mathbf{S}[j, :]}{\|\mathbf{S}[j, :]\|_1} \tag{5.14}$$

Each row then sums to 1 and can be interpreted as a probability distribution over buckets, motivating the cross-entropy loss described in Section 5.5.

5.3.7 Multimodal Sketches

Products have multiple representations: collaborative embeddings from co-purchase patterns (computed using Cleora), text embeddings from descriptions, image embeddings from photos. Each modality captures different information in a geometrically distinct space.

EMDE handles multiple modalities by constructing separate sketches and concatenating them. If there are M modalities, each producing a sketch of size $N \times B$, the multimodal sketch is their vertical concatenation:

$$\mathbf{S}^{\text{multi}} = [\mathbf{S}^{(1)}; \mathbf{S}^{(2)}; \ldots; \mathbf{S}^{(M)}] \tag{5.15}$$

Each modality uses its own partitioning computed from its own embedding space. This simple concatenation approach is effective because sketches already preserve fine-grained distributional information. The downstream network learns implicitly how to weight and combine modalities through the training objective, without requiring learned fusion mechanisms or architectural modifications when new modalities are added.

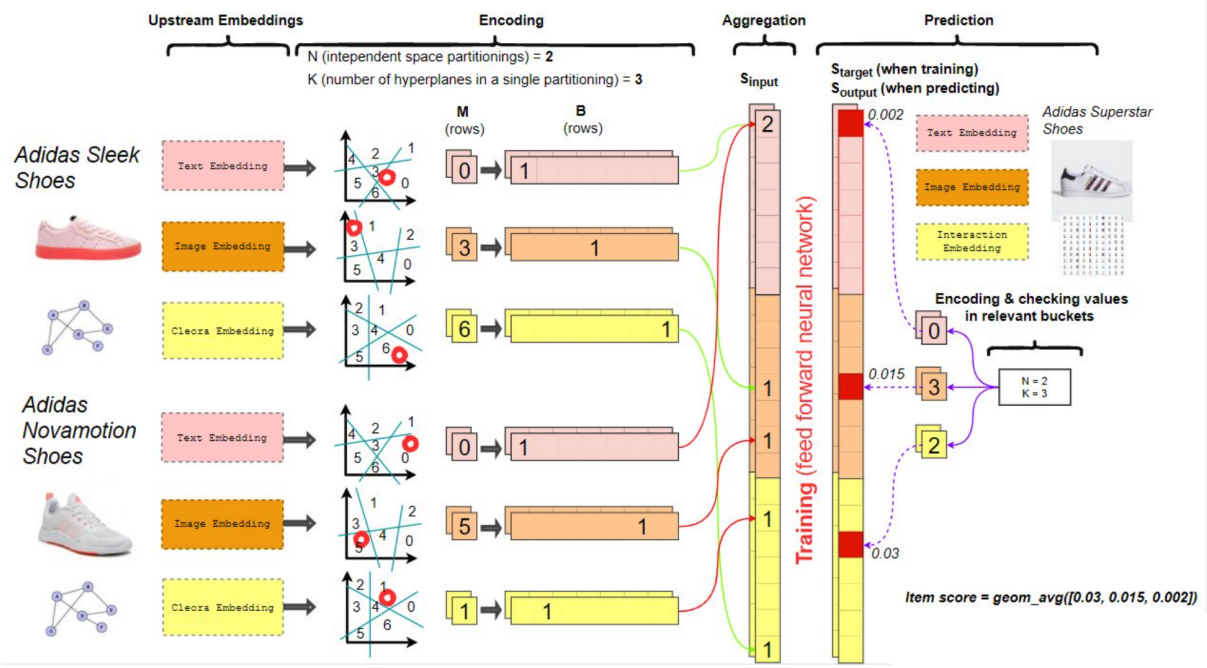


Figure 5.3: The EMDE pipeline. Upstream multimodal embeddings are mapped via density-dependent hyperplane partitions into bucket indices, then aggregated into fixed-size sketches. A feed-forward network maps the input sketch $\mathbf{S}_{\text{input}}$ to the output sketch $\mathbf{S}_{\text{output}}$; during training, the target $\mathbf{S}_{\text{target}}$ provides supervision. At inference, candidate items are encoded with the same partitions and scored by reading predicted sketch values at their bucket positions via geometric mean. The figure uses a simplified example with $N = 2$ partitionings and $K = 3$ hyperplanes; experiments use larger values (see Section 5.6.3).

5.3.8 The Complete Pipeline

Figure 5.3 illustrates the complete EMDE pipeline with a concrete example. Consider a user who has purchased two items (Adidas Sleek Shoes and Adidas Novamotion Shoes) and will next purchase Adidas Superstar Shoes. Each item has embeddings from multiple modalities (text description, product image, interaction graph), though the procedure is identical for any modality.

The pipeline proceeds from left to right in four stages:

Upstream Embeddings. Each item is represented by embeddings from multiple sources. In the figure, three modalities are shown: a text embedding from a sentence transformer, an image embedding (labeled "Deep Embedding" in the figure, as it is computed by a deep vision network), and a collaborative embedding from Cleora (labeled "Cleora Embedding" in the figure, capturing co-purchase interaction patterns). These upstream embeddings serve as the input to the EMDE pipeline and need not come from any particular source; the procedure applies uniformly to any embedding type.

Encoding. For each modality, every item embedding is projected onto a family of random hyperplanes with density-dependent thresholds. The resulting sign patterns define bucket indices, so that geometrically similar items map to the same buckets with high probability. In the figure, each item receives one bucket index per partition, which is then one-hot encoded into a sparse vector. Because these projections depend

only on the item embeddings and the fixed hyperplane parameters, bucket codes can be precomputed and cached for the entire catalog.

Aggregation. To build a behavioral profile, the one-hot vectors of all historical items are summed elementwise. Optional weights (recency decay, interaction type) can scale each item's contribution. After normalization, concatenating sketches across modalities yields a single fixed-size profile that summarizes the user's past behavior. Crucially, this profile has identical dimensions regardless of whether the history contains two items or two thousand. The same construction applied to future items produces the target profile used during training.

Prediction. A feed-forward network learns to map past profiles to future profiles using cross-entropy loss over sketch buckets. At inference time, the network outputs a predicted future profile for the current user. To score any candidate item, including cold-start items with no interaction history, encode it into bucket indices using the same hyperplanes as in the encoding step. The recommendation score is obtained by looking up the predicted profile values at those bucket positions and aggregating via geometric mean:

$$\text{score}(i) = \left(\prod_{m=1}^{M} \prod_{j=1}^{N} \hat{\mathbf{S}}^{(m)}[j, c^{(j,m)}(i)] \right)^{1/(NM)} \tag{5.16}$$

where $\hat{\mathbf{S}}^{(m)}[j, c^{(j,m)}(i)]$ is the predicted probability at the bucket containing item i for modality m and partitioning j.

The geometric mean requires consistent evidence across partitionings: an item scores well only if it has reasonable probability in most partitions, preventing single high values from dominating. Under the assumption that independent partitionings provide conditionally independent probability estimates, the geometric mean is the natural aggregation: it equals the geometric average of NM independent posterior estimates, which is what one would compute by multiplying probabilities under independence and then normalizing by the number of estimates. This lookup-based scoring requires only NM table accesses per candidate, independent of catalog size or embedding dimension. For a million-item catalog with $M = 3$ modalities and $N = 10$ partitionings, scoring completes in seconds on a single CPU, avoiding the large softmax layers or beam search decoding that other approaches require.

5.4 Properties

The sketch representation has several properties that matter for production recommendation systems.

Fixed dimensionality. Sketch size is determined entirely by algorithm parameters ($N \times B$), independent of history length, catalog size, or embedding dimension. A user with 5 purchases and one with 5,000 produce sketches of identical shape, simplifying neural network design and bounding memory usage.

Additive compositionality. Sketches are additive: the sketch of a union equals the sum of individual sketches. This enables incremental updates (add new interaction contributions to existing sketches),

flexible aggregation (precompute item sketches and assemble user sketches on the fly), and time-windowed representations (maintain separate sketches for different periods and combine with different weights).

Cold-start support. New items with content features receive meaningful bucket codes immediately, regardless of interaction history. The text embedding of "blue running shoes, men's size 10" places the item in the running shoe region of the manifold. At prediction time, new items are scored through the same mechanism as established items.

Computational efficiency. Sketch construction involves looking up precomputed codes and summing one-hot vectors: $O(|\mathcal{S}| \cdot M \cdot N)$ integer operations for $|\mathcal{S}|$ items, M modalities, and N partitionings. With typical values ($|\mathcal{S}| = 50$, $M = 3$, $N = 10$), construction takes microseconds. Scoring is similarly efficient, independent of embedding dimension.

Graceful degradation with missing modalities. When a modality is unavailable, its sketch positions are zero. The network learns to rely on available signals. Users with rich interaction history but no demographic data and new users with demographic data but sparse history can both receive meaningful predictions.

Relationship to Bloom filters. There is an instructive analogy to Bloom filters. A Bloom filter represents set membership by hashing an item into multiple positions of a bit array; querying an item checks whether all its hash positions are set. EMDE's output sketch plays an analogous role for full distributions: each item maps to NM bucket positions across partitionings and modalities, and the network predicts probability mass at those positions rather than bits. The geometric mean scoring then plays the role of the Bloom filter's conjunction test. This framing also explains why the method avoids a softmax over all items: instead of one wide classification layer, EMDE decomposes the prediction into NM narrow softmax layers of width B, each providing an independent approximation of the output distribution.

5.5 EMDE for Recommendation

This section describes how to use sketches for recommendation: framing the learning problem, training, and generating predictions.

5.5.1 Problem Formulation

EMDE frames recommendation as conditional density prediction. Given a user's past interactions, the goal is to predict the density of their future interactions over the item manifold. The target is a distribution, not a single item: a user who bought running shoes might next buy running socks, a water bottle, or a fitness tracker. The model should assign probability mass to each plausible outcome.

In sketch terms: given an input sketch $\mathbf{S}_{\text{past}}$ representing items before time t, predict an output sketch $\mathbf{S}_{\text{future}}$ representing items after time t. The neural network learns a function f_θ such that $f_\theta(\mathbf{S}_{\text{past}}) \approx \mathbf{S}_{\text{future}}$.

5.5.2 Training Data

For each user, the interaction history is split at a time point. Events before the split form the input set; events after form the target set. In training notation, the past sketch becomes $\mathbf{S}_{\text{input}}$ and the future sketch becomes $\mathbf{S}_{\text{target}}$. The pair $(\mathbf{S}_{\text{input}}, \mathbf{S}_{\text{target}})$ becomes one training example.

For session-based recommendation, the evaluation follows the protocol of Ludewig et al. [41]. Each dataset is split into five contiguous time slices. Within each session, input consists of all items except the last, and the target is the last item. The input sketch comprises two parts: one sketch for the last item before the target (providing recency signal), and another aggregating earlier items with exponential decay. These are concatenated before the network.

For top-k recommendation, 80% of randomly shuffled user items go into the input sketch and the remaining 20% into the target sketch.

5.5.3 Neural Network Architecture

The architecture is deliberately simple. The sketch representation performs the heavy representational lifting; the network need only learn how behavioral distributions evolve.

For session-based recommendation, the architecture is a three-layer residual feed-forward network with 3,000 neurons per layer, leaky ReLU activations, and batch normalization. For top-k recommendation, a single hidden layer with 12,000 neurons worked better. No attention mechanisms, no recurrence, no graph convolutions. The network treats sketch entries symmetrically, matching the permutation-invariant nature of set aggregation.

5.5.4 Loss Function

The target sketch is L_1-normalized to form a probability distribution over buckets. The network output is passed through softmax along each row. The loss is the KL-divergence (equivalent to cross-entropy with fixed target) between predicted and target distributions, averaged across rows:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^{N} \text{CE}\big(\hat{\mathbf{S}}_{\text{target}}[j,:],\ f_\theta(\tilde{\mathbf{S}}_{\text{input}})[j,:]\big) \tag{5.17}$$

This formulation bypasses the need for either a wide softmax over all items or negative sampling, both of which cause stability and efficiency problems. The softmax is only over B buckets (typically 128 or 350), not millions of items.

5.5.5 Training Configuration

For session-based experiments, training used Adam with dataset-specific learning rates (0.0005 to 0.004), batch sizes of 256 or 512, and learning rate decay (factor 0.5 to 1.0) after each epoch. Training ran for 5 to 50 epochs depending on dataset size. Sketch parameters were $N = 9$ or 10 partitionings with $K = 7$ ($B = 128$ buckets).

When aggregating items, exponential time decay was applied following $w(\Delta t) = \alpha \cdot \gamma^{\Delta t}$, where Δt is measured in seconds, with scaling factor $\alpha \approx 0.9$ and decay base $\gamma = 0.01$:

$$\text{Sketch}(t_2) = \alpha \cdot \gamma^{(t_2 - t_1)} \cdot \text{Sketch}(t_1) \tag{5.18}$$

For top-k experiments, training used Adam with learning rate 0.001 and batch size 256. Each modality used $N = 30$ partitionings with $K = 7$ hyperplanes ($2^7 = 128$ buckets per partitioning).

One trick that helped across settings: adding random codes alongside LSH-based codes. For each item, some bucket indices are random rather than geometry-based, allowing the model to distinguish items with identical LSH codes and capture item-specific popularity patterns.

Training is fast. On a single RTX 2080 Ti (11GB), session-based models trained in minutes to hours. Graph neural network baselines SR-GNN and TAGNN required over a week for hyperparameter tuning on larger datasets and eventually timed out.

5.5.6 Inference

At inference time:

1. Construct the input sketch $\mathbf{S}_{\text{input}}$ with time decay
2. Run the network to get $\mathbf{S}_{\text{output}} = f_\theta(\mathbf{S}_{\text{input}})$
3. Apply softmax to each row
4. For each candidate item, look up bucket codes and read corresponding values from $\mathbf{S}_{\text{output}}$
5. Aggregate via geometric mean
6. Score all items in the catalog and return the top-scoring ones

Unlike methods computing softmax over the entire catalog, EMDE scores items independently through lookups. For a million items with $M = 3$ modalities and $N = 10$ partitionings, scoring completes in seconds on a single CPU.

5.6 Experiments

This section evaluates EMDE on standard recommendation benchmarks. The experiments address all three hypotheses: H1 through comparisons with point-based methods and embedding ablations; H2 through unimodal versus multimodal comparisons; H3 through comparison with deep sequential and graph neural network architectures.

5.6.1 Datasets

Session-based recommendation. The evaluation uses six datasets from the framework of Ludewig et al. [41]:

- **RETAIL**: E-commerce data with item properties (price, category, vendor). 212,182 actions, 59,962 sessions, 31,968 items.
- **DIGI** (Diginetica): CIKM 2016 challenge data with purchases, product names, and search queries.
- **RSC15** (Yoochoose): RecSys 2015 challenge data. 5,426,961 actions, 1,375,128 sessions, 28,582 items.
- **30MU** (30Music): Last.fm listening logs with user-created playlists. 638,933 actions, 37,333 sessions, 210,633 items.
- **AOTM**: Art of the Mix playlist data. 306,830 actions, 21,888 sessions, 91,166 items.
- **NOWP**: Twitter-collected playlists. 271,177 actions, 27,005 sessions, 75,169 items.

Results are averaged across five contiguous time slices.

Top-k recommendation. Two movie rating datasets from Ferrari Dacrema et al. [19] are used:

- **MovieLens-20M**: 20,108 movies, 116,677 training users.
- **Netflix Prize**: 17,768 movies, 383,435 training users.

Ratings of 4 or higher are treated as positive interactions.

5.6.2 Baselines

For session-based recommendation: S-KNN and VS-KNN (nearest-neighbor methods), AR and SR (association rules), CT (context trees [42]), GRU4Rec [30], NARM [37], STAMP [40], SR-GNN [82], and TAGNN [85].

For top-k recommendation: EASE [61], MultVAE [38], SLIM [46], and RP3beta [47].

5.6.3 EMDE Configuration

Item interactions are grouped by user, artist, and session to create interaction graphs. Cleora computes embeddings with random search over iterations $[1, 2, 3, 4, 5]$, dimension $[512, 1024, 2048, 4096]$, and interaction groupings.

The neural network is a three-layer residual feed-forward network with 3,000 neurons, leaky ReLU, and batch normalization. Session-based experiments use $K = 7$ (128 buckets) with $N = 9$–10 partitionings. Top-k experiments use a single 12,000-neuron hidden layer with 30×350 sketches.

Results are reported for EMDE (interaction embeddings only) and EMDE MM (multimodal).

5.6.4 Session-Based Recommendation Results

Table 5.2 presents results across all six datasets.

EMDE achieves the highest MRR@20 on five of six datasets: RETAIL, RSC15, NOWP, 30MU, and AOTM. On DIGI, EMDE ranks second in MRR but leads on Precision, Recall, Hit Rate, and mAP.

Table 5.2: Session-based recommendation results. Top 5 methods per dataset. All metrics computed at cutoff $k = 20$: only the top 20 ranked items are considered. Best in bold.

Dataset	Model	MRR@20	P@20	R@20	HR@20	mAP@20
RETAIL	EMDE	**0.3524**	0.0526	**0.4709**	**0.5879**	0.0282
	VS-KNN	0.3395	0.0531	0.4632	0.5745	0.0278
	S-KNN	0.3370	**0.0532**	0.4707	0.5788	**0.0283**
	TAGNN	0.3266	0.0463	0.4237	0.5240	0.0249
	GRU4Rec	0.3237	0.0502	0.4559	0.5669	0.0272
RSC15	EMDE	**0.3104**	0.0730	0.4936	0.6619	0.0346
	CT	0.3072	0.0654	0.4710	0.6359	0.0316
	NARM	0.3047	**0.0735**	**0.5109**	**0.6751**	**0.0357**
	STAMP	0.3033	0.0713	0.4979	0.6654	0.0344
	SR	0.3010	0.0684	0.4853	0.6506	0.0332
DIGI	VS-KNN	**0.1784**	0.0584	0.3668	0.4729	0.0249
	EMDE	0.1724	**0.0602**	**0.3753**	**0.4761**	**0.0258**
	S-KNN	0.1714	0.0596	0.3715	0.4748	0.0255
	TAGNN	0.1697	0.0573	0.3554	0.4583	0.0249
	GRU4Rec	0.1644	0.0577	0.3617	0.4639	0.0247
NOWP	EMDE	**0.1108**	**0.0617**	**0.1847**	**0.2665**	0.0179
	CT	0.1094	0.0287	0.0893	0.1679	0.0065
	GRU4Rec	0.1076	0.0449	0.1361	0.2261	0.0116
	SR	0.1052	0.0466	0.1366	0.2002	0.0133
	SR-GNN	0.0981	0.0414	0.1194	0.1968	0.0101
30MU	EMDE	**0.2673**	**0.1102**	**0.2502**	**0.4104**	**0.0330**
	CT	0.2502	0.0308	0.0885	0.2882	0.0058
	SR	0.2410	0.0816	0.1937	0.3327	0.0240
	GRU4Rec	0.2369	0.0617	0.1529	0.3273	0.0150
	NARM	0.1945	0.0675	0.1486	0.2956	0.0155
AOTM	EMDE	**0.0123**	0.0083	0.0227	0.0292	0.0020
	CT	0.0111	0.0043	0.0126	0.0191	0.0006
	NARM	0.0088	0.0050	0.0146	0.0202	0.0009
	STAMP	0.0088	0.0020	0.0063	0.0128	0.0003
	S-KNN	0.0054	**0.0139**	**0.0390**	**0.0417**	**0.0037**

The AOTM result warrants qualification. Although EMDE leads on MRR@20, S-KNN dominates on all other metrics: Precision (0.0139 vs 0.0083), Recall (0.0390 vs 0.0227), Hit Rate (0.0417 vs 0.0292), and mAP (0.0037 vs 0.0020). AOTM is a music playlist dataset where co-occurrence structure is both strong and sparse: most playlists share few items. Nearest-neighbor methods exploit exact co-occurrence directly and handle sparse tail items well. EMDE's geometry-based representations smooth over these fine-grained co-occurrences, which helps on dense e-commerce datasets but hurts on sparse playlist data with many unique items.

Evidence for H3. These results support hypothesis H3 on two fronts. First, EMDE uses a simple feed-forward architecture while competing methods employ recurrent networks (GRU4Rec, NARM), attention

mechanisms (STAMP), or graph neural networks (SR-GNN, TAGNN). Second, the GNN methods struggled to scale: TAGNN and SR-GNN required over a week for hyperparameter tuning on RSC15 and 30Music before timing out. EMDE trained on these datasets in under one hour. The fact that a three-layer feed-forward network over density sketches matches or exceeds models with substantially more inductive bias confirms that the sketch captures structure the network would otherwise need to learn.

Evidence for H1. The session-based results provide indirect evidence for H1: EMDE's recency-weighted set representation, which preserves distributional structure rather than collapsing histories to a single point, outperforms architectures that explicitly model item-to-item transitions. The more direct test of H1 comes from the ablation in Section 5.6.7 (Table 5.8): replacing structured manifold embeddings with random vectors degrades MRR by 23.6%, confirming that performance depends on density estimation over a geometrically meaningful space, not merely on the sketching mechanism itself.

5.6.5 Multimodal Results

Table 5.3 compares unimodal EMDE against multimodal EMDE MM.

Table 5.3: Effect of multimodal data (metrics at cutoff $k = 20$). EMDE uses interaction embeddings only; EMDE MM adds text, metadata, or other available modalities.

Dataset	Model	MRR@20	P@20	R@20	HR@20	mAP@20
RETAIL	EMDE	0.3524	0.0526	0.4709	0.5879	0.0282
	EMDE MM	**0.3664**	**0.0571**	**0.5073**	**0.6330**	**0.0309**
RSC15	EMDE	0.3104	0.0730	0.4936	0.6619	0.0346
	EMDE MM	**0.3116**	**0.0743**	**0.5000**	**0.6680**	**0.0352**
DIGI	EMDE	0.1724	0.0602	0.3753	0.4761	0.0258
	EMDE MM	**0.1731**	**0.0620**	**0.3849**	**0.4908**	**0.0268**
30MU	EMDE	0.2673	0.1102	0.2502	0.4104	0.0330
	EMDE MM	**0.2703**	**0.1106**	**0.2503**	**0.4105**	**0.0331**

Evidence for H2. Multimodal sketches consistently outperform unimodal variants on every dataset where multimodal data is available. On RETAIL, Hit Rate improves from 0.5879 to 0.6330 (7.7% relative increase). The integration requires only concatenating modality-specific sketches, without learned fusion architectures.

5.6.6 Top-k Recommendation Results

Table 5.4 presents top-k results.

On MovieLens-20M, EMDE achieves best Recall@1, NDCG@10, and Recall@10. On Netflix, EMDE achieves best Recall@1 but trails EASE on other metrics. The weaker Netflix performance reflects the rating sparsity of this dataset: Netflix users rated few movies on average relative to the catalog size, giving Cleora less behavioral signal to build dense, well-separated embeddings. Methods like EASE operate directly

Table 5.4: Top-k recommendation on MovieLens-20M and Netflix Prize.

Model	Recall@1	NDCG@10	Recall@10	NDCG@20	Recall@20
MovieLens-20M					
EMDE	**0.0529**	**0.2535**	**0.2493**	0.3053	0.3523
EASE	0.0507	0.2530	0.2465	**0.3078**	**0.3542**
MultVAE	0.0514	0.2493	0.2488	0.3052	0.3589
SLIM	0.0474	0.2389	0.2318	0.2916	0.3350
RP3beta	0.0380	0.2004	0.2007	0.2501	0.3018
Netflix Prize					
EMDE	**0.0328**	0.1876	0.1652	0.2332	0.2432
EASE	0.0323	**0.2050**	**0.1782**	**0.2589**	**0.2677**
MultVAE	0.0313	0.1957	0.1756	0.2483	0.2645
SLIM	0.0307	0.1952	0.1688	0.2474	0.2552
RP3beta	0.0243	0.1191	0.0863	0.1578	0.1390

on the sparse interaction matrix without an intermediate embedding step, handling this sparsity more gracefully. EMDE's density estimation is only as good as the geometry it operates over.

The baselines in Table 5.4 are purely collaborative methods that cannot incorporate side information. To isolate the effect of multimodality, Table 5.5 compares EMDE against itself with and without additional modalities.

Table 5.5: Multimodal effects on top-k recommendation.

Model	Recall@1	NDCG@10	Recall@10	NDCG@20	Recall@20
MovieLens-20M					
EMDE MM	**0.0670**	**0.2890**	**0.2780**	**0.3358**	**0.3710**
EMDE	0.0529	0.2535	0.2493	0.3053	0.3523
Netflix Prize					
EMDE MM	**0.0388**	**0.2155**	**0.1875**	**0.2619**	**0.2645**
EMDE	0.0328	0.1876	0.1652	0.2332	0.2432

On MovieLens-20M, multimodal EMDE achieves 0.0670 Recall@1 compared to 0.0529 for unimodal (27% relative improvement).

5.6.7 Ablation Studies

Ablation experiments on RETAIL isolate the contribution of individual components.

Aggregation function. Table 5.6 compares aggregation functions for combining scores across partitionings.

Geometric mean outperforms all alternatives. The minimum (used in standard CMS) performs worst when operating on probabilities.

Table 5.6: Aggregation function comparison (RETAIL).

Aggregation	MRR	Precision	Recall	Hit Rate	mAP
Geometric mean	**0.3649**	**0.0556**	**0.4974**	**0.6202**	**0.0302**
Arithmetic mean	0.3592	0.0525	0.4786	0.5960	0.0287
Harmonic mean	0.3411	0.0517	0.4659	0.5777	0.0279
Minimum	0.3120	0.0463	0.4257	0.5250	0.0251

Pure vs. conditional EMDE. EMDE can operate in two modes. *Pure EMDE* uses the input sketch directly as a query: items are scored by how close they are to the user's history on the embedding manifold, similar to a nearest-neighbor lookup. *Conditional EMDE* places a neural network between input and output sketches, learning to transform one density into another. All experiments in this thesis use conditional EMDE unless stated otherwise. Table 5.7 quantifies the difference.

Table 5.7: Pure EMDE vs. conditional EMDE (MovieLens-20M).

Mode	NDCG@20	Recall@20
Pure EMDE	0.0930	0.1253
TopPop baseline	0.1201	0.1441
Pure EMDE + Popularity	0.1582	0.2003
Conditional EMDE	**0.3053**	**0.3523**

Conditional EMDE dramatically outperforms pure EMDE (NDCG@20: 0.305 vs 0.093). Notably, pure EMDE also falls below the popularity baseline (0.093 vs 0.1201). This is not a failure of density estimation itself, but a consequence of catalog statistics: item popularity is highly non-uniform, and many test items are popular. Pure EMDE retrieves geometrically similar items, which may be obscure films with few interactions. Popularity-blind retrieval misses the likely recommendations. Combining pure EMDE with popularity signal (Pure EMDE + Popularity) recovers to 0.1582, confirming that the geometric signal is real but insufficient alone. The conditional network learns both: it predicts which regions of the manifold are relevant and captures the global distribution structure, encoding popularity implicitly through the training objective.

Table 5.8: Effect of embedding structure (RETAIL).

Embedding Type	MRR@20	Relative Change
Structured (Cleora)	0.365	baseline
Random vectors	0.279	−23.6%

Input embedding quality.

Evidence for H1. Table 5.8 shows that replacing structured embeddings with random vectors causes MRR to drop from 0.365 to 0.279 (23.6% degradation). This confirms that EMDE's effectiveness depends on preserving geometric structure. Without meaningful geometry, EMDE degenerates to normalized CMS with random hash codes. The sharp performance degradation demonstrates that density sketches succeed precisely because they preserve distributional structure over a meaningful embedding manifold.

Table 5.9: Percentage of predictions with fully saturated scores (1.0) for top K destination nodes by recent popularity.

	TGN		**DyRep**	
K	tgbl-comment	tgbl-coin	tgbl-comment	tgbl-coin
50	90.1%	63.7%	87.1%	19.8%
100	86.5%	59.4%	86.9%	18.4%
1,000	69.6%	22.9%	86.1%	9.7%

5.7 Why Density Estimation Succeeds Where Temporal Graph Models Fail

The experiments above demonstrate strong performance on e-commerce recommendation benchmarks. This section examines a different but related task: dynamic link prediction on the Temporal Graph Benchmark (TGB) [31], which tests whether models can predict future edges in evolving graphs such as Reddit reply threads and cryptocurrency transaction networks. The datasets used here (tgbl-comment, tgbl-coin) are entirely distinct from the recommendation datasets above. The motivation for including this analysis is theoretical: if state-of-the-art temporal graph models fail at this task for structural reasons, it clarifies why density-based representations are better suited to behavioral prediction in general. This analysis is based on the NeurIPS 2023 workshop paper by Daniluk and Dąbrowski [13].

Temporal graph networks represent the state of the art for dynamic interaction data. Methods like TGN [56] and DyRep [72] maintain learned memory states for each node, updating representations as edges arrive and aggregating neighborhood information through attention. Yet closer examination reveals a fundamental limitation.

5.7.1 The Score Saturation Problem

After training TGN and DyRep on Temporal Graph Benchmark datasets [31], inspection of raw prediction scores revealed a striking pattern. For nodes popular in recent history, both models frequently output scores of exactly 1.0. Not approximately 1.0, but precisely 1.0 to floating-point precision. The sigmoid had saturated completely.

Table 5.9 quantifies this phenomenon.

On tgbl-comment (Reddit replies, 44 million edges), over 90% of top 50 popular nodes receive saturated TGN scores. If the model assigns identical scores to dozens of candidates, rankings become arbitrary. The model recognizes that popular items are good candidates but cannot discriminate among them.

5.7.2 A Diagnostic Baseline

This observation motivated a simple diagnostic: if neural models essentially track popularity, how would explicit popularity tracking perform?

Table 5.10: Dynamic link prediction MRR. PopTrack, a trivial popularity baseline, outperforms neural methods on datasets with strong global dynamics.

Method	tgbl-coin	tgbl-comment
DyRep	0.434	0.289
TGN	0.583	0.379
PopTrack	**0.725**	**0.729**

PopTrack maintains a single vector counting recent destination appearances. When a new edge arrives, the destination count increments; all counts decay by a fixed factor. No learning, no parameters, no source-specific predictions.

The results (Table 5.10) are striking. On tgbl-comment, PopTrack achieves 0.729 MRR versus 0.379 for TGN, nearly doubling performance. A zero-parameter method requiring minutes on CPU outperforms models trained for hours on GPUs.

5.7.3 Local Versus Global Dynamics

Temporal graph networks focus on *local* patterns: node-specific memories, immediate neighborhoods, particular source-destination dynamics. They miss *global* temporal dynamics: content rising in collective attention, attracting engagement in self-reinforcing cycles, then fading. On social platforms, this is not local; it sweeps through the graph system-wide.

When global dynamics dominate, models tracking only local patterns struggle. They may learn that popular nodes deserve high scores, but they cannot discriminate among popular nodes because that discrimination requires source-specific context that local memories do not adequately capture.

5.7.4 Why EMDE Avoids These Problems

Three EMDE design choices address exactly the failure modes temporal graph networks exhibit:

Distributional output. EMDE predicts a density sketch representing the distribution over possible destinations. The cross-entropy objective requires discriminating across the entire space simultaneously, forcing fine-grained discrimination without explicit hard negative mining.

Global structure through embeddings. Cleora embeddings capture global graph structure. Nodes in similar interaction patterns receive similar embeddings even without direct connections. When popularity shifts globally, the shift is reflected in which embedding space regions become active.

Non-contrastive training. Standard temporal graph networks train with negative sampling: distinguish positives from randomly sampled negatives. If negatives are easy (random unpopular nodes), learned discrimination is coarse. EMDE's cross-entropy objective sidesteps this by learning to predict full distributions.

Table 5.11 confirms this interpretation. Under hard evaluation (negatives sampled from recently popular nodes), TGN collapses from 0.583 to 0.010 MRR while EMDE maintains 0.468. The 46-fold gap demonstrates fundamentally different information capture.

Table 5.11: Performance under standard (MRR_{naive}) and hard (MRR_{top500}) evaluation on tgbl-coin.

Method	MRR_{naive}	MRR_{top500}
TGN	0.583	0.010
DyRep	0.434	0.013
EMDE	0.674	**0.468**

5.8 Hypothesis Evaluation

This section evaluates each hypothesis against the accumulated evidence.

H1: Density-based representations outperform point-based aggregation. *Supported.* EMDE achieved highest MRR@20 on five of six session-based datasets, outperforming methods that aggregate through attention-weighted combinations or graph message passing. The embedding ablation (Table 5.8) provides direct evidence: replacing structured embeddings with random vectors caused 23.6% MRR degradation. This confirms that density sketches succeed by preserving distributional structure over a meaningful manifold.

H2: Density sketching enables effective multimodal integration through concatenation. *Supported.* Multimodal sketches improved performance on all datasets where multimodal data was available (Table 5.3). On RETAIL, Hit Rate improved 7.7% from concatenating modality-specific sketches. On MovieLens-20M, multimodal EMDE achieved 27% higher Recall@1 than unimodal. Integration required only concatenation, with no learned fusion architecture.

H3: Density-based inputs reduce architectural complexity requirements. *Supported.* EMDE with a three-layer feed-forward network matched or exceeded GRU4Rec, NARM, SR-GNN, and TAGNN, methods employing recurrence, attention, and graph convolutions. The graph neural networks struggled to scale: both timed out on larger datasets after over a week of tuning. EMDE trained in under one hour. The temporal graph analysis reinforces this finding: under hard evaluation, TGN collapsed to 0.010 MRR while EMDE maintained 0.468.

Summary. Table 5.12 summarizes the hypothesis evaluation.

5.9 Summary

This chapter introduced EMDE, a method for representing user interaction histories as probability distributions over embedding manifolds. The core idea is simple: instead of averaging item embeddings to a single point, count how items distribute across regions of embedding space. The resulting sketch is a compact, fixed-size structure that preserves the distributional character of user preferences.

The technical contribution combines locality-sensitive hashing with Count-Min Sketch. Density-dependent partitioning places hyperplane thresholds at data quantiles, ensuring balanced bucket utiliza-

Table 5.12: Summary of hypothesis evaluation in this chapter.

Hypothesis	Verdict	Primary Evidence
H1: Density-based representations outperform point-based aggregation	Supported	Highest MRR@20 on 5/6 session-based datasets; 23.6% MRR degradation when geometric structure removed from embeddings
H2: Multimodal integration through concatenation	Supported	7.7% Hit Rate improvement on RETAIL; 27% Recall@1 improvement on MovieLens-20M; no fusion architecture required
H3: Density-based inputs reduce architectural complexity requirements	Supported	3-layer feed-forward network matches/exceeds GRU4Rec, NARM, SR-GNN, TAGNN; GNN baselines timed out after 1 week; 46-fold MRR gap under hard temporal evaluation

tion. Multiple independent partitionings provide resolution that grows multiplicatively while storage grows linearly. The sketch inherits useful properties from both parent algorithms: geometric awareness from LSH, additive compositionality from CMS.

The experimental evidence supports all three hypotheses addressed in this chapter:

H1 is supported by EMDE's state-of-the-art MRR on five of six session-based datasets and the 23.6% performance drop when geometric structure is removed from embeddings.

H2 is supported by consistent improvements from multimodal concatenation: 7.7% Hit Rate improvement on RETAIL, 27% Recall@1 improvement on MovieLens-20M, with no learned fusion architecture.

H3 is supported by three-layer feed-forward networks matching or exceeding recurrent, attention-based, and graph neural network architectures, while training in under one hour versus over one week for GNN baselines.

The analysis of temporal graph models explains why density estimation succeeds. TGN and DyRep saturate on popular nodes, unable to discriminate among plausible candidates. Under hard evaluation with negatives sampled from popular nodes, TGN collapses to 0.010 MRR while EMDE maintains 0.468. The 46-fold gap demonstrates that density-based representations capture fundamentally different structure than local message-passing approaches.

EMDE has limitations worth acknowledging. It treats histories as weighted sets with temporal decay rather than as ordered sequences, avoiding the modeling of specific item-to-item transitions. For domains where the precise order of interactions carries genuine meaning beyond recency, such as educational progressions or narrative content, this may hurt performance.

Despite these limitations, EMDE demonstrates that carefully designed input representations can substitute for architectural complexity. The sketch captures structure that would otherwise require the network to learn: which regions of item space a user occupies, how those regions relate to each other, how multiple modalities combine. With this structure explicit in the input, a simple feed-forward network suffices.

The next chapter builds on these foundations to create BaseModel, a foundation model for behavioral data that leverages self-supervised pretraining for cross-task transfer.

6 Learning Behavioral Dynamics: A Foundation Model Approach

This chapter is based on the following publication:

Michał Daniluk, Mikołaj Spytek, Dariusz Matlak, Jacek Dąbrowski. "BaseModel: A Foundation Model for Behavioral Data." *Under review at ACM RecSys 2026*.

My contributions: I designed the overall framework architecture, developed the self-supervised training methodology, conducted the experimental evaluation across all benchmarks, and led the production deployment. The work also resulted in a commercial product available at `https://basemodel.ai/`.

This chapter addresses three hypotheses introduced in Chapter 1:

Hypothesis H2: Density sketching enables effective multimodal integration through simple concatenation. Building on the EMDE results from Chapter 5, this chapter extends multimodal evaluation to relational prediction tasks, ablating the contribution of metadata, text, and visual features across multiple databases.

Hypothesis H4: Self-supervised pretraining on behavioral dynamics enables transfer across task types. If behavioral data exhibits regularities in how preference distributions evolve over time, then a model trained to predict future distributions from past distributions should learn transferable representations. This is evaluated through learning curve analysis, comparing models initialized from the pretrained backbone against identical architectures trained from scratch, and by measuring performance across recommendation, classification, and regression tasks using a single pretrained backbone.

Hypothesis H5: Density-based representations provide disproportionate advantages in cold-start scenarios. Content features can position new items in embedding space immediately, but collaborative filtering methods cannot leverage items without interaction history. This is evaluated on datasets with documented cold-start rates ranging from 18% to 71% of test items unseen during training, comparing full multimodal profiles against interaction-only baselines.

The previous chapters established two foundational components: Cleora for extracting entity embeddings from behavioral hypergraphs, and EMDE for aggregating these embeddings into fixed-size density sketches that preserve distributional structure. Together, they solve the representation problem that has plagued behavioral modeling: how to convert variable-length interaction histories into fixed-size inputs suitable for neural networks, without collapsing multimodal preferences into meaningless centroids.

Yet representation alone is insufficient. A user's behavioral profile at time t tells us where they have been in preference space, but recommendation requires predicting where they will go. Preferences evolve. A user browsing baby products today will need toddler supplies in two years. Someone who just purchased a camera may soon want lenses and accessories. The trajectory through preference space, not merely the current position, determines future behavior.

This chapter introduces BaseModel, a framework that learns these trajectories through self-supervised training. The central insight is simple but consequential: if we can represent a user's past behavior as a density over item space, we can also represent their future behavior as a density. The learning problem becomes predicting one density from another. This framing treats preference evolution as distributional shift rather than as a sequence of discrete item transitions, which fundamentally changes what the model must learn.

The approach yields what I call a *foundation model* for behavioral data. I use this term deliberately but precisely: BaseModel employs self-supervised pretraining on unlabeled behavioral data, learns a reusable backbone representation, and transfers to multiple downstream tasks with minimal task-specific engineering. I do not claim capabilities analogous to large language models or universal cross-domain generalization. The contribution is demonstrating that behavioral data can support a foundation model paradigm at all, with the associated benefits of reduced task-specific engineering and improved sample efficiency.

The experimental results that follow reveal a striking pattern. On sequential recommendation benchmarks, BaseModel improves over state-of-the-art methods by 54–138% on sparse Amazon datasets, with the largest gains precisely where interaction histories are short and item distributions are long-tailed. Against Meta's HSTU, a trillion-parameter architecture representing the current industrial frontier for sequential recommendation, BaseModel achieves over 100% higher hit rate on Amazon Books despite HSTU's architectural innovations specifically designed for behavioral sequences. On RelBench relational prediction tasks spanning recommendation, classification, and regression, BaseModel matches or exceeds specialized graph neural networks on 10 of 12 tasks using a single pretrained backbone. Production deployment at a major European retailer achieved 21% higher email engagement over an incumbent system built on hand-engineered features.

These results suggest something beyond incremental improvement: they indicate that the sequential framing itself may be mismatched to behavioral data. The chapter develops this argument through theory, methodology, and extensive empirical validation.

6.1 The Limits of Sequential Modeling

Before presenting BaseModel, it is worth examining why an alternative to sequential modeling is needed at all. The dominant paradigm treats user behavior as a sequence of tokens and imports machinery from natural language processing: recurrent networks, attention mechanisms, autoregressive generation. This approach has achieved impressive results and powers recommendation at companies like Meta, Google, and Amazon. Yet closer examination reveals fundamental tensions between the sequential assumption and the actual structure of behavioral data.

6.1.1 The Language Analogy and Its Failures

The analogy between user behavior and language is seductive. Both involve sequences of discrete symbols. Both exhibit patterns that can be learned from large corpora. If predicting the next word works brilliantly for language, perhaps predicting the next item will work for behavior.

The analogy has merit for certain domains. When users progress through educational content, they must learn prerequisites before advanced material. When viewers watch television series, they experience episodes in intended order. In these cases, the sequence carries genuine semantic information that models should capture.

But e-commerce, the focus of this thesis, sits in a fundamentally different regime. Consider a user who adds shampoo, a novel, and a phone charger to their shopping cart. The order typically reflects website layout, search ranking, or simple chance. The user did not intend a meaningful progression from personal care to literature to electronics. Purchase timestamps often reflect logistics, promotions, or inventory availability rather than preference evolution. Users batch unrelated items into single sessions. Ratings are entered weeks after consumption.

In these common scenarios, knowing *which* items a user interacted with matters far more than knowing *when* they interacted with them. Yet sequential models treat all orderings as informative, learning patterns from what may be noise.

6.1.2 Evidence from Industrial Scale

The most compelling evidence comes from Meta's own evaluation of HSTU, their trillion-parameter sequential architecture [88]. HSTU represents perhaps the largest industrial investment in sequential recommendation to date, with innovations specifically designed to make Transformers work for behavioral data: pointwise aggregated attention that preserves intensity information, stochastic length training that handles variable sequence lengths, and relative attention biases incorporating temporal information.

Yet HSTU's ablation studies reveal a surprising finding. Their stochastic length technique, which randomly truncates sequences during training, removes over 80% of tokens while maintaining model quality within 0.2% normalized entropy. If four-fifths of a user's interaction history can be discarded without meaningful degradation, the sequential structure that HSTU models so carefully may contain far less information than the architecture assumes.

This observation connects to findings from my earlier work on attention in language models [14]. When investigating memory-augmented neural language models with access to extended context windows, I found that models predominantly attended to only the five most recent positions, regardless of how much context was available. A simple concatenation of recent outputs matched sophisticated attention mechanisms. If attention struggles to leverage extended context even in language, where sequential structure genuinely matters, the problem for behavioral data may be more fundamental.

6.1.3 A Different Perspective

These observations motivate a different framing. Rather than asking "what sequence of items did this user interact with," I ask "what regions of item space has this user occupied, and how are those regions shifting over time?"

This distributional perspective treats user behavior as samples from an evolving preference distribution. A user who purchased running shoes, fitness trackers, and protein supplements has not traced a trajectory through product space; they have indicated which regions match their preferences. The meaningful signal is not the path but the density: which areas of the space have accumulated behavioral mass, and how that mass is migrating.

The distinction is not between "order matters" and "order is ignored." Temporal information enters the distributional model through recency weighting: recent interactions contribute more heavily than older ones. What the distributional view avoids is modeling specific item-to-item transitions. It does not try to learn that viewing running shoes predicts viewing running socks. Instead, it learns that a user whose recent behavioral density concentrates in the athletic footwear region is likely to remain interested in that region.

This framing leads directly to BaseModel's architecture: represent behavioral histories as density sketches, and learn how those densities evolve over time.

6.2 Problem Formulation

This section formalizes the learning problem that BaseModel addresses. Table 6.1 summarizes the notation introduced in this chapter, building on the core notation from Table 2.1 and the sketch notation from Table 5.1.

This formulation clarifies how the approach differs from both next-item prediction and sequence reconstruction, and provides the theoretical foundation for the methodology that follows.

6.2.1 Behavioral Data as Density Estimation

Consider a user u with interaction history $\mathcal{H}_u = \{(i_1, t_1), (i_2, t_2), \ldots, (i_n, t_n)\}$, where each pair (i_k, t_k) denotes an item i_k interacted with at time t_k. Each item i has an associated embedding $\mathbf{e}_i \in \mathbb{R}^d$ computed through methods like Cleora (Chapter 4).

The standard sequential formulation treats $\mathcal{H}_u$ as an ordered sequence and learns transition dynamics:

$$P(i_{n+1} | i_1, i_2, \ldots, i_n) \tag{6.1}$$

This requires the model to learn item-specific transition probabilities, which becomes problematic for long-tailed catalogs where most items have few observations.

I propose an alternative formulation. Rather than predicting the next item, the model predicts the next *behavioral distribution*. Define the empirical density of a user's interactions as a distribution over the embedding manifold:

$$\rho_u(\mathbf{x}) = \sum_{(i_k, t_k) \in \mathcal{H}_u} w(\Delta t_k) \cdot \delta(\mathbf{x} - \mathbf{e}_{i_k}) \tag{6.2}$$

Table 6.1: BaseModel notation.

Symbol	Description
Behavioral Distributions	
$\rho_u(\mathbf{x})$	Behavioral density of user u over embedding space
$\rho_u^{\text{past}}, \rho_u^{\text{future}}$	Densities before/after temporal split
τ	Temporal split point
Sketch Representations	
$\mathbf{S}_{\text{past}}, \mathbf{S}_{\text{future}}$	Sketches of past and future interactions
$\mathbf{S}_u^{(m)}$	User u's sketch for modality m
$\mathbf{S}_u$	Complete behavioral profile (concatenated sketches)
Foundation Model	
θ	Model parameters
f_θ	Foundation model with parameters θ
$\hat{\mathbf{S}}_{\text{future}}$	Predicted future sketch
$\mathcal{L}(\theta)$	Training loss
Recommendation Scoring	
$c^{(j,m)}(i)$	Bucket code for item i, partitioning j, modality m
$\text{score}(i)$	Recommendation score for item i

Note: u generalizes to any source entity in non-user-item contexts (see Section 6.8).

where $w(\Delta t_k)$ is a temporal weighting function (typically exponential decay, with Δt_k measuring recency) and δ is the Dirac delta. This density captures where in embedding space the user has concentrated their behavior, with more recent interactions weighted more heavily.

The learning problem becomes: given the density ρ_u^{past} computed from interactions before some time τ, predict the density ρ_u^{future} of interactions after τ:

$$f_\theta : \rho_u^{\text{past}} \to \rho_u^{\text{future}} \tag{6.3}$$

This formulation has several advantages over next-item prediction. First, it sidesteps the vocabulary problem: the model predicts distributions over a continuous manifold rather than probabilities over a discrete item catalog. Second, it naturally handles the multimodal nature of preferences: a user interested in both electronics and books has a bimodal density, which the formulation preserves rather than forcing a unimodal prediction. Third, it degrades gracefully with sparse data: even users with few interactions produce meaningful densities that can be compared and predicted.

6.2.2 From Continuous Densities to Discrete Sketches

The formulation above involves continuous densities, which are computationally intractable for high-dimensional embedding spaces. EMDE (Chapter 5) provides the bridge to practical computation.

Recall that EMDE approximates densities through locality-sensitive hashing with density-dependent thresholds. For N independent random partitionings, each with 2^K buckets, an item embedding $\mathbf{e}_i$ maps

to bucket indices $c^{(1)}(\mathbf{e}_i), \ldots, c^{(N)}(\mathbf{e}_i)$. A user's behavioral density is approximated by the sketch:

$$\mathbf{S}_u[j, b] = \sum_{(i_k, t_k) \in \mathcal{H}_u} w(\Delta t_k) \cdot \mathbf{1}[c^{(j)}(\mathbf{e}_{i_k}) = b] \tag{6.4}$$

which counts (with temporal weighting) how many of the user's interactions fall into bucket b of partitioning j.

The learning problem in sketch space becomes:

$$f_\theta : \mathbf{S}_{\text{past}} \to \mathbf{S}_{\text{future}} \tag{6.5}$$

where both input and output are fixed-size tensors of dimension $N \times 2^K$, regardless of the number of items in the user's history or the size of the item catalog.

6.2.3 Why Density-to-Density Prediction Should Work

The formulation raises a natural question: why should predicting future densities from past densities be easier than predicting future items from past items?

The answer lies in the smoothness of behavioral change. Users do not jump randomly between unrelated preferences; they drift. A user interested in photography today is likely to remain interested in photography tomorrow, perhaps expanding to related areas like photo editing software or travel. This drift manifests as gradual migration of probability mass across the embedding manifold, a smooth transformation that neural networks can learn efficiently.

In contrast, next-item prediction requires modeling the full joint distribution over item sequences. Even if underlying preferences change smoothly, the specific items a user encounters depend on catalog availability, promotions, interface design, and countless other factors that introduce high-frequency noise. The density formulation integrates over this noise: it does not matter *which* photography product the user buys next, only that their behavioral mass remains in the photography region.

This perspective explains why BaseModel achieves its largest improvements on sparse datasets (Section 6.7). When interaction histories are short, sequence models cannot reliably estimate item-to-item transition probabilities. But even a few interactions suffice to estimate which regions of the manifold a user prefers, and the patterns of how regional preferences evolve can be learned from aggregate behavior across many users.

The advantage over existing self-supervised alternatives is equally important. Contrastive methods such as SGL [80] and SimGCL [86] generate multiple views of the same user's data through graph augmentation, then train the model to produce stable representations under perturbation. The objective encourages compactness and invariance, not temporal prediction: a model that assigned nearly identical representations to all users after aggressive augmentation would satisfy the contrastive objective while learning nothing about preference dynamics. More fundamentally, contrastive training tells the model which histories are "similar," not where behavior is going. Masked-item prediction methods such as BERT4Rec ask the model to recover randomly removed items, which is structurally equivalent to next-item prediction at the item level and inherits the same vocabulary bottleneck: gradients are concentrated on a tiny subset of catalog items at each step, making it hard to learn from rare items or to generalize to new

ones. Distribution-to-distribution prediction avoids both failure modes. The cross-entropy objective over B buckets per partitioning provides dense gradients across the full behavioral distribution at every training step. The model must simultaneously predict which regions of the embedding manifold will become active, forcing it to learn the global structure of how preferences migrate rather than memorizing item co-occurrence statistics. This is not merely an engineering convenience; it is the property that makes the learned representations transferable to classification and regression tasks that share no items with the pretraining data.

6.3 Universal Behavioral Profiles

The first component of BaseModel is a representation that I call the *universal behavioral profile*. The name reflects an ambition: a single fixed-size vector that captures everything relevant about an entity's behavioral history, regardless of how many interactions they have, what modalities are available, or what downstream task we eventually want to solve.

The construction builds directly on Cleora and EMDE, combining them into a unified pipeline that produces rich, multimodal representations suitable for foundation model training.

6.3.1 Multimodal Embedding Sources

Real-world behavioral data involves heterogeneous signals. A product is not merely an identifier in a transaction log; it has a title, description, images, category hierarchy, and price. Each signal captures something different about what the product is and who might want it.

These signals fall into three categories:

Collaborative embeddings capture patterns of co-occurrence. Cleora processes the behavioral hypergraph, where each transaction links a user to products, brands, categories, and contextual attributes. After iterative averaging, entities that frequently co-occur receive similar embeddings. A product purchased mainly by young urban professionals ends up near other products popular with the same demographic, even if no explicit demographic features were provided.

Content embeddings capture what items are about. Product titles and descriptions are embedded using sentence transformers (all-MiniLM-L6-v2 throughout). Product images are embedded using vision transformers (ViT-base-patch16-224). These embeddings position items according to semantic and visual similarity, complementing the collaborative signal.

Structured features capture categorical and numerical attributes. Categories, brands, price ranges, and other metadata are encoded through embedding tables or numerical transformations. These provide direct signal about item properties that may not be inferable from collaborative or content patterns alone.

Not every entity has every modality. New products lack interaction history. Some items have images while others do not. The pipeline handles missing modalities gracefully by using what is available, a property that proves essential for cold-start scenarios.

6.3.2 From Embeddings to Sketches

Each embedding type lives in its own geometric space. Collaborative embeddings cluster according to co-purchase patterns. Text embeddings cluster according to semantic similarity. Image embeddings cluster according to visual appearance. These spaces are not aligned: two products near each other in text space may be distant in collaborative space.

EMDE is applied independently to each embedding type. For each modality m, density-dependent hyperplane partitions are fitted on the full set of entity embeddings of that type, producing bucket assignments that respect the geometry of that particular space. The partitions are computed once during preprocessing and remain fixed, allowing item codes to be precomputed and cached.

To construct a user's profile for modality m, the embeddings of all items in their history are retrieved, the precomputed bucket codes are looked up, and the counts are aggregated:

$$\mathbf{S}_u^{(m)}[j, b] = \sum_{(i_k, t_k) \in \mathcal{H}_u} w(\Delta t_k) \cdot \mathbf{1}[c^{(j,m)}(\mathbf{e}_{i_k}^{(m)}) = b] \tag{6.6}$$

where $\mathbf{e}_{i_k}^{(m)}$ is item i_k's embedding for modality m, and $c^{(j,m)}$ is the bucket assignment function for partitioning j of modality m.

6.3.3 Training Example Construction

The way training examples are constructed from raw interaction data significantly affects what models learn. For the foundation model pretraining stage, Consecutive Event Splits are always used to maximize training signal, as described in Section 6.4. For downstream task training, the choice of split strategy depends on task characteristics. Two strategies are employed, selected based on what the downstream task requires.

Consecutive Event Splits (CES). For tasks requiring dense temporal supervision, particularly recommendation, the split point is placed between every pair of consecutive events in a user's history. A user with chronologically ordered events $(e_1, e_2, \ldots, e_T)$ contributes $T - 1$ training examples:

Split Point	Past Sketch	Future Sketch
After e_1	$\{e_1\}$	$\{e_2, e_3, \ldots, e_T\}$
After e_2	$\{e_1, e_2\}$	$\{e_3, e_4, \ldots, e_T\}$
$\vdots$	$\vdots$	$\vdots$
After e_{T-1}	$\{e_1, e_2, \ldots, e_{T-1}\}$	$\{e_T\}$

This strategy offers three advantages. First, it maximizes the number of training examples: a user with 100 events contributes 99 examples rather than just one. Second, every example contains a non-empty future segment, ensuring the model always observes what comes next. Third, the model learns from the full spectrum of history lengths, ranging from predictions based on a single event to predictions based on a near-complete history.

Random Temporal Splits (RTS). For classification and regression tasks, where a single prediction per entity suffices, a single random timestamp τ is drawn uniformly within each user's activity window $[t_1, t_T]$. Events before τ define the past sketch; events after τ define the future sketch:

$$\mathbf{S}_{\text{past}} = \text{Sketch}\big(\{e_k : t_k < \tau\}\big) \tag{6.7}$$

$$\mathbf{S}_{\text{future}} = \text{Sketch}\big(\{e_k : t_k \geq \tau\}\big) \tag{6.8}$$

A critical difference from CES is that the future segment may be empty. If τ falls after the user's last event, the future sketch contains no items. This property is essential for tasks like churn prediction, where user inactivity is precisely the outcome of interest.

Both strategies respect dataset-level temporal boundaries. Training examples never incorporate information from validation or test periods.

Table 6.2 summarizes when each strategy is appropriate.

Table 6.2: Summary of training example construction strategies.

Characteristic	CES	RTS
Examples per user	$T - 1$	1
Empty future allowed	No	Yes
Primary use case	Recommendation	Classification, Regression
Temporal variability	Deterministic	Stochastic

6.3.4 Multimodal Concatenation

The complete behavioral profile concatenates sketches across all available modalities:

$$\mathbf{S}_u = [\mathbf{S}_u^{(1)}; \mathbf{S}_u^{(2)}; \ldots; \mathbf{S}_u^{(M)}] \tag{6.9}$$

This simple concatenation approach may seem naive. Complex fusion architectures have been proposed for combining heterogeneous features: attention mechanisms that learn to weight modalities, gating networks that select which signals to trust, cross-modal transformers that model interactions between modalities. I use none of these.

Why does concatenation suffice? The key insight is that EMDE sketches are already high-dimensional histograms that preserve fine-grained information. Unlike averaged embeddings, which collapse distributional structure into a single point, sketches maintain the full shape of where a user has been in each space. The downstream network has access to all this structure and can learn which modalities matter for which predictions, which combinations of bucket activations are informative, and how to weight signals appropriately. The fusion happens implicitly through learning rather than through architectural constraints.

There is also a practical benefit. Adding a new modality requires only fitting partitions on the new embedding type and concatenating the resulting sketch. No architectural changes, no retraining of fusion components. This modularity simplifies deployment and experimentation.

6.3.5 Properties of the Representation

The universal behavioral profile has several properties that matter for building a foundation model.

Fixed dimensionality. Profile size depends only on sketch parameters (N partitions $\times$ 2^K buckets $\times$ M modalities), not on history length or catalog size. A new user with three interactions and a power user with three thousand produce profiles of identical dimension. This simplifies batching, architecture design, and deployment.

Additive construction. Sketches inherit additivity from EMDE. When a user interacts with a new item, the item's precomputed bucket codes are looked up and added to the existing sketch. Real-time systems can maintain running profiles without reprocessing entire histories. This matters for production deployment where latency budgets are tight.

Cold-start support. Any item with content features receives meaningful bucket codes immediately. A product added to the catalog today, with no purchase history, still has a text description and perhaps an image. These produce embeddings that map to buckets, positioning the item in the relevant regions of content space. This is not an approximation or fallback mechanism; it is the same inference procedure used for established items.

Graceful degradation. When modalities are missing, the corresponding sketch positions are zero. The network learns to rely on available signals. This flexibility proves essential for handling the heterogeneous data quality encountered in real-world systems.

6.4 The Foundation Model

The previous section described how to represent an entity's behavioral history as a fixed-size density sketch. This section describes how BaseModel learns temporal dynamics over that representation: predicting future behavioral distributions from past ones through self-supervised training.

6.4.1 Self-Supervised Training Objective

The training procedure requires no task-specific labels. The supervision signal comes entirely from the temporal structure of behavioral data itself.

For each entity, the interaction history is split at a temporal boundary. Events before the boundary are aggregated into $\mathbf{S}_{\text{past}}$; events after form $\mathbf{S}_{\text{future}}$. The model learns to predict the future sketch from the past:

$$f_\theta : \mathbf{S}_{\text{past}} \to \mathbf{S}_{\text{future}} \tag{6.10}$$

The foundation model uses Consecutive Event Splits (CES) to maximize the training signal available from each user's history. A user with events $(e_1, e_2, \ldots, e_T)$ contributes up to $T - 1$ training examples: predicting behavior after e_1 from $\{e_1\}$, predicting behavior after e_2 from $\{e_1, e_2\}$, and so on. This dense supervision exposes the model to behavioral distributions at every stage of history development, from sparse early interactions to mature profiles with hundreds of events. The model learns how probability mass migrates across the embedding manifold as users accumulate experience, capturing patterns that transfer across the population.

Training examples respect dataset-level temporal boundaries and never incorporate information from validation or test periods. The split strategies used for downstream task training, described in Section 6.3.3, may differ from the foundation stage depending on task requirements.

6.4.2 Network Architecture

The architecture is deliberately simple. The concatenated input sketch is first projected into a hidden representation through a factorized linear layer (Section 6.4.3). This representation then passes through a stack of residual inverted bottleneck blocks. Each block consists of:

1. Layer normalization
2. Expansion to a wider hidden dimension
3. GELU nonlinearity
4. Projection back to the original hidden size
5. Skip connection adding the block's input to its output

All linear projections in the architecture, including those within the bottleneck blocks and the final output mapping to the predicted future sketch, use the factorized variant described in Section 6.4.3.

There are no attention mechanisms. No recurrence. No sequence operations of any kind. The network is a feed-forward transformation from input sketch to output sketch.

This simplicity is intentional. The representational complexity resides in the behavioral profiles themselves, which already encode high-resolution density information across multiple modalities. The sketches preserve which specific items a user interacted with, not just aggregate statistics. They maintain the geometry of each embedding space through the bucket structure. They integrate collaborative and content signals through concatenation.

Given this rich input, the network's job is comparatively modest: learn how probability mass tends to shift between sketch regions when moving from past to future. This is a smooth, structured transformation that a feed-forward network can capture well. The alternative, building a complex architecture to compensate for impoverished inputs, would be solving the wrong problem. That said, the input is a concatenation of modality-specific sketches, and there is structure in how these components interact that purely additive projections cannot exploit. The factorized linear layer, described next, addresses this.

6.4.3 Factorized Linear Layer

The concatenated input sketch contains density bins from multiple modalities and multiple random partitions. A standard linear layer computes additive combinations of these features: each output is a weighted sum of all inputs plus a bias. This first-order computation cannot model how the joint presence of density in two different sketch regions should affect the output beyond what their individual contributions predict. For a multimodal sketch, the fact that a user has high density in a particular text bucket *and* a particular collaborative bucket simultaneously may be more informative than either signal alone, but a linear layer has no way to express this.

The factorized linear layer (FMLinear) addresses this by augmenting each linear projection with a low-rank quadratic interaction term. Given input $\mathbf{x} \in \mathbb{R}^n$, FMLinear produces output $\mathbf{y} \in \mathbb{R}^d$ where each output dimension j is computed as:

$$y_j = \frac{1}{\sqrt{2}} \left(\sum_{m=1}^{k} \left(\mathbf{v}_{j,m}^\top \mathbf{x} \right)^2 + \mathbf{w}_j^\top \mathbf{x} + b_j \right) \tag{6.11}$$

where $\mathbf{v}_{j,m} \in \mathbb{R}^n$ are k factorization vectors for output dimension j, $\mathbf{w}_j \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$ are standard linear parameters, and the $1/\sqrt{2}$ factor prevents the summed terms from inflating activation variance.

The quadratic term captures pairwise feature interactions through low-rank factorization. Expanding the squared inner product reveals the connection to factorization machines [53]:

$$\sum_{m=1}^{k} \left(\mathbf{v}_{j,m}^\top \mathbf{x} \right)^2 = \sum_{m=1}^{k} \sum_{i=1}^{n} \sum_{l=1}^{n} x_i \, x_l \, v_{j,m,i} \, v_{j,m,l} = \sum_{i=1}^{n} \sum_{l=1}^{n} x_i \, x_l \left\langle \mathbf{v}_i^{(j)}, \mathbf{v}_l^{(j)} \right\rangle \tag{6.12}$$

where $\mathbf{v}_i^{(j)} = (v_{j,1,i}, \ldots, v_{j,k,i})^\top \in \mathbb{R}^k$ collects the factorization weights for input feature i across all k factors at output j. Each pairwise interaction $x_i \, x_l$ is weighted by the inner product of the corresponding k-dimensional factor vectors, enabling the layer to model $O(n^2)$ interactions using only $O(nk)$ parameters per output dimension. Unlike classical factorization machines, which exclude self-interactions ($i = l$), FMLinear includes them for computational simplicity; any systematic effect is absorbed during training.

In practice, all factorization vectors are stored as a single matrix $V \in \mathbb{R}^{n \times kd}$. The quadratic term is computed in three steps: (1) multiply $\mathbf{x}V$ to obtain a vector of length kd, (2) reshape into a $d \times k$ matrix, and (3) square element-wise and sum along the k dimension to produce $\mathbf{q} \in \mathbb{R}^d$. This maps directly onto GPU tensor operations: one matrix multiply, one reshape, one element-wise square, and one reduction. The total parameter count is $n(k+1)d + d$, compared to $nd + d$ for a standard linear layer. With $k = 8$ (the default used in all experiments), FMLinear has roughly nine times as many parameters per layer. For a projection from a 30,000-dimensional sketch to 3,000 hidden units, this means 810M versus 90M parameters. The parameter count is substantial, but the computational overhead is not: FMLinear adds only one extra matrix multiplication per layer, which is small relative to the cost of data loading, embedding computation, and sketch construction in the full pipeline.

The factorization vectors are initialized with Xavier-uniform scaling adapted to the expanded fan-out: $V_{ij} \sim \mathcal{U}(-\gamma, \gamma)$ where $\gamma = \sqrt{6/(n + kd)}$. This keeps the variance of the quadratic and linear terms comparable at initialization, so neither dominates the gradient early in training.

FMLinear replaces every standard linear projection in the architecture: the input projection, both projections within each inverted bottleneck block, and the output projection. No other architectural changes are needed; the network remains purely feed-forward.

6.4.4 Loss Function

The model is trained using a distributional matching objective. Each sketch consists of N partitionings, where each partitioning is a histogram over B buckets. Each partitioning is normalized independently to obtain a probability distribution, and the cross-entropy is computed between the predicted distribution

and the ground-truth distribution:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{j=1}^{N} \mathrm{CE}\left(\hat{\mathbf{S}}_{\text{future}}^{(j)}, f_\theta(\mathbf{S}_{\text{past}})^{(j)}\right) \tag{6.13}$$

where $\hat{\mathbf{S}}_{\text{future}}^{(j)}$ is the L_1-normalized ground-truth histogram at partitioning j, and $f_\theta(\mathbf{S}_{\text{past}})^{(j)}$ is the corresponding softmax output of the network.

This objective encourages the model to match the shape of the future distribution, not just its mode. If a user's future behavior will spread across three product categories, the predicted sketch should allocate probability mass to all three, in roughly the right proportions. A model that concentrates all mass on the single most likely category would incur high loss even if that category is correct, because it fails to capture the distributional structure.

For multimodal profiles, losses across all modalities are averaged uniformly. The network learns implicitly how to weight different modalities based on their predictive value.

6.4.5 What the Foundation Model Learns

The pretraining ablation in Section 6.9.3 provides the clearest evidence of what the foundation stage actually captures. On Amazon Beauty, the pretrained model reaches after 106 iterations a performance level that the scratch model does not achieve until iteration 318. On rel-hm churn, a binary classification task with a structurally different objective from the pretraining, the pretrained model starts and remains ahead throughout. This transfer to a different task type suggests the backbone learns something about behavioral dynamics in general, not just the distribution-prediction objective it was trained on.

The most straightforward interpretation is that the self-supervised objective forces the model to learn which sketch regions tend to co-activate, how regional density shifts over time within a user's profile, and how these patterns generalize across users. A model that merely memorized per-user statistics would not transfer to the churn task, because the churn label was never observed during pretraining. The transfer happens because the temporal dynamics of behavioral density are predictive of future engagement broadly, whether the downstream question is "what will this user buy" or "will this user stop using the platform."

The model also learns relationships between modalities. On datasets with both collaborative and content features, the foundation model discovers how these signals interact. A user whose past sketch shows strong collaborative signal but weak content signal differs systematically from a user with the reverse pattern, and the foundation model learns these distinctions and how they relate to future behavior.

Importantly, none of this requires task-specific labels. The supervision comes entirely from the temporal structure of behavioral data itself. This is the foundation model paradigm applied to behavioral domains: learn general patterns from abundant unlabeled data, then adapt to specific tasks with minimal additional supervision.

6.5 Downstream Task Adaptation

The foundation model learns general patterns of behavioral evolution. This section describes how the same pretrained backbone supports diverse downstream tasks by attaching lightweight task-specific heads.

The sketching pipeline and foundation architecture remain unchanged; only the prediction head and loss function differ.

6.5.1 Recommendation

Recommendation is framed as direct decoding from the predicted future sketch. At inference time, the pretrained backbone receives the entity's current behavioral profile and outputs a predicted future sketch: a multimodal probability distribution over the item manifold representing where the user's behavior is likely to concentrate.

To score any candidate item, the bucket codes are computed under the same random hyperplanes used during sketching, the corresponding buckets in the predicted sketch are looked up, and the probabilities are aggregated via geometric mean:

$$\text{score}(i) = \left(\prod_{m=1}^{M} \prod_{j=1}^{N} \hat{\mathbf{S}}^{(m)}[j, c^{(j,m)}(i)] \right)^{1/(NM)} \tag{6.14}$$

where $\hat{\mathbf{S}}^{(m)}[j, c^{(j,m)}(i)]$ is the predicted probability at the bucket containing item i for modality m and partitioning j.

The geometric mean requires consistent evidence across partitions: an item scores well only if it has reasonable probability in most partitions, preventing single high values from dominating. This choice is theoretically grounded in the literature on combining independent probability estimates and empirically outperforms arithmetic mean and minimum aggregation.

This lookup-based decoding has several attractive properties:

- It requires no learned item embeddings at inference time, only precomputed bucket codes.
- It avoids similarity search over large item catalogs, since scoring reduces to table lookups.
- It sidesteps beam search or other expensive decoding procedures that generative retrieval methods require.
- It handles cold-start items seamlessly: any item with content features receives bucket codes immediately.

The computational cost scales with the number of items to score, not with the complexity of the model. Scoring a single item requires $N \times M$ lookups and one geometric mean computation.

6.5.2 Classification

Binary and multi-class prediction tasks attach simple classifiers to the behavioral representation. For binary tasks like churn prediction or fraud detection, a linear layer maps the profile to a single logit, followed by sigmoid. For multi-class tasks, the linear layer produces one logit per class, followed by softmax.

Training proceeds by fine-tuning the entire model end-to-end with cross-entropy loss on labeled examples. Because the behavioral profile already captures long-term patterns through foundation pretraining, these classifiers perform well even with limited labeled data.

6.5.3 Regression

Continuous prediction tasks use the same architecture pattern with a different output layer. Customer lifetime value forecasting, expected future spend, and time-to-next-purchase prediction all attach regression heads to the behavioral profile.

The simplest configuration uses a single linear layer mapping the profile to a scalar output, trained with mean squared error or mean absolute error. The pretrained backbone provides a strong prior that improves both convergence speed and final performance, as the foundation model has already learned how behavioral patterns relate to future outcomes.

6.5.4 Unified Training Pipeline

Across all task types, the training pipeline follows the same pattern. The universal behavioral profile provides a fixed-size input representation regardless of history length or available modalities. The pretrained foundation backbone transforms this profile into a hidden representation that captures temporal dynamics. A lightweight task-specific head produces the final prediction.

This modularity simplifies deployment. A single behavioral profile can feed multiple downstream models simultaneously: one for product recommendations, another for churn risk scoring, a third for lifetime value estimation. The expensive computation (constructing profiles, running the backbone) happens once; only the cheap final layer differs per task.

6.6 Experimental Setup

The experiments are designed to answer three questions. First, does BaseModel improve over state-of-the-art sequential recommenders, particularly in sparse and cold-start settings? Second, does the approach generalize beyond user-item sequences to multi-table relational prediction? Third, what components of the behavioral profile drive the performance gains?

6.6.1 Evaluation Protocol

For all baselines, results are reported exactly as published in their respective papers. These methods are not reimplemented or retuned; all comparisons reflect officially reported numbers under each method's original evaluation protocol. This approach ensures fairness but means that some implementation details differ across comparisons. Where protocols differ, BaseModel is run under both and each comparison is reported against the appropriate baseline.

For BaseModel, variation across five training runs with different random seeds is reported. Results appear as mean $\pm$ standard deviation. This quantifies the stability of the method and provides context for interpreting differences.

6.6.2 Datasets

The evaluation spans datasets covering different domains, scales, and task types.

Amazon Product Reviews. Three category subsets from the Amazon review corpus: Beauty, Sports and Outdoors, and Toys and Games. These are sparse datasets with short user histories (median sequence length around 6 interactions) and moderately sized item catalogs (12,000 to 18,000 items). Only collaborative embeddings from Cleora are used, following the setup of prior work.

MovieLens. Two versions of the MovieLens rating dataset: ML-1M with approximately one million ratings from 6,000 users, and ML-20M with twenty million ratings from 138,000 users. These datasets have longer histories per user (minimum 20 ratings by construction) and represent a dense interaction setting.

Amazon Books. A larger Amazon subset with over 600,000 users. This dataset is evaluated under two protocols: leave-one-out for comparison with HSTU, and temporal split for comparison with beeFormer. The temporal split holds out the last 20% of interactions chronologically.

RelBench. A benchmark for relational machine learning comprising four multi-table databases: rel-amazon (e-commerce), rel-avito (online advertising), rel-hm (fashion retail), and rel-stack (question-answering). Tasks include link prediction (recommendation), entity classification, and entity regression. These datasets include substantial cold-start exposure: 71% of test items in rel-avito and 32% in rel-amazon review were never seen during training.

Table 6.3 summarizes key statistics.

Table 6.3: Dataset statistics. Modalities: I = interactions (Cleora), M = metadata/text, V = visual.

Dataset	Users	Items	Interactions	Modalities
Amazon Beauty	22,363	12,101	198K	I
Amazon Sports	35,598	18,357	296K	I
Amazon Toys	19,412	11,924	167K	I
MovieLens-1M	6,040	3,706	1.0M	I
MovieLens-20M	138,493	27,278	20.0M	I
Amazon Books	634,964	63,305	8.3M	I+M
rel-amazon	—	—	15.0M	I+M
rel-avito	—	—	20.7M	I+M
rel-hm	—	—	16.7M	I+M+V
rel-stack	—	—	4.2M	I+M

6.6.3 Baselines

The comparison includes state-of-the-art methods spanning multiple paradigms:

Generative retrieval: TIGER [52] encodes items into hierarchical Semantic IDs via residual quantization, then uses a Transformer encoder-decoder with beam search.

Sequential Transformers: HSTU [88] represents Meta's industrial frontier with pointwise aggregated attention and stochastic length training. SASRec [33] and BERT4Rec [62] provide additional reference points.

Sentence transformer fine-tuning: beeFormer [73] fine-tunes sentence transformers on interaction data to align semantic embeddings with collaborative signals.

Graph neural networks: ContextGNN [87] and RelGNN [10] are designed for relational databases with complex multi-table schemas.

6.6.4 Implementation Details

All experiments use the RAdamScheduleFree optimizer with momentum coefficients $(0.9, 0.95)$ and weight decay 0.1. Batch size is 512 throughout. Learning rates and epoch counts vary by dataset, with early stopping on validation metrics determining the final checkpoint.

Sketch configurations scale with dataset complexity. Sketch depth ranges from 50 to 130 partitions; width ranges from 32 to 1024 buckets. Cleora uses 3 to 6 propagation iterations depending on graph density. The backbone hidden dimension ranges from 2048 to 5120.

Table 6.4 provides complete hyperparameter specifications for reproducibility.

Table 6.4: Hyperparameters across all experiments.

Dataset	Task	Cleora Iter.	Sketch N×B	Hidden	LR	Epochs
rel-avito	ad-visit	5	130×1024	3584	3e-4	1
rel-avito	visits	5	130×1024	3584	3e-3	1
rel-avito	clicks	5	130×1024	3584	1e-3	1
rel-hm	purchase	3	110×128	5120	3e-4	4
rel-hm	churn	3	110×128	5120	3e-4	1
rel-amazon	purchase	6	110×512	5120	3e-4	2
rel-amazon	churn	6	110×512	5120	1e-4	1
rel-stack	comment	5	70×32	2048	3e-4	1
Amazon Books	temporal	4	50×128	5120	3e-4	2
Amazon Books	LOO	5	90×512	4096	3e-4	3
ML-20M	LOO	5	90×512	4096	3e-4	1
ML-1M	LOO	5	90×512	4096	3e-4	3
Amazon Sports	LOO	3	100×128	5120	1e-4	3
Amazon Beauty	LOO	3	100×128	5120	1e-4	3
Amazon Toys	LOO	3	100×128	5120	1e-4	4

6.7 Sequential Recommendation Results

This section presents results on sequential recommendation benchmarks, comparing BaseModel against TIGER, HSTU, and beeFormer. The results reveal a consistent pattern: BaseModel achieves its largest improvements on sparse datasets where sequential models struggle to learn reliable item representations.

6.7.1 Comparison with TIGER

Table 6.5 presents results on three Amazon category datasets under the leave-one-out protocol.

BaseModel substantially outperforms all baselines, achieving 54–82% improvements over TIGER depending on dataset and metric. Both approaches model behavioral evolution, but they differ fundamentally in mechanism. TIGER uses a Transformer encoder-decoder with residual quantization to generate hierarchical Semantic IDs, requiring beam search at inference time. BaseModel predicts a continuous density sketch through a feed-forward network, then scores items through direct bucket lookups.

Table 6.5: Sequential recommendation on Amazon datasets (leave-one-out). Values are percentages. Relative improvement over TIGER shown in the final row.

(a) Beauty

Method	**Recall@5**	**NDCG@5**	**Recall@10**	**NDCG@10**
GRU4Rec	1.64	0.99	2.83	1.37
BERT4Rec	2.03	1.24	3.47	1.70
SASRec	3.87	2.49	6.05	3.18
S^3-Rec	3.87	2.44	6.47	3.27
TIGER	4.54	3.21	6.48	3.84
BaseModel	**8.15 $\pm$ 0.12**	**5.74 $\pm$ 0.10**	**11.47 $\pm$ 0.15**	**6.80 $\pm$ 0.11**
vs. TIGER	+79.5%	+78.8%	+77.0%	+77.1%

(b) Sports and Outdoors

Method	**Recall@5**	**NDCG@5**	**Recall@10**	**NDCG@10**
GRU4Rec	1.29	0.86	2.04	1.10
BERT4Rec	1.15	0.75	1.91	0.99
SASRec	2.33	1.54	3.50	1.92
S^3-Rec	2.51	1.61	3.85	2.04
TIGER	2.64	1.81	4.00	2.25
BaseModel	**4.78 $\pm$ 0.11**	**3.30 $\pm$ 0.09**	**6.69 $\pm$ 0.13**	**3.91 $\pm$ 0.10**
vs. TIGER	+81.1%	+82.3%	+67.3%	+73.8%

(c) Toys and Games

Method	**Recall@5**	**NDCG@5**	**Recall@10**	**NDCG@10**
GRU4Rec	0.97	0.59	1.76	0.84
BERT4Rec	1.16	0.71	2.03	0.99
SASRec	4.63	3.06	6.75	3.74
S^3-Rec	4.43	2.94	7.00	3.76
TIGER	5.21	3.71	7.12	4.32
BaseModel	**8.06 $\pm$ 0.11**	**5.71 $\pm$ 0.09**	**11.51 $\pm$ 0.15**	**6.82 $\pm$ 0.11**
vs. TIGER	+54.7%	+53.9%	+61.7%	+57.9%

The improvements are largest on Beauty and Sports, the sparsest of the three datasets. With median sequence lengths around 6 interactions, there is little sequential structure for sequence models to exploit. TIGER must learn item embeddings and transition patterns from these sparse observations. BaseModel sidesteps this problem: Cleora propagates information through co-occurrence structure, giving even rarely-seen items meaningful representations, and the density formulation allows learning from how regional preferences evolve rather than specific item-to-item transitions.

6.7.2 Comparison with HSTU

Table 6.6 presents results on MovieLens and Amazon Books under the leave-one-out protocol used by HSTU. This comparison pits BaseModel against Meta's state-of-the-art sequential architecture, which represents the current industrial frontier.

Table 6.6: Results on MovieLens and Amazon Books (leave-one-out). Values are percentages. Relative change over HSTU-large shown.

	Method	HR@10	HR@50	NDCG@10	NDCG@200
ML-1M	SASRec	28.53	54.74	16.03	24.98
	HSTU	30.97	57.54	17.20	26.06
	HSTU-large	32.94	59.35	18.93	27.71
	BaseModel	**34.04 ± 0.17**	**60.00 ± 0.13**	**19.98 ± 0.14**	**28.74 ± 0.10**
	vs. HSTU-large	+3.3%	+1.1%	+5.5%	+3.7%
ML-20M	SASRec	29.06	54.99	16.21	25.21
	HSTU	32.52	58.85	18.78	27.74
	HSTU-large	35.67	**61.49**	21.06	29.71
	BaseModel	**36.41 ± 0.20**	61.02 ± 0.16	**22.39 ± 0.18**	**30.76 ± 0.12**
	vs. HSTU-large	+2.1%	−0.8%	+6.3%	+3.5%
Books	SASRec	2.92	7.29	1.56	3.50
	HSTU	4.04	9.43	2.19	4.50
	HSTU-large	4.69	10.66	2.57	5.08
	BaseModel	**9.52 ± 0.26**	**15.65 ± 0.29**	**6.12 ± 0.19**	**8.39 ± 0.21**
	vs. HSTU-large	+103.0%	+46.8%	+138.1%	+65.2%

The results reveal a striking pattern that illuminates fundamental differences between sequential and distributional approaches.

On MovieLens, BaseModel achieves modest but mostly consistent improvements: 3.3% on HR@10 for ML-1M, 2.1% for ML-20M. The exception is HR@50 on ML-20M, where HSTU-large edges ahead by 0.8% (61.49 vs. 61.02). At very wide recall cutoffs, the sequential model's advantage in ranking borderline items appears to recover slightly. These gains are meaningful but not dramatic overall. MovieLens ratings are collected through explicit rating sessions where users often score films watched months or years earlier. The temporal order of ratings does not reflect actual viewing sequences, which limits the signal available to any temporal model.

On Amazon Books, the picture changes dramatically. BaseModel improves over HSTU-large by 103% on HR@10 (9.52 vs 4.69) and 138% on NDCG@10 (6.12 vs 2.57). These are not incremental gains; they represent a qualitative difference in prediction quality.

The 138% improvement on NDCG@10 deserves emphasis. NDCG weights top-ranked positions heavily, so this metric directly measures whether the model places genuinely relevant items at the very top of the recommendation list. BaseModel's dramatic advantage indicates that it identifies the most relevant items with substantially higher precision than HSTU, not merely that it ranks items somewhat better on average.

Why Does the Distributional Approach Win?

The performance gap stems from a representational mismatch in sequential approaches. HSTU treats user histories as ordered token sequences, importing the assumption from language modeling that position carries essential information. But as discussed in Section 6.1, even sophisticated attention mechanisms collapse to local context when processing behavioral sequences.

HSTU's own ablation studies support this interpretation. Their stochastic length technique removes over 80% of sequence tokens while maintaining quality within 0.2% normalized entropy. If the vast majority of interaction history can be discarded without meaningful degradation, the sequential structure contains less information than the architecture assumes.

Amazon Books captures real purchase behavior where timestamps correspond to actual purchasing decisions, unlike MovieLens ratings entered retrospectively. Yet BaseModel, which discards explicit temporal ordering in favor of distributional representation, captures the predictive structure more effectively than HSTU's sequential transduction.

The density sketch representation captures exactly what remains informative after aggressive sequence pruning: the distributional pattern of which items were engaged with and how recently, without the computational overhead of attention over positions that prove dispensable.

Architectural Implications

Beyond predictive performance, the architectural disparity has practical implications. HSTU's efficiency innovations, including pointwise attention, stochastic length training, and hierarchical activation management, were necessary to make trillion-parameter sequential models deployable. These represent substantial engineering investment to overcome fundamental scaling challenges of attention-based architectures.

BaseModel avoids these challenges entirely. Profile construction requires only summing precomputed bucket codes. The feed-forward backbone involves no attention computation, no sequence processing, no KV caching. Training and inference costs scale with profile dimension and network depth, not with sequence length.

6.7.3 Comparison with beeFormer

Table 6.7 presents results on Amazon Books under the temporal split protocol used by beeFormer. This evaluation holds out the last 20% of interactions chronologically, simulating streaming prediction.

BaseModel outperforms all baselines, improving over the best beeFormer variant by 6–22% across metrics. The improvements are particularly pronounced at deeper ranks: 6.2% on Recall@20, but 18.6% on Recall@50 and 21.9% on NDCG@100. This suggests that BaseModel captures relevant items that other methods miss entirely, not just that it ranks the same items slightly better.

The comparison with language model approaches is instructive. beeFormer's domain-adapted models leverage pretrained text representations with billions of parameters, fine-tuned specifically on this domain. BaseModel achieves better results by representing behavior directly as density sketches rather than routing through language model embeddings. The behavioral signal, when properly represented, dominates the semantic signal from item descriptions.

Table 6.7: Amazon Books with temporal split. Values are percentages.

Method	Recall@20	Recall@50	NDCG@100
Sentence Transformers			
all-mpnet-base-v2	2.18	3.36	1.93
nomic-embed-text-v1.5	3.87	5.60	3.20
bge-m3	3.98	5.46	3.13
Collaborative Filtering			
KNN	3.70	5.62	3.03
ALS MF	3.44	5.80	3.13
ELSA	3.67	6.28	3.46
SANSA	4.21	6.78	3.62
beeFormer (interaction-aware)			
Llama-goodbooks-mpnet	6.49	9.31	5.15
Llama-goodlens-mpnet	6.17	8.91	4.92
Llama-amazbooks-mpnet	7.06	10.45	5.71
BaseModel	**7.50 ± 0.14**	**12.39 ± 0.21**	**6.96 ± 0.12**
vs. best baseline	+6.2%	+18.6%	+21.9%

6.8 Relational Prediction Results

Sequential recommendation evaluates temporal modeling on user-item logs, but many real-world prediction tasks involve richer relational structure: multiple entity types, complex table joins, and heterogeneous features. The RelBench benchmark provides exactly this challenge, with multi-table databases spanning e-commerce, advertising, fashion retail, and question-answering platforms.

Table 6.8 summarizes results across recommendation, classification, and regression tasks.

6.8.1 Recommendation Tasks

BaseModel achieves state-of-the-art results on four of six recommendation tasks. The improvements are substantial where the method's strengths align with task structure.

On rel-amazon, BaseModel improves over ContextGNN by 36% for predicting 5-star ratings and 55% for predicting detailed reviews. These tasks have long-tailed target distributions where most users never leave ratings or reviews. Density-based profiles handle this naturally by representing the full behavioral distribution rather than optimizing for the majority class.

The rel-avito ad-visit task tests cold-start handling directly: 71% of test items were never seen during training. BaseModel improves over RelGNN by 19% because sketch-based profiles incorporate content features that give new items meaningful representations immediately.

On rel-hm fashion purchases, BaseModel achieves a 25% gain over ContextGNN. Fashion recommendation benefits from multimodal integration: visual similarity captured through image embeddings complements collaborative signal from purchase patterns.

The exceptions reveal where BaseModel's design is less well-matched. On rel-amazon purchase, ContextGNN edges ahead by 1% (2.93 vs 2.89 mAP), a margin within run-to-run variance. On rel-stack

Table 6.8: RelBench results. Best results in bold. For recommendation (mAP) and classification (AUC), higher is better. For regression (MAE), lower is better.

(a) Recommendation (Mean Average Precision)

Dataset	Task	RelGNN	ContextGNN	BaseModel
rel-amazon	purchase	0.77	**2.93**	2.89 ± 0.05
	rate	0.92	2.25	**3.06 ± 0.06** (+36%)
	review	0.52	1.63	**2.53 ± 0.07** (+55%)
rel-avito	ad-visit	3.94	—	**4.68 ± 0.08** (+19%)
rel-hm	purchase	2.81	2.93	**3.67 ± 0.05** (+25%)
rel-stack	comment	**14.00**	13.34	13.06 ± 0.12

(b) Classification (AUC)

Dataset	Task	RelGNN	BaseModel
rel-amazon	churn	70.99	**71.02 ± 0.07**
rel-avito	visits	66.18	**66.25 ± 0.09**
	clicks	68.23	**68.54 ± 0.06**
rel-hm	churn	70.93	**71.25 ± 0.05**
rel-stack	engage	**90.75**	90.00 ± 0.10

(c) Regression (MAE, lower is better)

Dataset	Task	RelGNN	ContextGNN	BaseModel
rel-amazon	user-ltv	**14.23**	—	14.26 ± 0.02

comment prediction, RelGNN leads with 14.00 versus 13.06 mAP. Stack Exchange has unusual structure: users form tight communities around specific topics and repeatedly engage with the same threads over years. GNN message passing aggregates signals from immediate neighbors, which aligns well when these persistent local patterns dominate. Density sketches summarize behavior globally and may dilute strong local signals.

6.8.2 Classification and Regression Tasks

For classification, BaseModel matches or slightly exceeds RelGNN on four of five tasks. The margins are small, suggesting that for classification on relational data, the choice of representation matters less than for recommendation. Both approaches extract sufficient signal to approach performance ceilings.

For regression, BaseModel ties with RelGNN on user lifetime value prediction (14.26 vs 14.23 MAE).

6.8.3 Summary

Across 12 RelBench tasks spanning three prediction types, BaseModel matches or exceeds specialized graph neural network methods on 10 tasks. The unified density-based framework achieves this without

task-specific architectural modifications: the same behavioral profiles and foundation backbone support recommendation, classification, and regression through different output heads.

6.9 Ablation Studies

The previous sections demonstrated that BaseModel achieves strong results across diverse tasks. This section investigates which components drive these gains through systematic ablation.

6.9.1 The Role of Metadata in Cold-Start Scenarios

The role of metadata is evaluated on two RelBench datasets with substantial item cold-start: rel-amazon (18–32% cold-start depending on task) and rel-avito (71% cold-start). Table 6.9 compares the full model against a variant using only interaction-based collaborative signals.

Table 6.9: Metadata ablation. Cold-start Recall measures performance on items unseen during training; Cold-start Rate indicates the fraction of recommendations that are new items.

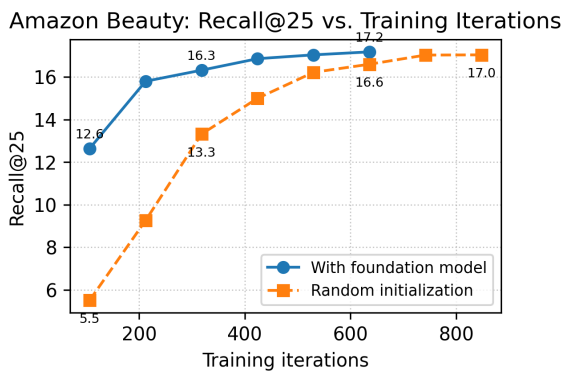
Configuration	mAP	Prec	Recall	Cold-start Recall	Cold-start Rate
rel-amazon purchase (18% cold-start)					
Interactions only	2.76	1.09	5.69	0.00	0.00
+ metadata	**2.89**	**1.15**	**6.03**	**2.58**	**5.95**
rel-amazon review (32% cold-start)					
Interactions only	2.24	0.95	4.54	0.00	0.00
+ metadata	**2.53**	**1.09**	**5.30**	**2.90**	**6.96**
rel-avito ad-visit (71% cold-start)					
Interactions only	3.55	3.06	4.23	0.00	0.00
+ metadata	**4.68**	**5.07**	**5.51**	**3.71**	**80.7**

The results reveal two distinct effects.

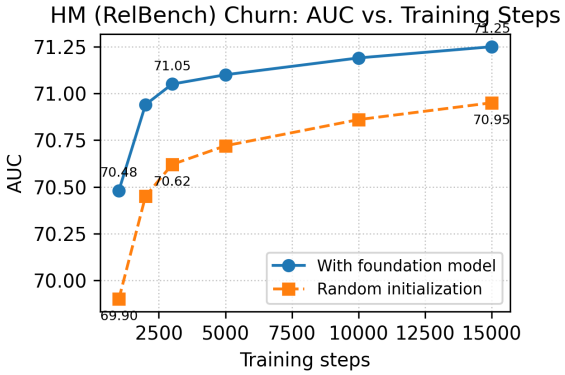
First, metadata enables cold-start inference. For items with no interaction history, metadata is the only available signal. Removing it eliminates cold-start capability entirely: cold-start recall drops to zero because the model has literally nothing to work with for unseen items. With metadata, the model achieves 2.6–3.7% cold-start recall while naturally surfacing new items in 6–81% of recommendations.

Second, metadata improves overall performance even on items with interaction history. On rel-amazon, adding metadata improves mAP by 5–13% and Recall by 6–17%. The semantic information in product descriptions enriches behavioral profiles beyond what co-occurrence patterns alone provide.

The rel-avito results merit particular attention. In a dataset where 71% of test items are unseen during training, metadata transforms the model from unusable to effective. The interaction-only model achieves 3.55 mAP, but this performance comes entirely from established items. Adding metadata improves mAP to 4.68 (+32%), Precision from 3.06 to 5.07 (+66%), and Recall from 4.23 to 5.51 (+30%). Most strikingly, 80.7% of items in top recommendations are cold-start, demonstrating that the model treats new and established items on genuinely equal footing.



(a) Recommendation on Amazon Beauty



(b) Classification on rel-hm

Figure 6.1: Effect of foundation pretraining. Models initialized from the pretrained backbone converge faster and achieve higher final performance than identical architectures trained from scratch.

6.9.2 The Role of Visual Features

Fashion recommendation depends heavily on visual appearance. Table 6.10 shows that adding visual embeddings on rel-hm yields consistent improvements of 3–4% across metrics.

Table 6.10: Visual feature ablation on rel-hm purchase prediction.

Configuration	mAP@12	Precision@12	Recall@12
Interactions only	3.56	1.53	7.65
+ Visual features	**3.67**	**1.59**	**7.87**

These gains are smaller than those from metadata, reflecting the nature of the information. Metadata provides direct semantic signal; visual features capture subtler aspects of appearance. Both contribute complementary information that collaborative signals alone cannot provide.

6.9.3 Foundation Pretraining

Does self-supervised pretraining actually help, or would training from scratch work equally well? Two variants of identical architecture are compared: one initialized from the pretrained foundation backbone, and one with random initialization.

Figure 6.1 shows learning curves for both recommendation and classification tasks.

On Amazon Beauty recommendation, the pretrained model reaches 12.64 Recall@25 at iteration 106 while the scratch model sits at 5.51. By iteration 318, the pretrained model achieves 16.30, a level the scratch model does not reach until iteration 636. The pretrained backbone converges roughly twice as fast and retains a small performance advantage at convergence (17.2 vs. 17.0 Recall@25).

On rel-hm churn prediction, a binary classification task structurally different from the recommendation-style pretraining objective, pretraining still helps. The foundation-initialized model starts at 70.48 AUC after 1,000 steps versus 69.90 for random initialization, and the gap persists throughout training. This

Table 6.11: Factorized linear layer ablation on rel-amazon. All values are percentages. Relative change shows the effect of replacing standard linear layers with FMLinear.

Task	Metric	Standard Linear	FMLinear	Relative
purchase	mAP@10	2.78	**2.89**	+4.0%
	Recall@10	5.92	**6.07**	+2.5%
rate	mAP@10	2.93	**3.06**	+4.4%
	Recall@10	6.17	**6.35**	+2.9%
review	mAP@10	2.38	**2.53**	+6.3%
	Recall@10	5.15	**5.35**	+3.9%

demonstrates that the foundation stage learns transferable behavioral structure, not just task-specific patterns.

6.9.4 The Role of Factorized Feature Interactions

The factorized linear layer (Section 6.4.3) introduces pairwise feature interactions at every projection in the network. Does this actually help, or would standard linear layers suffice given that the density sketch already provides a rich input representation?

Table 6.11 compares the full BaseModel (with FMLinear) against an identical architecture where every factorized linear layer is replaced with a standard linear projection, with the bottleneck expansion factor increased from $1\times$ to $4\times$ to compensate for the removal of the quadratic term. All other hyperparameters, training procedures, and evaluation protocols remain unchanged.

FMLinear improves performance consistently across all three tasks and both metrics, with relative gains ranging from 2.5% to 6.3%. The largest improvements appear on review prediction, which has the highest cold-start rate among the three (32% of test items unseen during training). This is consistent with the intuition that cross-modal interactions matter most when the model must rely on metadata-derived sketch components in the absence of interaction history, though the correlation could also reflect other task-specific factors.

All BaseModel results reported elsewhere in this thesis use FMLinear as the default projection layer.

6.10 Computational Analysis

Competitive performance means little if the method is computationally impractical. This section examines the efficiency properties that enable BaseModel to operate at production scale.

6.10.1 Pipeline Stages and Timing

The computational pipeline divides into two phases. The *fit phase* executes once per dataset: data loading, graph construction, Cleora embedding, EMDE sketching, and foundation model training. The *downstream phase* optimizes task-specific heads, reusing the pretrained backbone.

Table 6.12 reports wall-clock times on a single NVIDIA H100 GPU.

Table 6.12: Computational performance. Fit includes all preprocessing and foundation training. Train reports downstream optimization only.

Dataset	Task	Fit (min)	Train (min)	Throughput	p50 (ms)
rel-avito	ad-visit	53.9	28.6	2,976/s	9.00
rel-avito	clicks	53.9	1.2	3,201/s	0.10
rel-hm	purchase	181.1	61.2	3,758/s	0.18
rel-hm	churn	181.1	28.2	4,571/s	0.03
rel-amazon	purchase	78.1	48.3	2,680/s	0.99
Amazon Books	temporal	124.1	82.3	3,952/s	0.15
Amazon Books	LOO	123.1	107.6	1,201/s	1.21

Fit phase duration scales with dataset complexity: rel-hm requires 181 minutes due to visual feature extraction, while simpler datasets complete in under an hour. Downstream training is consistently fast, ranging from one minute for classification to under two hours for the largest recommendation settings. Most tasks achieve sub-millisecond inference latency.

6.10.2 Comparison with Graph Neural Networks

The efficiency advantage over graph neural networks is substantial. In the session-based experiments of Chapter 5, SR-GNN and TAGNN required over one week for hyperparameter tuning on RSC15 and 30Music before timing out. BaseModel variants trained in under one hour. The disparity stems from architectural differences: graph neural networks perform expensive neighborhood aggregation during both training and inference, while BaseModel's graph operations occur only during the one-time fit phase.

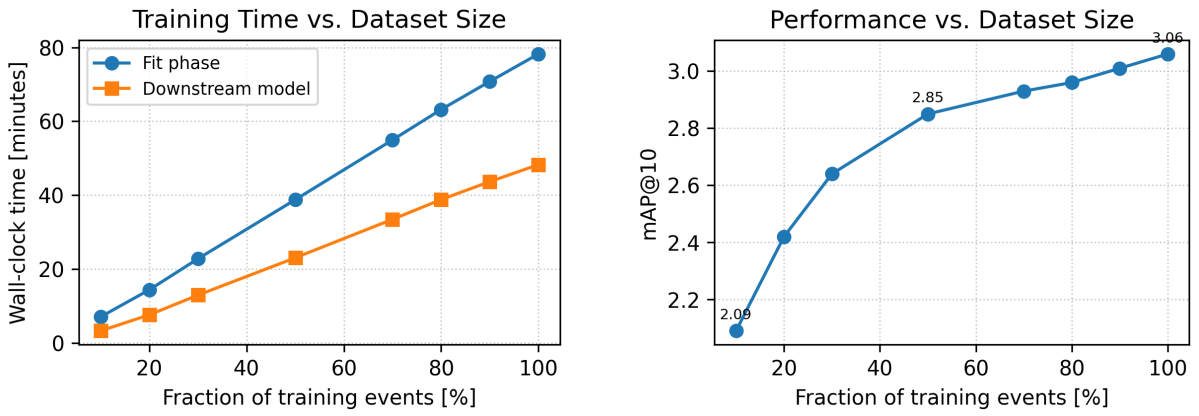
6.10.3 Scaling Behavior

To understand how BaseModel behaves as data volume varies, a controlled experiment on rel-amazon created subsets containing 10% to 100% of training events.

Figure 6.2 shows that both time and performance scale predictably. Training time grows nearly linearly with event count. Performance increases monotonically with diminishing returns: half of the data recovers most of the final accuracy. These properties have practical implications: BaseModel provides useful predictions even with limited historical data, while the near-linear scaling enables regular retraining as data accumulates.

6.11 Production Deployment

Laboratory benchmarks measure what a model can do under controlled conditions. Production deployment reveals what it does under real constraints: latency budgets, infrastructure limitations, data freshness requirements, and the unpredictable diversity of actual user behavior. This section reports results from multiple deployments across different industries, demonstrating that the methods generalize beyond controlled experimental settings.



(a) Training time

(b) Performance (mAP@10)

Figure 6.2: Scaling behavior on rel-amazon. Time grows linearly; performance shows diminishing returns.

6.11.1 Fashion E-commerce: Email Personalization

The first production deployment integrated BaseModel into the personalization stack of a large European fashion e-commerce platform operating across multiple countries. BaseModel replaced the personalization component in the email marketing channel, where the incumbent system combined collaborative filtering with an XGBoost-based ranker built on hand-engineered behavioral and product features refined over several years.

The evaluation followed a rigorous A/B testing protocol. Users were randomly assigned to control (incumbent) or treatment (BaseModel) groups across multiple weekly campaign cycles. The test covered different campaign types and seasonal periods, producing approximately 2,030,000 emails with balanced assignment between groups.

The treatment variant powered by BaseModel improved all primary engagement metrics. Open rates increased by 21%, click-through rates by 18%, and conversion rates by 7%. These gains occurred without changes to promotional incentives, email templates, or sending schedules. The conversion rate improvement, while smaller in percentage terms, has direct revenue impact: users who opened and clicked were more likely to complete purchases. Notably, these improvements were achieved without additional discounting, preserving profit margins while increasing engagement.

6.11.2 Financial Services: Predictive Analytics

The financial services sector presents distinct challenges for behavioral modeling: stringent regulatory requirements, heightened data sensitivity, and the need for explainable predictions that can withstand audit scrutiny. Two banking deployments validated that the density-based approach meets these requirements while delivering substantial predictive improvements.

BNP Paribas became the first Polish bank to deploy BaseModel in production during 2024 [4]. The proof-of-concept phase trained eight distinct business models, with initial deployment focused on retail banking personalization and planned expansion to SME and corporate segments. The deployment archi-

tecture satisfied the bank's security requirements through on-premises installation, demonstrating that the containerized deployment model supports even the most regulated environments.

A second banking deployment, documented in aggregate to protect client confidentiality, achieved fourfold improvement in customer retention prediction accuracy compared to the incumbent heuristic-based system [66]. The practical impact of this improvement was substantial: in one documented case, timely intervention triggered by BaseModel's detection of diminishing engagement patterns secured retention of a six-figure deposit that would otherwise have churned. The improvement in prediction accuracy directly supports hypothesis H4, demonstrating that self-supervised pretraining on behavioral dynamics transfers effectively to classification tasks in domains far removed from the e-commerce settings used for initial development.

mBank, serving 5.6 million retail clients across Poland, Czech Republic, and Slovakia, reported operational benefits beyond prediction accuracy [64]. Lead generation increased by 60% through improved targeting precision. Perhaps more significantly for practical deployment, time-to-market for new predictive use cases decreased from several months to approximately one week. This acceleration reflects the foundation model paradigm: rather than engineering features and training models from scratch for each business question, analysts fine-tune the pretrained backbone with task-specific heads, dramatically reducing the iteration cycle.

6.11.3 Telecommunications: Digital Channel Optimization

Orange Polska deployed the platform to optimize digital channel personalization across web and mobile touchpoints [65]. The telecommunications context differs from retail in important ways: longer customer lifecycles, complex product bundles, and the need to balance acquisition, retention, and upselling objectives simultaneously.

The deployment achieved 75% conversion rate on dynamically personalized content, compared to substantially lower rates for static content. Annual digital channel sales increased by 2.1%, while additional services attachment improved from 3% to 14% of eligible transactions. These results demonstrate that the density-based behavioral representation captures patterns relevant to subscription businesses, not merely transactional e-commerce.

6.11.4 Cross-Industry Synthesis

The consistency of positive results across fashion retail, banking, and telecommunications suggests that the methods capture something general about behavioral data rather than exploiting domain-specific patterns. Three properties of the approach proved decisive across all deployments:

The cold-start capability proved valuable across all settings. Fashion retail continuously introduces new products; banking launches new financial instruments; telecommunications bundles evolve with market conditions. In each case, the ability to generate meaningful predictions for entities with limited interaction history provided operational advantages that pure collaborative filtering approaches cannot match.

The efficiency of the representation pipeline enabled deployment patterns that would be infeasible with more computationally intensive approaches. Banking deployments required on-premises installation

within existing infrastructure constraints. The fashion e-commerce deployment needed to meet strict campaign generation deadlines. In both cases, the fixed computational budget of sketch-based profiling, independent of catalog size or history length, proved essential.

The transfer learning properties validated by the RelBench experiments manifested concretely in production. The banking deployment's fourfold improvement in churn prediction accuracy was achieved by fine-tuning the same pretrained backbone used for recommendation, confirming that the foundation model learns behavioral dynamics that generalize across prediction tasks.

6.11.5 Commercial Trajectory

The methods developed in this thesis have followed two deployment paths that together stress-test different aspects of the methodology. The primary path integrated EMDE and the foundation model framework into the Synerise Platform, which by late 2024 was processing 42 billion behavioral events monthly across over 200 organizations [18]. The deployments at BNP Paribas, mBank, Orange Polska, Empik, Modivo, Nike, and the fashion e-commerce platform described above operate within this infrastructure.

A secondary path released the core methodology as a standalone product, BaseModel.ai, in January 2023, available through Microsoft Azure Marketplace and Snowflake Marketplace as a native application [67]. The standalone packaging validates a different deployment constraint: organizations that cannot route behavioral data through an external platform. The containerized model runs on a single GPU-equipped server without cluster infrastructure, and supports air-gapped installation for regulated environments.

Strategic partnerships have extended the reach of the methodology into new market segments. VTEX, a major commerce platform serving 3,500 online stores across 43 countries, invested $8.5 million in 2024 with a commitment to develop AI products based on the BaseModel methodology [68]. Dunnhumby, the global customer data science company with 35 years of retail analytics expertise, announced a strategic partnership in early 2025 to combine their customer science capabilities with Synerise's real-time behavioral AI [17].

This dual trajectory, with broad deployment through an integrated platform and targeted availability as a standalone product, provides validation that controlled benchmarks cannot: the methodology meets requirements ranging from enterprise-scale infrastructure to focused analytical workloads, operating reliably when data quality varies, latency budgets are strict, and prediction accuracy is measured by revenue outcomes rather than offline metrics.

6.12 Limitations

Several limitations warrant acknowledgment.

Dataset-specific training. Although BaseModel employs self-supervised pretraining, the foundation stage trains independently for each dataset. Behavioral profiles depend on dataset-specific embeddings. Cross-dataset transfer, where a model trained on one behavioral domain generalizes to another without retraining, remains unexplored.

Reduced emphasis on sequential structure. EMDE represents histories as weighted sets, with temporal information entering through recency weighting rather than explicit sequence modeling. For

domains where sequential dependencies genuinely matter (educational progressions, narrative content), this may not capture all relevant structure.

Local structure limitations. The rel-stack results revealed that when tight local community structure dominates, graph neural networks that aggregate neighborhood signals outperform the global density representation. Density sketches may dilute strong local patterns.

Random partitions. EMDE uses random hyperplane partitions with density-dependent thresholds. Learned partitions that adapt to the prediction task might improve performance, but would complicate the pipeline and sacrifice efficiency benefits.

6.13 Hypothesis Evaluation

This chapter addressed three of the five hypotheses introduced in Chapter 1. The evidence accumulated across the experimental sections now permits evaluation of each.

Hypothesis H2: Density sketching enables effective multimodal integration through simple concatenation. This hypothesis predicted that different modalities could be integrated through concatenation of modality-specific sketches without requiring learned fusion architectures, and that such integration would yield consistent performance improvements.

Verdict: Supported.

The evidence consistently supports this hypothesis across all experimental settings. In the metadata ablation experiments (Table 6.9), adding product metadata through sketch concatenation improved mAP by 5–13% on rel-amazon and 32% on rel-avito. On rel-hm, concatenating visual feature sketches improved mAP by 3.1% (Table 6.10). In every case, the integration mechanism was identical: compute sketches independently for each modality, concatenate them, and let the downstream network learn to weight the signals appropriately. No learned fusion architectures, gating mechanisms, or cross-modal attention were required. The network learns implicitly how to weight different modalities based on their predictive value for each task.

Hypothesis H4: Self-supervised pretraining on behavioral dynamics enables transfer across task types. This hypothesis predicted that models initialized from a pretrained foundation backbone would converge faster and achieve better final performance than identical architectures trained from scratch, even for tasks structurally different from the pretraining objective.

Verdict: Supported.

The pretraining ablation experiments (Section 6.9.3) provide direct evidence. On Amazon Beauty recommendation, the pretrained model converged roughly twice as fast and retained a small performance advantage at convergence (17.2 vs. 17.0 Recall@25; Figure 6.1a). On rel-hm churn prediction, a binary classification task structurally different from the recommendation-style pretraining objective, pretraining improved AUC by 0.3 points throughout training (Figure 6.1b). The convergence speed advantage was consistent across both tasks; the final-performance gains, while present, were modest. Together, these ablations indicate that the foundation stage learns transferable representations of behavioral dynamics

rather than task-specific patterns, with the primary practical benefit being reduced training cost rather than large accuracy improvements.

The RelBench results provide complementary, though indirect, evidence. A single pretrained backbone with lightweight task-specific heads matched or exceeded specialized graph neural networks on 10 of 12 tasks spanning recommendation, classification, and regression (Table 6.8). This demonstrates that the pretrained system generalizes across task types, but because the comparison is against different architectures rather than against the same architecture without pretraining, it does not isolate the marginal contribution of pretraining from the representational advantages tested under H1 and H3. The direct ablation also did not include regression tasks, so the regression evidence for H4 remains indirect. Nonetheless, the combination of faster convergence in controlled ablation and strong cross-task performance in the broader evaluation supports the hypothesis.

Hypothesis H5: Density-based representations provide disproportionate advantages in cold-start scenarios. This hypothesis predicted that content features would enable non-trivial recommendation performance on previously unseen items through the same inference mechanism used for established items, whereas interaction-only approaches would achieve zero cold-start recall.

Verdict: Strongly supported.

The metadata ablation experiments (Table 6.9) demonstrated dramatic effects. On rel-avito, where 71% of test items were unseen during training, adding metadata improved mAP by 32% and enabled the model to surface new items at an 80.7% rate among top recommendations. On rel-amazon, metadata enabled 2.6–3.1% cold-start recall where the interaction-only model achieved literally zero. Without content features, the model had no signal to work with for unseen items.

The density sketch representation handles cold-start naturally because it operates on embedding space rather than item identities. Any item with content features (text description, image, categorical attributes) receives meaningful bucket codes immediately through the same EMDE encoding used for established items. These codes position the item in relevant regions of the predicted distribution without approximation or fallback mechanisms. The inference procedure is identical for cold-start and established items; only the source of the embedding differs (content features alone versus content features plus collaborative signal).

Summary. Table 6.13 summarizes the hypothesis evaluation.

All three hypotheses addressed in this chapter are supported by the experimental evidence. The density-based representation enables effective multimodal integration (H2), the self-supervised pretraining objective learns transferable behavioral structure (H4), and content features provide immediate cold-start capability through the same inference mechanism used for established items (H5). These findings complement the evidence for H1 and H3 presented in Chapter 5, collectively validating the density estimation perspective on behavioral data that motivates this dissertation.

Table 6.13: Summary of hypothesis evaluation in this chapter.

Hypothesis	Verdict	Primary Evidence
H2: Multimodal integration through concatenation	Supported	+5–32% mAP from concatenating modality sketches; no fusion architecture required
H4: Self-supervised pretraining enables cross-task transfer	Supported	2× faster convergence (direct ablation); +0.3 AUC on classification (direct); 10/12 RelBench tasks (indirect)
H5: Cold-start advantages from density representation	Strongly supported	80.7% of recommendations are cold-start items on rel-avito; zero cold-start recall without content features

6.14 Summary

This chapter introduced BaseModel, a foundation model framework that treats behavioral data as evolving distributions over a learned manifold rather than as sequences of tokens.

The core insight is that behavioral change is better characterized as distributional drift than as item-to-item transitions. A user interested in photography today will likely remain interested tomorrow, perhaps expanding to related areas. This drift manifests as gradual migration of probability mass across embedding space, a smooth transformation that neural networks can learn efficiently. In contrast, predicting specific next items requires modeling high-frequency noise from catalog availability, interface design, and countless other factors.

BaseModel implements this insight through three components. Universal behavioral profiles combine Cleora hypergraph embeddings with EMDE density sketches, producing fixed-size multimodal representations that preserve distributional structure. A self-supervised foundation model learns how these distributions evolve over time, trained by predicting future sketches from past sketches without task-specific labels. Lightweight task-specific heads adapt the pretrained backbone to recommendation, classification, and regression.

The experimental results validate this approach across diverse settings:

- Against TIGER, 54–82% improvements on sparse Amazon datasets
- Against HSTU, 103–138% improvements on Amazon Books
- Against graph neural networks, state-of-the-art on 10 of 12 RelBench tasks
- In production, 21% higher email engagement at a major European retailer

The pattern of results is telling. Gains are largest precisely where sequential models struggle: sparse data, long-tailed item distributions, substantial cold-start exposure. The density formulation sidesteps these challenges by learning from how regional preferences evolve across many users, rather than trying to estimate transition probabilities from limited individual histories.

These results support the hypotheses introduced at the start of this thesis. Hypothesis H2 (multimodal integration through concatenation) is supported by consistent improvements from adding metadata and

visual features without fusion architectures. Hypothesis H4 (self-supervised pretraining enables cross-task transfer) is supported by faster convergence and better final performance when initializing from the pretrained backbone, across both recommendation and classification tasks. Hypothesis H5 (cold-start advantages) is supported by the rel-avito results where metadata enables 80% of recommendations to be cold-start items that the interaction-only model cannot surface at all.

The methods developed here have moved from research into commercial deployment, providing validation that controlled benchmarks cannot address: infrastructure flexibility, production-scale latency, and operation within real-world constraints. The trajectory from doctoral research to systems serving millions of users provides evidence that density-based behavioral modeling offers not only theoretical advantages but practical utility.

7 Validation Through Competitive Benchmarks

This chapter is based on the following publications:

Michał Daniluk, Barbara Rychalska, Konrad Gołuchowski, and Jacek Dąbrowski. "Modeling Multi-Destination Trips with Sketch-Based Model." *ACM WSDM Workshop on Web Tourism (WebTour'21)*, March 2021. **2nd place, Booking.com Data Challenge. Best Paper Award.**

Michał Daniluk, Jacek Dąbrowski, Barbara Rychalska, and Konrad Gołuchowski. "Synerise at KDD Cup 2021: Node Classification in Massive Heterogeneous Graphs." *KDD Cup 2021*, August 2021. **3rd place, MAG240M-LSC Track.**

Michał Daniluk, Jacek Dąbrowski, Barbara Rychalska, and Konrad Gołuchowski. "Synerise at RecSys 2021: Twitter User Engagement Prediction with a Fast Neural Model." *ACM RecSys Challenge Workshop*, October 2021. **2nd place, Twitter Engagement Prediction Challenge.**

My contributions: I led the development of all three competition solutions, designed the system architectures, implemented the core algorithms, and was the first author on all resulting publications. The Booking.com solution received the Best Paper Award at the WebTour 2021 workshop.

7.1 Introduction

The preceding chapters established the theoretical foundations and benchmark performance of density-based behavioral modeling. Chapter 5 demonstrated that EMDE achieves state-of-the-art results on session-based recommendation datasets. Chapter 6 showed that BaseModel outperforms sophisticated sequential architectures on standard evaluation protocols. These results provide necessary evidence for the viability of the proposed methods, but they are not sufficient to establish their practical value.

Academic benchmarks, despite their importance, possess inherent limitations as validation instruments. Datasets are fixed and publicly available, allowing researchers to iterate indefinitely until successful configurations emerge. Evaluation protocols are published in advance, enabling implicit optimization toward known metrics. Perhaps most critically, the community's collective exploration of popular benchmarks creates a form of selection bias: methods that succeed on these specific datasets receive attention and publication, while methods that fail quietly disappear. A technique that achieves strong benchmark

performance may have been implicitly tuned to exploit particular dataset characteristics rather than capturing generalizable patterns in behavioral data.

Machine learning competitions provide a complementary form of validation that addresses several of these concerns. Test sets remain hidden until after submission deadlines, precluding iterative optimization on evaluation data. Strict time constraints limit the extent of hyperparameter exploration. Competitors face unfamiliar data distributions and must make architectural decisions without knowing which choices will prove effective. Success requires methods that generalize beyond the specific patterns observed during development.

This chapter documents the application of density-based methods to three international competitions conducted in 2021. The consistency of strong results across diverse settings provides evidence that the methods capture fundamental properties of behavioral data rather than exploiting benchmark-specific artifacts:

- The **Booking.com WSDM Data Challenge** addressed multi-destination travel prediction, requiring systems to recommend the final city of multi-city trips based on preceding reservations. The solution placed second among 40 teams and received the Best Paper Award at the WebTour workshop.
- The **KDD Cup 2021 MAG240M-LSC Track** required node classification in a heterogeneous academic graph containing 244 million nodes and over one billion edges. The solution placed third among approximately 150 teams.
- The **ACM RecSys Challenge 2021** demanded Twitter engagement prediction under strict computational constraints, limiting inference to approximately 6 milliseconds per prediction on a single CPU core. The solution placed second overall.

These competitions span distinct domains (travel, academic publishing, social media), different prediction tasks (next-destination recommendation, multi-class classification, multi-label regression), and varying computational constraints (the KDD Cup tested scalability to graphs requiring 250GB of storage; the RecSys challenge enforced real-time inference requirements). The common methodological thread across all three solutions was the combination of Cleora graph embeddings with EMDE density sketches, processed by shallow feed-forward neural networks.

The chapter is organized as follows. Section 7.2 presents the Booking.com challenge, which provides the most direct test of whether density-based representations outperform sequential alternatives for behavioral prediction. Section 7.3 describes the KDD Cup solution, demonstrating that the approach scales to massive graphs while maintaining competitive accuracy. Section 7.4 covers the Twitter engagement prediction challenge, validating that density sketches meet production latency requirements. Section 7.5 synthesizes observations across all three experiences, evaluates the evidence they provide for the thesis hypotheses, and discusses the limitations of competition-based validation.

7.2 Multi-Destination Trip Prediction

The Booking.com WSDM WebTour 2021 Challenge focused on a practical problem in travel recommendation: predicting where travelers will go next based on their itinerary so far. This competition provides

particularly valuable evidence for the thesis because it enables direct comparison between density-based and sequential approaches on identical data with identical evaluation protocols. The complete solution is available at `https://github.com/Synerise/booking-challenge`.

7.2.1 Task Description

Booking.com released a dataset of 1,166,835 anonymized hotel reservations spanning 217,686 trips across 39,901 unique cities in 195 countries. Each reservation included check-in and check-out dates, booking channel, device type (desktop or mobile), booker country, and hotel country. Trips were defined as sequences of at least four consecutive reservations by the same user.

The task required predicting the final city of each trip given the preceding reservations. The test set comprised 70,662 trips with the final destination concealed. Evaluation used Precision@4: a prediction was considered correct if the true final city appeared among the top four recommendations. This metric directly corresponds to Booking.com's user interface, which displays four suggested destinations to users considering trip extensions. The competition attracted over 800 registered participants, with 97 making final submissions organized into 40 teams.

Several characteristics distinguish this problem from standard sequential recommendation. First, travel itineraries exhibit substantial structural heterogeneity: some travelers visit cities linearly along a route, others make round trips returning to their origin, and many follow complex patterns with backtracking. Second, the same city may serve different functional roles depending on context, functioning as a transit hub, a primary destination, or a return point. Third, approximately 62% of trips contain at least one repeated city, creating cyclic structures that challenge models assuming linear progression through item space.

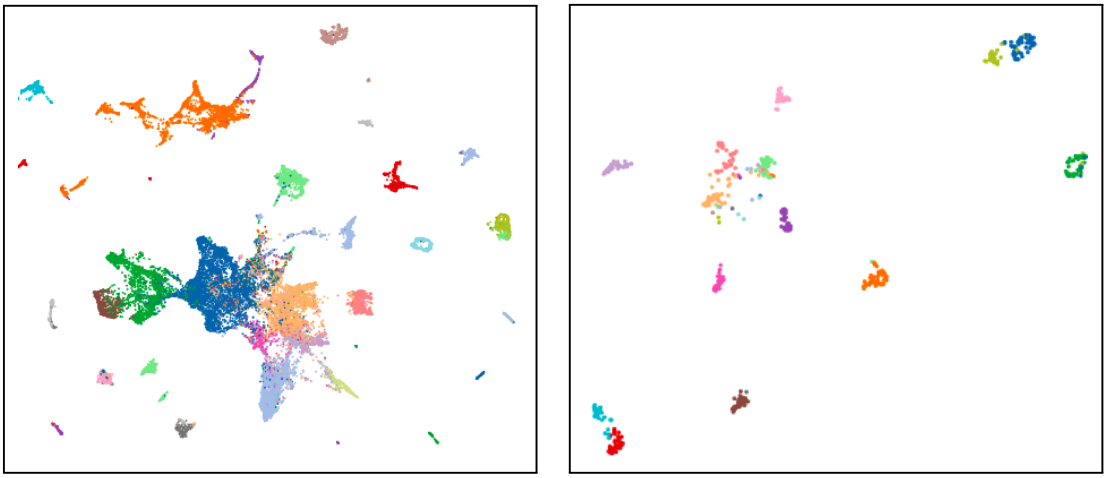
7.2.2 Solution Architecture

The solution architecture directly instantiates the density-based framework developed in this thesis: Cleora embeddings capture the relational structure of travel patterns, EMDE aggregates trip histories into fixed-size distributional representations, and a feed-forward network predicts the final destination.

Graph construction and city embeddings. The trip data naturally induces a directed graph where cities are nodes and travel segments are edges. Each booking creates a directed edge from the origin city to the destination city, with edge weights reflecting the frequency of that transition across the entire dataset. This graph was constructed from both training and test data (excluding the concealed final destinations) to maximize the information available for embedding computation.

Cleora was applied to compute city embeddings from this graph structure. To capture complementary aspects of city similarity, embeddings were computed with two different iteration counts: $I = 1$ captures immediate neighborhood structure (cities directly connected by frequent travel), while $I = 3$ propagates information through longer paths (cities connected through common intermediate destinations). Both configurations used embedding dimension 1024.

An emergent property of these embeddings merits attention. Despite receiving no geographic information whatsoever, the learned representations exhibit clear geographic clustering. Figure 7.1 shows a UMAP projection of the embedding space: cities from the same country cluster together, and geographically



(a) Top 20 most popular countries (b) Top 60 to 80 most popular countries

Figure 7.1: UMAP projection of Cleora city embeddings colored by country. Left: the 20 most popular countries by reservation count. Right: countries ranked 60–80 by popularity. Geographic structure emerges from co-visitation patterns despite the absence of any geographic features in the input data.

proximate countries occupy adjacent regions. This structure emerges entirely from co-visitation patterns in the behavioral data. Geographic proximity correlates with travel transitions due to practical constraints (flight availability, travel time, visa requirements), and Cleora captures these relationships without explicit geographic features. This observation illustrates the representational power of graph-based embeddings: they can recover latent structure that was never explicitly provided.

Trip representation with EMDE. Given city embeddings, each partial trip was represented as a density sketch using EMDE with $N = 40$ partitionings and $B = 128$ buckets. The two Cleora embedding configurations (iterations $I = 1$ and $I = 3$) were treated as complementary modalities, with separate sketches computed for each and concatenated in the final representation. This design choice reflects hypothesis H2: different embedding configurations capture different aspects of city similarity, and simple concatenation allows the downstream network to learn appropriate weightings without requiring explicit fusion mechanisms.

The input representation decomposed each trip into three sketch components designed to capture distinct predictive signals:

1. **First city sketch**: A sketch containing only the trip's origin city. This component addresses the empirical observation that approximately 15% of trips terminate at their starting point (round trips), making the origin a strong predictor for this subset.

2. **Previous city sketch**: A sketch containing only the immediately preceding city. This captures local transition patterns: the tendency to travel to nearby or well-connected destinations.

Table 7.1: Ablation study on the Booking.com validation set. Components are added incrementally to the baseline EMDE model.

Configuration	Precision@4	Incremental Gain
EMDE with city sketches only	0.552	—
+ Intermediate destination training	0.573	+0.021
+ Categorical and numerical features	0.595	+0.022
+ Popularity adjustment	0.598	+0.003
+ Ensemble of 5 models	0.601	+0.003

3. **Historical sketch**: An aggregated sketch of all other cities in the trip, with exponential temporal decay reducing the influence of earlier destinations. This captures the overall character of the journey.

This decomposition allows the model to learn distinct patterns for return trips, local transitions, and global trip characteristics. The three sketches were L_2-normalized and concatenated with embeddings of categorical features (device type, booking channel, countries) and encoded numerical features (trip duration, days since previous booking, number of cities visited).

Model architecture and training. The prediction model was a three-layer residual feed-forward network with 3,000 hidden units per layer, leaky ReLU activations, and batch normalization. The output was a sketch representing the predicted distribution over final destinations.

Training data was augmented by generating multiple examples from each trip. A trip visiting cities $A \to B \to C \to D$ yielded three training instances: predicting B from $\{A\}$, predicting C from $\{A, B\}$, and predicting D from $\{A, B, C\}$. This data augmentation served two purposes: it increased the effective training set size, and it exposed the model to predictions at different trip stages, improving generalization across varying history lengths.

Training proceeded in two stages. Initial training included all intermediate predictions, helping the model learn general patterns of trip evolution. Fine-tuning then focused specifically on final destination prediction, adapting to the different distribution of trip-ending cities. Training completed in approximately 45 minutes on a single NVIDIA RTX 2080 Ti GPU.

7.2.3 Experimental Results

Ablation study. Table 7.1 presents an ablation study isolating the contribution of each solution component on the validation set.

The baseline EMDE model using only city sketches achieved 0.552 Precision@4, demonstrating that density-based aggregation of city embeddings captures substantial predictive signal even without auxiliary features. Including intermediate destinations in training added 2.1 percentage points, confirming that patterns learned from partial trip prediction transfer effectively to final destination prediction. Categorical and numerical features contributed another 2.2 percentage points. Popularity adjustment (boosting scores of frequently visited cities) and ensembling provided smaller but consistent improvements.

Table 7.2: Comparison of aggregation methods on the Booking.com validation set. All configurations use identical Cleora embeddings, identical auxiliary features, and identical downstream architecture (three-layer feed-forward network). The only difference is how trip histories are aggregated: EMDE treats trips as weighted sets, while GRU processes them as ordered sequences.

Aggregation Method	Embedding Source	Precision@4
GRU (sequential)	Learned during training	0.579
GRU (sequential)	Cleora (pretrained)	0.588
EMDE (density-based)	Cleora (pretrained)	**0.598**

Table 7.3: Performance of EMDE with various graph embedding methods on the Booking.com validation set.

Embedding Method	Precision@4
Cleora	**0.598**
Node2Vec	0.596
LINE	0.595
Word2Vec (on city sequences)	0.591
GRU (learned during training)	0.587

Comparison with sequential models. The central question for this thesis is whether density-based representations outperform sequential modeling when both operate on identical input information. Table 7.2 addresses this question directly through controlled comparison.

The comparison controls for confounding factors by holding constant everything except the aggregation method. The GRU model used hidden size 1024 (selected through validation), processing cities sequentially and producing a hidden state that was fed to the same three-layer feed-forward architecture used by EMDE. All auxiliary features remained identical across configurations.

The results provide direct evidence for hypothesis H1. EMDE achieved 0.598 Precision@4, outperforming GRU with Cleora embeddings (0.588) by 1.7% absolute despite avoiding explicit modeling of city-to-city transition sequences. Temporal structure is captured through recency weighting and separate sketches for the origin, previous city, and historical destinations. The improvement of Cleora embeddings over learned embeddings (0.588 vs. 0.579) demonstrates the value of graph-derived representations, but the further improvement from sequential to density-based aggregation (0.588 to 0.598) isolates the contribution of the representational framework itself.

This result challenges the assumption that sequential structure is essential for trip prediction. The precise order in which cities were visited appears to contain less predictive information than the distributional pattern of which cities were visited and how recently. This finding aligns with the structural characteristics of the data: the 62% of trips containing repeated cities create cyclic patterns that sequential models struggle to represent, while EMDE's set-based aggregation handles such patterns naturally.

Comparison of embedding methods. Table 7.3 compares different graph embedding methods for representing cities within the EMDE framework.

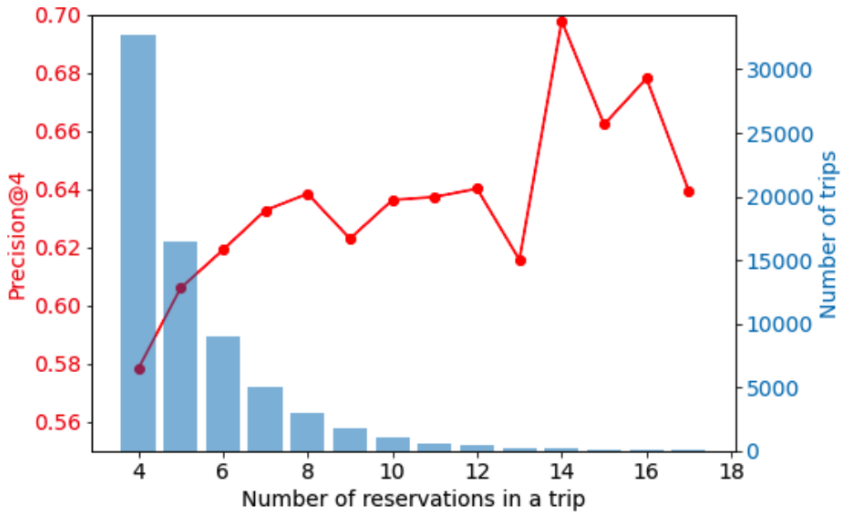


Figure 7.2: Model performance (red line, left axis) as a function of the number of reservations in a trip. The blue histogram (right axis) shows the distribution of trips by length. Performance improves with longer histories, consistent with better density estimation from additional samples.

Cleora achieved the best performance while requiring substantially less computation time than alternatives. Node2Vec and LINE produced competitive results, confirming that the choice of graph embedding method matters less than the choice of aggregation framework. Embeddings learned by a GRU during sequence modeling performed worst, suggesting that end-to-end learned embeddings may overfit to training patterns that do not generalize to the test distribution.

Performance by trip length. Figure 7.2 analyzes model performance as a function of trip length. Performance increases monotonically with the number of reservations: from approximately 0.58 Precision@4 for four-city trips to over 0.68 for trips with 18 or more bookings. This pattern is consistent with the density estimation perspective: longer histories provide more samples from the traveler's preference distribution, enabling more accurate characterization of which regions of city-embedding space they prefer.

The model also learned specific structural patterns effectively. For the 15% of trips where the final destination matches the first city (round trips), the model achieved 0.902 Precision@4, demonstrating successful capture of this common travel pattern.

Competition outcome. The submitted solution achieved 0.578 Precision@4 on the hidden test set, placing second among 40 teams. The gap between validation performance (0.598 for the single EMDE model) and the test result reflects distribution shift between the publicly available validation trips and the withheld test set, a common source of generalization loss in competitions where the validation data can be iterated upon freely. The winning solution (0.594) combined Transformers, GRUs, and MLPs in a deep ensemble, capturing both long-range sequential dependencies and local transition patterns simultaneously. The density-based approach achieved competitive results with a simpler architecture that explicitly avoids sequential modeling, supporting hypothesis H3 that density-based inputs reduce the benefits of architectural complexity. The solution received the Best Paper Award at the WebTour 2021 workshop, recognizing both the technical approach and its presentation.

7.3 Node Classification in Massive Heterogeneous Graphs

The KDD Cup has been organized annually since 1997 as part of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining and represents one of the most prestigious competitions in applied machine learning. The 2021 edition focused on large-scale graph learning, with three tracks testing different aspects of graph-based prediction. The MAG240M-LSC track required node classification in a heterogeneous academic graph of unprecedented scale. The solution placed third among approximately 150 teams. The complete implementation is available at `https://github.com/Synerise/kdd-cup-2021`.

7.3.1 Task Description

The Microsoft Academic Graph (MAG) dataset contained 121 million academic papers, 122 million authors, and 26,000 institutions. Three relation types connected these entities: citation links between papers, authorship links between papers and authors, and affiliation links between authors and institutions. Each paper was represented by a 768-dimensional RoBERTa embedding computed from its title and abstract.

The task was multi-class classification of papers into 153 arXiv subject categories (e.g., cs.LG for Machine Learning, physics.hep-th for High Energy Physics Theory). Only approximately 1.4 million papers had ground-truth labels, creating a semi-supervised learning setting where models must leverage unlabeled graph structure to improve predictions. The data was split temporally: training on papers published through 2018, validation on 2019 papers, and testing on 2020 papers. This temporal split reflects a realistic deployment scenario: classifying newly published papers based on patterns learned from historical data.

The scale presented substantial computational challenges. The complete dataset required approximately 250GB of storage. Standard graph neural networks that aggregate multi-hop neighborhoods would need to access millions of nodes for a single prediction due to the high connectivity of the citation graph. The temporal split created asymmetric information flow: test papers could cite training papers, but training papers could not cite test papers.

7.3.2 Solution Architecture

The solution addressed the scale challenge through a precomputation-based architecture that avoided expensive graph operations during training and inference. All graph structure was distilled into fixed features computed once during preprocessing, enabling a simple feed-forward classifier to achieve competitive accuracy.

EMDE on text embeddings. EMDE was fitted on the RoBERTa embeddings of all papers using sketch depth 40 and width 256, partitioning the semantic embedding space into regions where topically similar papers cluster together. For each paper, aggregate sketches were computed for three types of related papers:

- **Cited papers**: Works referenced by the target paper, capturing its intellectual foundations.
- **Citing papers**: Works that reference the target paper, capturing its intellectual impact.

- **Co-author papers**: Other publications by the same authors, capturing research trajectories.

These sketches summarize the intellectual neighborhood of each paper. A machine learning paper will cite, be cited by, and share authors with other machine learning papers, so its neighborhood sketches will concentrate probability mass in corresponding regions of the embedding space. The aggregation exploits EMDE's additivity: computing a sketch for hundreds of cited papers requires only summing precomputed per-paper sketches, completing in microseconds regardless of neighborhood size.

Temporal constraints were carefully respected: only papers from the same year or earlier than the target paper were included in neighborhood aggregations, preventing information leakage from future publications.

Label propagation with Cleora. A key insight was that known labels could be propagated through the graph to provide soft predictions for unlabeled nodes. Cleora was initialized with a modified scheme: labeled papers received one-hot vectors encoding their subject categories, authors received averaged label vectors from their labeled papers, and unlabeled nodes started with zero vectors. The embedding dimensionality was 153 (matching the number of categories) for all entities.

Cleora's iterative averaging then diffuses label information through the graph. After two iterations, each node's embedding approximates its expected label distribution based on graph proximity to labeled nodes. A paper closely connected to machine learning papers through citations and co-authorship will have high weight on the cs.LG dimension even without a direct label.

To prevent information leakage between Cleora features and training labels, the labeled training data was split: half was used for Cleora propagation, and the other half was used for classifier training. This ensures the classifier cannot exploit direct label information present in its input features.

Institution embeddings. Author-institution affiliations provide additional structural signal. Institutions were embedded using Cleora on a graph constructed through clique expansion: when an author belongs to multiple institutions, all those institutions become connected. This graph contains only institution nodes, with embeddings computed using three Cleora iterations. Institution sketches were then computed for each paper by aggregating across its authors' affiliations using EMDE.

Feature assembly. The complete input concatenated multiple feature types:

- RoBERTa paper embedding (768 dimensions)
- EMDE sketches of cited, citing, and co-author papers ($3 \times 40 \times 256 = 30{,}720$ dimensions)
- EMDE sketch of affiliated institutions ($40 \times 128 = 5{,}120$ dimensions)
- Cleora-propagated label vectors for paper and authors (306 dimensions)
- Numerical features encoded with Fourier Feature Encoding: citation counts, author count, publication year, and L_2 norms of label vectors

The total input dimension was 41,381 features per paper.

Model architecture and training. The classifier was a four-layer residual feed-forward network with 3,500 hidden units per layer, leaky ReLU activations, and batch normalization. Training used AdamW with weight decay 0.01, batch size 512, and a two-stage learning rate schedule (0.0001 for the first epoch, 0.00005 for the second). Training a single model required approximately 42 minutes on one NVIDIA V100 GPU.

For the final submission, 60 models were trained with different random seeds and different splits for Cleora label propagation, with predictions averaged across the ensemble.

7.3.3 Results

Table 7.4 presents the final competition standings.

Table 7.4: KDD Cup 2021 MAG240M-LSC final standings (top 4 teams and baseline).

Rank	Team	Test Accuracy
1	Baidu	0.7549
2	DeepMind	0.7519
3	**Synerise AI**	**0.7460**
4	Topology_mag	0.7447
—	Baseline (R-GAT)	0.6949

The solution achieved 0.7460 test accuracy, improving over the R-GAT baseline by 7.4% absolute. Notably, the solution achieved 1st place on the initial leaderboard during the competition before final evaluation. The accuracy gap between third place and the winning Baidu solution (0.89% absolute) reflects the capabilities of iterative message-passing graph neural networks: the top two teams (Baidu and DeepMind) used variants of R-UNIMP and deep message-passing architectures that aggregate multi-hop neighborhood information through multiple propagation rounds, directly exploiting graph structure during training rather than distilling it into precomputed features. The competitive performance of the precomputation-based approach nonetheless demonstrates that density sketches capture substantial graph structure without the associated computational overhead.

Computational efficiency. The efficiency characteristics of the solution merit emphasis. Training a single model required approximately 42 minutes on one GPU with 16GB memory. Inference on the complete test set completed in about 7 minutes. The preprocessing steps (Cleora embedding, EMDE fitting, feature aggregation) required roughly 3 hours but were performed only once and amortized across all training runs and ensemble members.

This efficiency enabled the exploration of 60 ensemble members within practical time constraints. Solutions employing iterative graph neural networks would face substantially higher computational costs, as each training iteration requires neighborhood aggregation across the massive graph. The precomputation-based approach trades off some representational flexibility for dramatic improvements in training efficiency, supporting hypothesis H3 that density-based representations reduce the computational requirements for achieving competitive performance.

7.4 Twitter Engagement Prediction

The ACM RecSys Challenge is an annual competition associated with the Conference on Recommender Systems. The 2021 edition, hosted by Twitter, focused on predicting user engagement at production scale. Unlike most academic benchmarks, this challenge imposed strict computational constraints that reflect real-world deployment requirements: models had to complete inference within tight latency budgets on limited hardware. The solution placed second among all participating teams. The implementation is available at `https://github.com/Synerise/recsys-challenge-2021`.

7.4.1 Task Description

Twitter released approximately one billion data points representing user-tweet interactions over four weeks (February 4 to March 4, 2021). For each user-tweet pair, the task required predicting probabilities for four engagement types: Like, Reply, Retweet, and Quote. The dataset included features describing both the tweet author (engaged user) and the reader (engaging user), along with tweet content represented as BERT token identifiers across 66 languages.

Three aspects distinguished this challenge from standard recommendation benchmarks:

Scale. One billion training examples exceeds typical academic datasets by orders of magnitude, testing whether methods scale to industrial data volumes without prohibitive computational costs.

Latency constraints. Models had to execute on a single CPU core with 64GB memory, completing all test predictions within 24 hours. This translates to approximately 6 milliseconds per prediction, precluding large neural networks, complex inference procedures, or on-the-fly embedding computation.

Fairness evaluation. Metrics were computed separately for tweet authors in five popularity quintiles (based on follower counts), then averaged. This evaluation protocol penalizes models that perform well only for popular authors while neglecting accounts with smaller followings, reflecting Twitter's interest in equitable recommendation quality.

The first three weeks served as training data, with the final week split into validation and test sets. Two weeks before the deadline, validation labels were released for fine-tuning, simulating a production environment where models adapt to recent trends.

7.4.2 Solution Architecture

The latency constraint fundamentally shaped the solution architecture. Any operation requiring more than a few milliseconds per prediction was infeasible, ruling out neural network inference for text embedding, complex graph traversals, or large embedding lookups. The solution achieved accuracy through careful feature engineering within this constraint, with EMDE providing efficient text representation.

Tweet representation with EMDE. Computing BERT embeddings during inference would violate the latency constraint. Instead, a precomputation strategy using EMDE created efficient tweet representations.

First, a DistilBERT model (distilbert-base-multilingual-cased) was fine-tuned on the training data to adapt its representations to the Twitter domain. Using this fine-tuned model, the average embedding for

each token across all tweets in the corpus was computed, producing a fixed embedding vector for each vocabulary token.

EMDE was then fitted on these token embeddings, partitioning the semantic space so that tokens with similar meanings cluster in the same regions. To represent a tweet at inference time, the procedure required only looking up precomputed token sketches (one per token in the tweet) and summing them, followed by L_2 normalization. This aggregation completes in microseconds regardless of tweet length, compared to the hundreds of milliseconds that DistilBERT inference would require.

Interaction features. The majority of predictive signal came from historical engagement patterns between users:

- Engagement counts between the specific user pair across all four reaction types
- Total engagements received by the tweet author (measuring their overall popularity)
- Total engagements given by the engaging user (measuring their activity level)
- Engagements with tweets in the same language as the current tweet
- Engagements with the specific hashtags present in the current tweet

User similarity features. User similarity was computed through follower overlap, measured by Jaccard similarity on follower sets. For each engaging user, engagement counts with users similar to the current tweet author were aggregated. This captures the intuition that users tend to engage with authors similar to those they have engaged with previously.

Community detection on directed graphs of engagement interactions provided additional structural features. Using the Leiden algorithm, Modularity Vertex Partitions were computed separately for each engagement type. Two features were extracted per graph: a binary indicator of whether both users belong to the same partition, and the inverse of partition cardinality when they share a partition. This captures communities of mutual interaction, with larger communities indicating weaker interaction strength.

Numerical feature encoding. All numerical features (interaction counts, follower counts, account age) were encoded using the multi-scale sinusoidal encoding described in Section 5.3.5. This encoding transforms each scalar value into a 16-dimensional vector using sine and cosine functions at eight different scales, providing numerically stable representations without requiring explicit normalization across features with vastly different magnitudes.

Model architecture. The classifier was a three-layer residual feed-forward network with 1,500 hidden units per layer, leaky ReLU activations, and batch normalization. The relatively small architecture was dictated by latency constraints. The network output four probabilities (one per engagement type), trained with binary cross-entropy loss.

Training proceeded in two stages: initial training on the three-week training set (approximately 24 hours on a Tesla V100 GPU), followed by fine-tuning on the released validation set (approximately 60 minutes). This two-stage approach allowed the model to adapt to the temporal distribution of the test data.

Table 7.5: RecSys 2021 Challenge final standings (top 5 teams). AP denotes Average Precision; RCE denotes Relative Cross-Entropy.

Rank	Team	AP Like	AP Reply	AP Retweet	AP Quote	Time
1	NVIDIA	0.7216	0.2649	0.4614	0.0692	23h
2	**Synerise AI**	**0.7046**	**0.2559**	**0.4514**	**0.0662**	**18h**
3	LAYER6 AI	0.6836	0.2490	0.4317	0.0660	13h
4	test_lightgbm	0.6636	0.2118	0.4060	0.0520	5h
5	final1	0.6559	0.2077	0.3940	0.0459	19h

Table 7.6: Feature ablation on the RecSys validation set. Features are added incrementally (+) or removed (-) from the full model.

Configuration	AP Like	AP Reply	AP Retweet	AP Quote
Interaction history only	0.697	0.170	0.405	0.059
+ Tweet content (EMDE sketch)	0.730	0.221	0.427	0.064
+ User account features	0.733	0.222	0.429	0.064
+ Community features	0.734	0.227	0.431	0.066
Full model — EMDE (use averaged embeddings)	0.730	0.213	0.424	0.064

7.4.3 Results

Table 7.5 presents the final competition standings.

The solution achieved second place with strong performance across all engagement types. The winning NVIDIA solution employed GPU-accelerated gradient boosting and neural network ensembles with substantially greater computational resources. The density-based approach achieved comparable accuracy while completing inference in 18 hours, with per-prediction latency of approximately 4 milliseconds, comfortably within the 6-millisecond budget.

Ablation analysis. Table 7.6 isolates the contribution of each feature category.

Interaction history provided the strongest baseline, confirming that past behavior strongly predicts future engagement. Tweet content features improved Reply prediction most substantially (from 0.170 to 0.221 AP), consistent with the intuition that tweet content strongly influences whether users will respond. Account features and community membership provided smaller but consistent gains across all engagement types.

The final row isolates the contribution of density-based aggregation: replacing EMDE sketches with averaged DistilBERT token embeddings reduced AP Reply from 0.227 to 0.213 and AP Retweet from 0.431 to 0.424. The benefit is concentrated in these two engagement types, while Like prediction is essentially unaffected (0.730 in both cases), consistent with Like being driven primarily by interaction history rather than tweet content. This suggests that density sketches carry meaningful signal when content governs the engagement decision, but offer little over averaging when engagement is largely independent of text.

Inference efficiency. A single prediction required approximately 4 milliseconds on CPU, comfortably within the 6-millisecond budget. This efficiency derived from the precomputation strategy: token sketches,

user interaction counts, and community assignments were all computed offline and stored for lookup during inference. At prediction time, the model performed only feature assembly (lookups and summation) and a single forward pass through the small feed-forward network.

7.5 Discussion

The three competitions documented in this chapter span distinct domains, tasks, and computational constraints. This section synthesizes observations across all three experiences, evaluates the evidence they provide for the thesis hypotheses, and discusses the inherent limitations of competition-based validation.

7.5.1 Evidence for Thesis Hypotheses

Hypothesis H1: Density-based representations outperform point-based aggregation. The Booking.com challenge provides the most direct test of this hypothesis through its controlled comparison between EMDE and GRU-based aggregation. With identical embeddings, identical auxiliary features, and identical downstream architecture, EMDE achieved 0.598 Precision@4 compared to 0.588 for GRU, a 1.7% absolute improvement attributable solely to the aggregation method. The sequential information that GRU explicitly models appears to be less informative than the distributional structure that EMDE preserves.

This finding aligns with the structural characteristics of the travel data. The 62% of trips containing repeated cities create cyclic patterns that sequential models struggle to represent. The order in which cities are visited often reflects practical constraints such as flight availability and hotel pricing rather than meaningful preference progression. EMDE's density-based representation sidesteps these issues by focusing on which regions of city-embedding space the traveler has visited and how recently, rather than modeling the specific sequence of transitions between cities.

Hypothesis H2: Density sketching enables effective multimodal integration through simple concatenation. All three solutions demonstrated successful integration of heterogeneous information sources through concatenation of density sketches:

- The Booking.com solution concatenated sketches from two Cleora embedding configurations (iterations $I = 1$ and $I = 3$), capturing complementary views of city similarity at different graph scales.
- The KDD Cup solution concatenated EMDE sketches of text embeddings with Cleora-propagated label vectors and institution affiliation features, integrating semantic, structural, and organizational signals.
- The RecSys solution concatenated tweet content sketches with interaction history features and community membership signals, combining content-based and collaborative information.

In each case, simple concatenation proved effective without requiring learned fusion mechanisms, attention-based weighting, or modality-specific architectural components. The downstream networks learned appropriate weightings implicitly through the training objective.

Table 7.7: Summary of competition-based evidence for thesis hypotheses.

Hypothesis	Verdict	Competition Evidence
H1: Density-based representations outperform point-based aggregation	Supported	Booking.com: EMDE outperformed GRU by 1.7% absolute with identical embeddings and architecture
H2: Multimodal integration through concatenation	Supported	All three solutions integrated heterogeneous signals through sketch concatenation without fusion architectures
H3: Density-based inputs reduce architectural complexity requirements	Supported	KDD Cup: 4-layer MLP placed 3rd on 244M-node graph; 42min training vs. week+ for GNN baselines

Hypothesis H3: Density-based inputs reduce the benefits of architectural complexity. All three solutions employed shallow feed-forward networks (three to four layers) without attention mechanisms, recurrence, or graph convolutions during inference. This architectural simplicity was not merely a choice but a necessity in the RecSys challenge, where latency constraints precluded complex models. Yet even in the unconstrained KDD Cup setting, the feed-forward architecture achieved competitive accuracy against sophisticated graph neural network approaches.

The Booking.com results quantify this effect: EMDE with a three-layer feed-forward network outperformed GRU-based sequential modeling despite the GRU's greater architectural expressiveness. The density sketch representation performs representational work that would otherwise require the network to learn, enabling simpler architectures to achieve equivalent or superior performance.

The KDD Cup result provides perhaps the strongest evidence. A 244-million node heterogeneous graph, precisely the setting for which graph neural networks were designed, yielded to a feed-forward classifier over precomputed features. Training required 42 minutes on a single GPU, compared to the substantially greater computational requirements of iterative graph neural network approaches employed by other teams.

Summary. Table 7.7 summarizes the competition-based evidence for thesis hypotheses. This evidence complements the controlled benchmark experiments in Chapters 5 and 6.

7.5.2 Methodological Patterns

Beyond the specific hypothesis tests, several methodological patterns recurred across all three competitions:

Graph structure captures rich implicit information. In all three domains, graph-derived embeddings captured relationships that were not explicitly provided in the input features. Cleora city embeddings recovered geographic structure from co-visitation patterns alone. Citation graph embeddings captured topical similarity beyond what text embeddings provided. Engagement graph structure revealed community memberships that predicted future interactions.

This pattern suggests that behavioral data contains implicit structure that graph-based methods can extract. The graph need not be explicitly provided; it can be constructed from co-occurrence patterns (cities visited together, papers cited together, users engaging with the same content). Cleora's efficiency makes such construction and embedding practical even at large scales.

Precomputation enables efficiency without sacrificing accuracy. All three solutions separated representation construction from model training, precomputing embeddings and features that remained fixed during training iterations. This separation provides multiple benefits: it enables rapid iteration during model development, it supports ensemble methods by amortizing preprocessing costs, and it satisfies strict latency constraints by moving expensive computation offline.

The KDD Cup solution demonstrates this pattern most clearly. Graph operations, which would dominate computational costs in an end-to-end graph neural network, occurred once during preprocessing. The classifier then operated on fixed features, enabling fast training and inference regardless of graph size.

Simple classifiers suffice when inputs are informative. Complex neural architectures are often justified as necessary to extract subtle patterns from raw or minimally processed inputs. When inputs already encode rich structure through density sketches and graph embeddings, this justification weakens. The feed-forward networks in all three solutions performed a relatively simple function: mapping informative features to predictions. The representational complexity resided in the features themselves.

7.5.3 Limitations of Competition-Based Validation

Competition results must be interpreted with appropriate caution. Several factors limit the conclusions that can be drawn from these experiences.

Evaluation protocol specificity. Each competition defined particular metrics, data splits, and evaluation procedures that may not capture all aspects of real-world performance. The Precision@4 metric in the Booking.com challenge matches their UI design but might not reflect other measures of recommendation quality such as diversity or user satisfaction. Classification accuracy in the KDD Cup treats all misclassifications equally, though confusing closely related categories (cs.LG with cs.AI) is arguably less severe than confusing distant ones (cs.LG with physics.hep-th).

Team composition effects. Competition success reflects team effort, available computational resources, and time invested, not methodology alone. While I led the development of all three solutions, collaboration with colleagues contributed to the results in ways that cannot be cleanly separated from the methodological contributions.

7.5.4 Complementary Validation

Despite these limitations, the competition results provide validation that complements the controlled benchmark experiments in preceding chapters. Benchmarks allow careful ablation and comparison under standardized conditions but may reflect community-wide optimization toward specific datasets.

Competitions test whether methods generalize to unfamiliar settings under time pressure, providing evidence that strong performance reflects genuine methodological contributions rather than benchmark-specific tuning.

The consistency of strong results across three distinct domains, with different data types, different prediction tasks, and different computational constraints, strengthens the evidence that density-based representations capture fundamental properties of behavioral data. If the approach succeeded through chance or domain-specific optimization, performance would vary substantially across settings.

7.6 Summary

This chapter documented the application of density-based methods to three international machine learning competitions in 2021, achieving consistent top-three placements across diverse domains and constraints:

- **Booking.com Data Challenge**: Second place among 40 teams, with Best Paper Award. The direct comparison between EMDE and GRU-based aggregation demonstrated that density-based representations outperform sequential modeling for trip prediction (H1), with identical embeddings and architecture producing 1.7% absolute improvement. The solution integrated multiple embedding configurations through simple concatenation (H2) and achieved competitive results with a three-layer feed-forward network (H3).
- **KDD Cup 2021**: Third place among approximately 150 teams. The solution scaled to 244 million nodes through precomputation-based efficiency, training in 42 minutes per model on a single GPU. The feed-forward classifier over density sketches achieved 7.4% absolute improvement over the graph attention network baseline, demonstrating that density-based representations reduce architectural requirements for competitive performance (H3).
- **ACM RecSys Challenge 2021**: Second place overall. The solution met strict latency constraints (4ms per prediction versus 6ms budget) while achieving strong engagement prediction across author popularity levels. EMDE-based tweet representation outperformed averaged embeddings, confirming the value of density-based aggregation even under severe computational constraints.

The competition results complement the benchmark evaluations in preceding chapters by providing validation under realistic constraints: hidden test sets, strict deadlines, and comparison against diverse competing approaches. The consistency of strong results across domains from travel to academic citations to social media, achieved without domain-specific architectural innovations, supports the thesis that density-based representations capture fundamental properties of behavioral data rather than exploiting benchmark-specific artifacts.

8 Conclusions

This dissertation began with a fundamental question: how should we represent what a user wants? The dominant answer in recommendation research has been to borrow from natural language processing, treating behavioral sequences as sentences and importing architectures designed for linguistic structure. This thesis developed and validated an alternative: representing behavioral histories as probability distributions over learned embedding manifolds. The experimental evidence, accumulated across academic benchmarks, international competitions, and production deployment, demonstrates that this distributional perspective captures structure in behavioral data that sequential and point-based representations systematically destroy.

This final chapter synthesizes the contributions, evaluates the research hypotheses against accumulated evidence, reflects on what the research revealed, acknowledges limitations, and identifies directions for future investigation.

8.1 Summary of Contributions

The primary contribution of this dissertation is a unified framework for behavioral prediction built on density-based representations. The framework comprises two main methodological contributions developed during this doctoral research, which leverage existing graph embedding techniques as foundational infrastructure.

EMDE (Chapter 5), developed collaboratively at Synerise [12], transforms variable-length sets of embeddings into fixed-size density sketches. My contributions included proposing density-dependent partitioning, the key modification that positions hyperplane thresholds at data quantiles rather than at zero, making EMDE a manifold density estimator rather than a standard LSH hash table; the theoretical grounding connecting this to manifold density estimation; experimental design and execution across session-based and top-k recommendation benchmarks; and the analysis of why density-based representations succeed where temporal graph networks fail through the PopTrack diagnostic baseline. This ensures that partitions cut through regions where items actually exist, allocating representational capacity where it matters. Multiple independent partitionings provide resolution that grows multiplicatively while storage grows only linearly. The resulting sketches preserve distributional structure that averaging destroys: a user interested in both electronics and books produces a bimodal profile, not a meaningless centroid between unrelated categories.

BaseModel (Chapter 6) represents the primary original contribution of this dissertation. I conceptualized the foundation model approach for behavioral data, designed the unified architecture supporting

recommendation, classification, and regression tasks, implemented and conducted all experiments, and led the production deployment. The architecture introduces a factorized linear layer (FMLinear) that models pairwise cross-modal interactions within the concatenated sketch through low-rank factorization, yielding consistent 3–6% improvements over standard linear projections. The training objective predicts future behavioral distributions from past distributions. No task-specific labels are required; the supervision comes entirely from the temporal structure of behavioral data itself. The same pretrained backbone then supports diverse downstream tasks through lightweight task-specific heads. Production deployment at a major European retailer achieved 21% higher email engagement compared to an incumbent system built on hand-engineered features.

The framework builds upon **Cleora** (Chapter 4), a scalable hypergraph embedding algorithm developed by colleagues at Synerise. While I did not develop Cleora, its properties (deterministic computation, linear time complexity, and inductive inference for new entities) make it well-suited as the embedding foundation for the density-based methods. Understanding Cleora is essential for understanding how the subsequent contributions operate, which is why it is presented in detail.

Beyond methodology, the dissertation contributes validation through **competition solutions** (Chapter 7) where I led the research and served as first author. The Booking.com Data Challenge (2nd place, Best Paper Award) demonstrated that distributional trip representations outperform sequential models. The KDD Cup (3rd place among approximately 150 teams, behind only Baidu and Google DeepMind) showed that density sketches enable simple architectures to scale to graphs with 244 million nodes. The ACM RecSys Challenge (2nd place, behind only NVIDIA) validated that the approach meets strict real-time latency constraints. Placing consistently among the strongest industrial research laboratories, across domains from travel to academic citations to social media, provides evidence that density-based representations capture something fundamental about behavioral data rather than exploiting properties of any single domain.

8.2 Evaluation of Hypotheses

Chapter 1 introduced five hypotheses that guided the research. Table 8.1 summarizes the evaluation; the following discussion provides detailed assessment.

H1: Density-Based Representations Outperform Point-Based Aggregation. This hypothesis predicted that representing behavioral histories as density sketches would achieve higher accuracy than averaged embeddings, and that performance would degrade when geometric structure is removed from input embeddings.

Density sketches consistently outperformed point-based aggregation across multiple experimental settings, with the mechanism confirmed by ablation. In the session-based recommendation experiments of Chapter 5, EMDE achieved the highest MRR@20 on five of six datasets, outperforming methods that aggregate session items through attention-weighted combinations or graph message passing. The ablation on input embedding quality proved particularly telling: when structured embeddings were replaced with random vectors lacking geometric structure, MRR dropped from 0.365 to 0.279 on RETAIL, a 23% relative

Table 8.1: Summary of hypothesis evaluation across all experimental settings.

ID	Hypothesis	Verdict	Key Evidence
H1	Density-based representations outperform point-based aggregation	Supported	103–138% over HSTU; 23% drop with random embeddings
H2	Multimodal integration through concatenation	Supported	7–32% gains across datasets
H3	Reduced architectural complexity requirements	Supported	3-layer MLP beats GNNs; 42min vs. week+ training
H4	Self-supervised pretraining enables cross-task transfer	Supported	2× faster convergence; +0.3 AUC on classification; 10/12 RelBench tasks (indirect)
H5	Cold-start advantages	Strongly supported	80.7% cold-start items surfaced; 0% without content

decrease. This sharp degradation confirms that the density sketch representation succeeds precisely because it preserves distributional structure over a meaningful embedding manifold.

The Booking.com competition (Chapter 7) provided a direct comparison against sequential aggregation. With identical Cleora embeddings and identical downstream architecture, EMDE achieved 0.598 Precision@4 compared to 0.588 for GRU-based sequential aggregation, a 1.7% absolute improvement from the aggregation method alone.

The most compelling evidence comes from the comparison with Meta's HSTU (Chapter 6), which represents the current industrial frontier for sequential recommendation with 1.5 trillion deployed parameters. On Amazon Books, BaseModel's density-based approach achieved 103% higher HR@10 and 138% higher NDCG@10 than HSTU-large. This result is remarkable given the architectural disparity: HSTU employs pointwise aggregated attention, stochastic length training, relative attention biases, and hierarchical activation management, all innovations specifically designed to make sequential transduction scale to industrial recommendation. BaseModel achieves superior results with a feed-forward network operating on density sketches, without any attention or recurrence.

HSTU's own ablation studies illuminate why the density-based approach succeeds. Their stochastic length technique removes over 80% of sequence tokens while maintaining quality within 0.2% normalized entropy. If the vast majority of a user's interaction history can be discarded without meaningful degradation, the sequential structure that HSTU models so carefully contains less information than the architecture assumes. The density sketch representation captures exactly what remains after this pruning: the distributional pattern of which items were engaged with and how recently, without the computational overhead of attention over positions that prove dispensable.

H2: Density Sketching Enables Effective Multimodal Integration. This hypothesis predicted that different modalities could be integrated through simple concatenation of modality-specific sketches, without requiring learned fusion architectures, and that such integration would yield consistent performance improvements.

Every experimental setting where multimodal data was available showed improvement from concatenated sketches, with no changes to the downstream architecture required. In the session-based experiments, multimodal EMDE outperformed unimodal variants on every dataset where multimodal data was available. On RETAIL, Hit Rate improved from 0.588 to 0.633 when text and metadata sketches were concatenated with interaction sketches, a 7.7% relative improvement requiring no architectural changes.

The RelBench experiments in Chapter 6 extended this finding to relational prediction tasks. On rel-amazon, adding metadata improved mAP by 5 to 13% depending on the task. On rel-avito, where 71% of test items were unseen, metadata improved mAP by 32%. On rel-hm, visual features from a Vision Transformer added 3 to 4% improvement in fashion recommendation. In every case, the integration mechanism was the same: compute sketches independently for each modality, concatenate them, and let the downstream network learn to weight the signals appropriately.

The competition solutions reinforced this pattern. The Booking.com solution concatenated two embedding configurations computed with different iteration counts. The KDD Cup solution combined EMDE text sketches with label-propagation vectors and institutional affiliation features. The RecSys solution integrated tweet content sketches with interaction history and community membership signals. Simple concatenation proved effective across all three domains.

H3: Density-Based Inputs Reduce Architectural Complexity Requirements. This hypothesis predicted that when input representations already capture distributional structure, complex neural architectures would yield diminishing returns, and that shallow feed-forward networks would match or exceed deep sequential models and graph neural networks.

Shallow feed-forward networks over density sketches consistently matched or exceeded graph neural networks and sequential models, though with important nuances. In the session-based experiments, EMDE with a three-layer feed-forward network achieved state-of-the-art MRR on five of six datasets, outperforming GRU4Rec, NARM, SR-GNN, and TAGNN. The graph neural network methods struggled to scale: SR-GNN and TAGNN timed out after over a week of hyperparameter tuning on RSC15 and 30Music, while EMDE trained in under one hour.

The KDD Cup result provides the most striking evidence. A 244-million node heterogeneous graph, precisely the setting for which graph neural networks were designed, yielded to a four-layer feed-forward classifier over precomputed features. Training required 42 minutes on a single GPU. The solution placed third among approximately 150 teams.

However, the hypothesis requires qualification. On rel-stack in the RelBench benchmark, graph neural networks that explicitly model message passing through social network structure outperformed BaseModel's density representation. When tight local community structure dominates and users repeatedly engage with the same threads over years, local neighborhood aggregation provides advantages that global density sketches do not capture. The hypothesis holds most strongly when behavioral patterns are distributional rather than strictly local.

The factorized linear layer ablation (Section 6.9.4) adds a further dimension. Replacing standard linear projections with factorized layers that model pairwise feature interactions yielded consistent 3–6% improvements across rel-amazon tasks, while keeping the architecture feed-forward and computationally lightweight. The implication is that H3 is best understood as a statement about topological complexity: attention, recurrence, and message passing are unnecessary when the input representation is rich enough. But within the feed-forward paradigm, aligning the projection mechanism with the cross-modal structure of concatenated sketches still pays off.

H4: Self-Supervised Pretraining Enables Cross-Task Transfer. This hypothesis predicted that models initialized from a pretrained foundation backbone would converge faster and achieve better final performance than identical architectures trained from scratch, even for tasks structurally different from the pretraining objective.

The convergence benefit was more pronounced than the final-accuracy benefit, but both appeared. On Amazon Beauty recommendation, the pretrained model converged roughly twice as fast and retained a small performance advantage at convergence (17.2 vs. 17.0 Recall@25). On rel-hm churn prediction, a binary classification task structurally different from the recommendation-style pretraining objective, pretraining improved AUC by 0.3 points throughout training. The convergence speed benefit was the more pronounced effect; final-performance gains, while consistent, were modest. Nonetheless, the benefit appeared across both task types, indicating that the foundation stage captures transferable behavioral structure rather than task-specific patterns.

The RelBench results provide complementary evidence of a different kind. A single pretrained backbone with lightweight task-specific heads matched or exceeded specialized graph neural networks on 10 of 12 tasks spanning recommendation, classification, and regression. Because this comparison is against different architectures rather than against the same architecture without pretraining, it demonstrates that the pretrained system generalizes across task types but does not isolate how much of that advantage stems from pretraining versus from the density-based representation itself (tested under H1 and H3). The direct ablation also covered only recommendation and classification; regression tasks lack a pretrained-versus-scratch comparison, leaving that aspect of cross-task transfer supported indirectly. The overall pattern, fast convergence from pretraining combined with strong cross-task performance of the pretrained system, supports the hypothesis while suggesting that pretraining's primary practical contribution is reducing training cost.

H5: Density-Based Representations Excel in Cold-Start Scenarios. This hypothesis predicted that content features would enable non-trivial recommendation performance on previously unseen items through the same inference mechanism used for established items, whereas interaction-only approaches would achieve zero cold-start recall.

On rel-avito, where 71% of test items were unseen during training, adding metadata improved mAP by 32% and surfaced new items at an 80.7% rate among top recommendations, compared to 0% without content features. On rel-amazon, metadata enabled 2.6 to 3.1% cold-start recall where the interaction-only model achieved literally zero. The metadata ablation experiments in Chapter 6 confirmed this pattern across multiple datasets.

The density sketch representation handles cold-start naturally because it operates on embedding space rather than item identities. Any item with content features (text description, image, categorical attributes) receives meaningful bucket codes immediately. These codes position the item in relevant regions of the predicted distribution without approximation or fallback mechanisms. The inference procedure is identical for cold-start and established items; only the source of the embedding differs.

8.3 Reflections on the Research

Beyond the specific technical contributions, this research yielded broader insights worth articulating.

The sequential assumption deserves scrutiny. The dominant framing of behavioral modeling as sequence prediction inherited assumptions from natural language processing without sufficient examination of whether those assumptions transfer. Language has grammar: word order encodes syntactic and semantic structure essential to meaning. Behavioral sequences often lack analogous structure. The order in which a user browses products frequently reflects interface design, search ranking, or chance rather than meaningful preference progression.

The success of density-based representations, which replace explicit item-to-item transition modeling with recency-weighted distributional aggregation, suggests that the field may have over-indexed on sequential modeling. This is not to claim that order never matters: educational progressions and narrative content exhibit genuine sequential dependencies. But the default assumption of sequential structure warrants more critical examination than it typically receives.

Representational choices dominate architectural choices. A recurring pattern across all experiments was that careful input representation reduced the importance of model architecture. Feedforward networks over density sketches consistently matched or exceeded sophisticated architectures operating on less structured inputs. This observation has practical implications: engineering effort may be better invested in representation design than in architectural innovation.

The factorized linear layer illustrates this from a different angle. Concatenated multimodal sketches benefit from factorized pairwise interactions precisely because the input is a concatenation of modality-specific components; the architecture should match the structure of its input, not impose structure of its own.

The pattern also suggests a research direction. If the right representation makes simple models competitive, what other representational insights might be hiding behind architectural complexity in other domains?

Graph structure encodes more than connectivity. Cleora embeddings consistently captured implicit structure not present in explicit features. City embeddings recovered geographic relationships from co-visitation patterns alone. Citation embeddings captured topical similarity beyond text content. This emergence of latent structure from behavioral co-occurrence suggests that graph-based preprocessing deserves more attention as a general technique for enriching behavioral representations.

Academic benchmarks and production deployment test different things. Strong benchmark performance did not automatically translate to production impact, and vice versa. The engagement improvements observed across retail deployments emerged from properties that benchmarks do not measure: robustness to data quality issues, graceful handling of missing features, and stability under distribution shift. The fourfold improvement in churn prediction at the banking deployment, achieved by fine-tuning the same backbone used for recommendation benchmarks, demonstrated transfer learning in a setting where the prediction task, data distribution, and success criteria differed substantially from academic evaluation protocols.

The cross-industry validation proved particularly instructive. Fashion e-commerce, banking, and telecommunications present different data characteristics, regulatory constraints, and business objectives. That the same methodology produced consistent improvements across these settings suggests the density-based representation captures fundamental properties of behavioral data rather than domain-specific artifacts. This breadth of validation, spanning controlled A/B tests, regulated financial deployments, and subscription business models, provides confidence that benchmark results reflect genuine methodological contributions.

This experience suggests that the field would benefit from evaluation protocols that better approximate deployment conditions: latency constraints, missing data, temporal distribution shift, and business-relevant metrics. The production deployments documented in this thesis offer one model for such validation, though the confidentiality requirements of commercial relationships inevitably limit the detail that can be shared.

8.4 Limitations

Honest assessment requires acknowledging what this work does not accomplish and where the methods fall short.

Dataset-specific training. Although BaseModel employs self-supervised pretraining, the foundation stage trains independently for each dataset. Behavioral profiles depend on dataset-specific embeddings, and the foundation model learns dynamics particular to each domain. Cross-dataset transfer, where a model trained on one behavioral domain generalizes to another without retraining, remains unexplored. Privacy constraints that prevent sharing behavioral data across organizations, combined with the semantic heterogeneity of behavioral signals across domains, make this a genuinely difficult challenge rather than a straightforward extension.

Reduced emphasis on sequential structure. EMDE represents interaction histories as weighted sets rather than ordered sequences, with temporal information entering through recency weighting rather than explicit sequential modeling. This design choice works well when the meaningful signal lies in which items a user engaged with and how recently, rather than in the specific order of engagement. However, for domains where sequential dependencies genuinely matter, such as educational progressions where concepts build on prerequisites, narrative content where plot development matters, or multi-step task completion with logical dependencies, the recency-weighted set representation may not capture all

relevant structure. The experiments in this dissertation focused on e-commerce, travel, social media, and citation networks, where temporal order often reflects interface design or logistical constraints rather than meaningful preference progression. The approach may be less suitable for domains with stronger sequential dependencies.

Local structure limitations. The rel-stack results revealed that when tight local community structure dominates, graph neural networks that aggregate neighborhood signals outperform the global density representation. The sketches summarize behavior across the entire manifold and may dilute strong local patterns. A user who interacts exclusively within a small, dense community might be better served by methods that emphasize local connectivity over distributional breadth.

Random partitions. EMDE uses random hyperplane partitions with density-dependent thresholds. While this works well empirically, learned partitions that adapt to the prediction task might improve performance. End-to-end training of sketch parameters is technically feasible but would complicate the pipeline, sacrifice the efficiency benefits of precomputed bucket codes, and potentially introduce optimization challenges. Whether the gains would justify these costs remains unknown.

8.5 Future Directions

Several research directions emerge naturally from this work.

Cross-domain foundation models. The foundation model paradigm in NLP succeeds partly because language has universal vocabulary: words mean approximately the same thing across corpora. Behavioral data lacks this property, since item and user identifiers are platform-specific. Yet the underlying patterns of preference evolution may share structure across domains. A user exploring a new hobby exhibits similar behavioral dynamics whether the hobby involves books, sports equipment, or musical instruments. Developing representations that capture these domain-invariant patterns while respecting privacy constraints presents both theoretical and practical challenges worth pursuing.

Hybrid local-global representations. The limitations on rel-stack suggest value in combining density sketches with local neighborhood features. A user's profile might include both a global density sketch capturing their distributional preferences and local neighborhood information capturing their position in specific communities. The challenge lies in integrating these complementary views without sacrificing the efficiency and simplicity that make density sketches practical.

Learned sketch partitions. Replacing random hyperplanes with learned partitions could improve discrimination in regions where random partitions happen to be suboptimal. Differentiable relaxations of the discrete bucket assignment might enable end-to-end training. The research question is whether the potential gains justify the added complexity and computational cost.

Temporal hierarchy. Users exhibit preferences at multiple timescales: immediate context (what they are shopping for right now), medium-term interests (hobbies they are actively pursuing), and long-term tastes (stable preferences that persist over years). A hierarchical temporal model that maintains separate density representations at different scales might capture these dynamics more effectively than the current single-scale approach.

Causal inference. The current framework predicts future behavior from past behavior, but prediction differs from causal understanding. If a user's history contains both purchases and ad exposures, predicting their next purchase does not reveal whether the ads caused the purchase. Extending density-based representations to support causal inference, identifying which interventions would actually change behavior, could substantially expand their practical utility.

Integration with large language models. Recent advances in language models open possibilities for systems that combine density-based behavioral representations with natural language understanding of explicit preferences expressed through conversation. Such hybrid systems could leverage the complementary strengths of implicit behavioral signals and explicit preference articulation.

8.6 Closing Remarks

This dissertation was conducted as an industrial doctorate, and that context shaped the research in important ways. Access to production-scale data enabled validation that academic benchmarks alone could not provide. The requirement to deploy models under strict latency constraints forced attention to computational efficiency from the outset. The opportunity to measure impact through business metrics rather than proxy measures provided evidence that offline experiments cannot establish.

The methods developed here now power production recommendation systems serving major European retailers. The codebase has been open-sourced to enable reproduction and extension. The competition solutions placed consistently in the top three against hundreds of competing teams from leading research organizations worldwide. The research has been commercialized as BaseModel.ai, providing validation of deployment requirements that academic benchmarks cannot address: infrastructure flexibility, production-scale latency, and regulatory compliance.

Yet the most important outcome may be conceptual rather than technical. The density estimation perspective on behavioral data, treating interaction histories as recency-weighted distributions over learned manifolds rather than as explicit sequences of item transitions or single aggregated points, offers a different way of thinking about what recommendation systems should model. Whether this perspective ultimately proves more productive than alternatives is a question for future research to answer. What this dissertation demonstrates is that the perspective is viable, practical, and worthy of further investigation.

The gap identified in Chapter 1 between the apparent maturity of recommender systems and their practical limitations remains wide. Users still encounter irrelevant suggestions, filter bubbles, and recommendation failures that simple heuristics would avoid. Closing this gap will require advances across many fronts: better evaluation protocols, more thoughtful handling of fairness and diversity, deeper integration

of causal reasoning. The contribution of this work is more modest: one alternative representation, carefully validated, that captures something about behavioral data that other representations miss.

That something, the distributional structure of human preferences, turns out to matter more than the sequential structure that has dominated recent research. This finding, if it generalizes beyond the settings studied here, suggests a reorientation of how the field approaches behavioral modeling. Not as sequence prediction borrowed from language, but as density estimation suited to the actual structure of how humans express preferences through their actions.

Bibliography

[1] Himan Abdollahpouri, Masoud Mansoury, Robin Burke, and Bamshad Mobasher. The unfairness of popularity bias in recommendation. In *Proceedings of the Workshop on Recommendation in Multi-stakeholder Environments (RMSE), RecSys*, 2019.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[3] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(2):423–443, 2019.

[4] Bank BNP Paribas. Bank BNP Paribas wdrożył rozwiązanie AI do dokładniejszej personalizacji oferty dla klientów, 2024. URL `https://bank.pl/bank-bnp-paribas-wdrozyl-rozwiazanie-ai-do-dokladniejszej-personalizacji-oferty-dla-klientow/`.

[5] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003. doi: 10.1162/089976603321780317.

[6] Boston Consulting Group. Capturing the $2 trillion personalization opportunity with AI. BCG Press Release, October 2024.

[7] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[8] Shaosheng Cao, Wei Lu, and Qiongkai Xu. GraRep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 891–900, 2015. doi: 10.1145/2806416.2806512.

[9] Allison JB Chaney, Brandon M Stewart, and Barbara E Engelhardt. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 224–232, 2018.

[10] Tianlang Chen, Charilaos Kanatsoulis, and Jure Leskovec. RelGNN: Composite message passing for relational deep learning. In *Forty-second International Conference on Machine Learning*, 2025.

[11] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[12] Jacek Dąbrowski, Barbara Rychalska, Michał Daniluk, Dominika Basaj, Konrad Gołuchowski, Piotr Bąbel, Andrzej Michałowski, and Adam Jakubowski. An efficient manifold density estimator for all recommendation systems. In *International Conference on Neural Information Processing*, pages 323–337. Springer, 2021.

[13] Michał Daniluk and Jacek Dąbrowski. Temporal graph models fail to capture global temporal dynamics. In *Temporal Graph Learning Workshop at NeurIPS*, 2023.

[14] Michał Daniluk, Tim Rocktäschel, Johannes Welbl, and Sebastian Riedel. Frustratingly short attention spans in neural language modeling. In *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[15] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, pages 271–280, 2007.

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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)*, pages 4171–4186, 2019.

[17] dunnhumby. dunnhumby and Synerise form strategic AI partnership, 2025. URL **https://www.dunnhumby.com/about-us/news/dunnhumby-and-synerise-form-strategic-ai-partnership/**.

[18] European Investment Bank. EIB group ramps up support for technological innovation in Poland, 2024. URL **https://www.eib.org/en/press/all/2025-455-grupa-europejskiego-banku-inwestycyjnego-znaczaco-zwieksza-wsparcie-dla-rozwoju-innowacji-technologicznych-w-polsce**.

[19] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 101–109, 2019. doi: 10.1145/3298689.3347058.

[20] Chen Gao, Xiang Wang, Xiangnan He, and Yong Li. Self-supervised learning for recommendation. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pages 5136–5139, 2022.

[21] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315, 2022.

[22] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems*, 6(4):1–19, 2015.

[23] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[24] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[25] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in neural information processing systems*, pages 1024–1034, 2017.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[27] Ruining He and Julian McAuley. VBPR: Visual Bayesian personalized ranking from implicit feedback. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, pages 144–150, 2016.

[28] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182, 2017.

[29] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 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*, pages 639–648, 2020.

[30] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

[31] Shenyang Huang, Farimah Poursafaei, Jacob Danovitch, Matthias Fey, Weihua Hu, Emanuele Rossi, Jure Leskovec, Michael M. Bronstein, Guillaume Rabusseau, and Reihaneh Rabbany. Temporal graph benchmark for machine learning on temporal graphs. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[32] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998.

[33] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[34] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[35] Anton Klenitskiy and Alexey Vasilev. Turning dross into gold loss: Is BERT4Rec really better than SASRec? In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1120–1125, 2023.

[36] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[37] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM Conference on Information and Knowledge Management*, pages 1419–1428, 2017.

[38] Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, pages 689–698, 2018.

[39] Chengkai Liu, Jianghao Chen, Jianxin Wu, Weinan Zhang, Yong Yu, and Jun Wang. Mamba4rec: Towards efficient sequential recommendation with selective state space models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1874–1884, 2024.

[40] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. STAMP: Short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1831–1839, 2018. doi: 10.1145/3219819.3219950.

[41] Malte Ludewig and Dietmar Jannach. Evaluation of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 28(4-5):331–390, 2019.

[42] Fei Mi and Boi Faltings. Context tree for adaptive session-based recommendation. *arXiv preprint arXiv:1806.03733*, 2018.

[43] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26, pages 3111–3119, 2013.

[44] Neal Mohan. Remarks at CES 2018, 2018. YouTube Chief Product Officer keynote, Consumer Electronics Show, Las Vegas.

[45] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G Azzolini, et al. Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*, 2019.

[46] Xia Ning and George Karypis. Slim: Sparse linear methods for top-n recommender systems. In *2011 IEEE 11th International Conference on Data Mining*, pages 497–506. IEEE, 2011.

[47] Bibek Paudel, Fabian Christoffel, Chris Newell, and Abraham Bernstein. Updatable, accurate, diverse, and scalable recommendations for interactive applications. In *ACM Transactions on Interactive Intelligent Systems (TiiS)*, volume 7, pages 1–34, 2017.

[48] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*, pages 325–341. Springer, 2007.

[49] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[50] Aleksandr Petrov and Craig Macdonald. A systematic review and replicability study of BERT4Rec for sequential recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 436–447, 2022.

[51] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[52] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315, 2023.

[53] Steffen Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.

[54] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[55] Marco Rossetti, Fabio Stella, and Markus Zanker. Contrasting offline and online results when evaluating recommendation algorithms. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 31–34, 2016.

[56] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[57] Barbara Rychalska, Piotr Bąbel, Konrad Gołuchowski, Andrzej Michałowski, Jacek Dąbrowski, and Przemysław Biecek. Cleora: A simple, strong and scalable graph embedding scheme. In *International Conference on Neural Information Processing*, pages 338–352. Springer, 2021.

[58] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, pages 285–295, 2001.

[59] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260, 2002.

[60] Guy Shani, David Heckerman, and Ronen I Brafman. An MDP-based recommender system. volume 6, pages 1265–1295, 2005.

[61] Harald Steck. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*, pages 3251–3257, 2019. doi: 10.1145/3308558.3313710.

[62] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1441–1450, 2019.

[63] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Xinghua Qu, Jie Zhang, and Cong Geng. Are we evaluating rigorously? benchmarking recommendation for reproducible evaluation and fair comparison. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 23–32, 2020.

[64] Synerise. Data-driven customer experience in mBank, 2023. URL **https://www.synerise.com/case-studies/mbank**.

[65] Synerise. Orange - digital channels optimization, 2023. URL **https://www.synerise.com/case-studies/orange**.

[66] Synerise. BaseModel AI boosts customer retention accuracy x4 in finance industry, 2024. URL **https://www.synerise.com/case-study/bnp-paribas**.

[67] Synerise. BaseModel.ai launches native app on Snowflake marketplace, 2024. URL **https://www.synerise.com/blog/basemodel-ai-launches-native-app-on-snowflake-marketplace**.

[68] Synerise. Synerise strengthens growth strategy with new $8.5 million investment in series B+ round, 2024. URL **https://www.synerise.com/blog/synerise-strengthens-growth-strategy**.

[69] Synerise AI Team. EMDE illustrated. Synerise AI Research Blog, 2022. URL **https://sair.synerise.com/emde-illustrated/**. Accessed: 2026-03-01.

[70] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077, 2015. doi: 10.1145/2736277.2741093.

[71] Ke Tran, Arianna Bisazza, and Christof Monz. Recurrent memory networks for language modeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 321–331, San Diego, California, 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1036.

[72] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. DyRep: Learning representations over dynamic graphs. In *International Conference on Learning Representations*, 2019.

[73] Vojtech Vancura, Pavel Kordík, and Milan Straka. beeFormer: Bridging the gap between semantic and interaction similarity in recommender systems. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 1102–1107, 2024. doi: 10.1145/3640457.3691707.

[74] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

[75] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. DropoutNet: Addressing cold start in recommender systems. In *Advances in Neural Information Processing Systems*, volume 30, pages 4957–4966, 2017.

[76] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 950–958, 2019.

[77] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 165–174, 2019.

[78] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. MMGCN: Multi-modal graph convolution network for personalized recommendation of micro-video. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1437–1445, 2019.

[79] Jason Weston, Ron Weiss, and Hector Yee. Nonlinear latent factorization by embedding multiple user interests. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 65–68, 2013.

[80] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. Self-supervised graph learning for recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 726–735, 2021.

[81] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

[82] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01): 346–353, 2019.

[83] Yaochen Yang, Bowen Li, Xiang Wang, and Xiangnan He. Do we really need knowledge graphs for recommender systems? a critical re-evaluation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 112–121, 2024.

[84] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 974–983, 2018.

[85] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Tagnn: Target attentive graph neural networks for session-based recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1921–1924, 2020.

[86] Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 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*, pages 1294–1303, 2022.

[87] Yiwen Yuan, Zecheng Zhang, Xinwei He, Akihiro Nitta, Weihua Hu, Manan Shah, Blaz Stojanovic, Shenyang Huang, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. ContextGNN: Beyond two-tower recommendation systems. In *International Conference on Learning Representations*, 2025.

[88] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Fangda Gu, Michael He, Yinghai Lu, et al. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. *Proceedings of the 41st International Conference on Machine Learning*, 2024.

[89] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

[90] Hongyu Zhou, Xin Zhou, Zhiwei Zeng, Lingzi Zhang, and Zhiqi Shen. A comprehensive survey on multimodal recommender systems: Taxonomy, evaluation, and future directions. *arXiv preprint arXiv:2302.04473*, 2023.