A GEOMETRIC APPROACH TO PERSONALIZED RECOMMENDATION WITH SET-THEORETIC CONSTRAINTS USING BOX EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

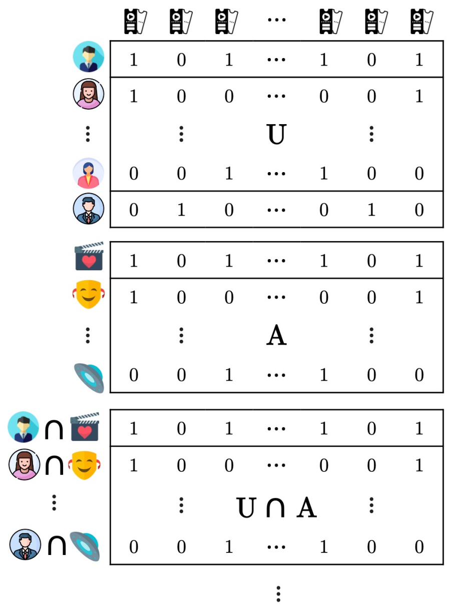
Personalized item recommendation typically suffers from data sparsity, which is most often addressed by learning vector representations of users and items via low-rank matrix factorization. While this effectively densifies the matrix by assuming users and movies can be represented by linearly dependent latent features, it does not capture more complicated interactions. For example, vector representations struggle with set-theoretic relationships, such as negation and intersection, *e.g.* recommending a movie that is "comedy and action, but not romance". In this work, we formulate the problem of personalized item recommendation as matrix completion where rows are *set-theoretically dependent*. To capture this set-theoretic dependence we represent each user and attribute by a hyper-rectangle or *box* (*i.e.* a Cartesian product of intervals). Box embeddings can intuitively be understood as trainable Venn diagrams, and thus not only inherently represent similarity (via the Jaccard index), but also naturally and faithfully support arbitrary set-theoretic relationships. Queries involving set-theoretic constraints can be efficiently computed directly on the embedding space by performing geometric operations on the representations. We empirically demonstrate the superiority of box embeddings over vector-based neural methods on both simple and complex item recommendation queries by up to 30% overall.

1 INTRODUCTION

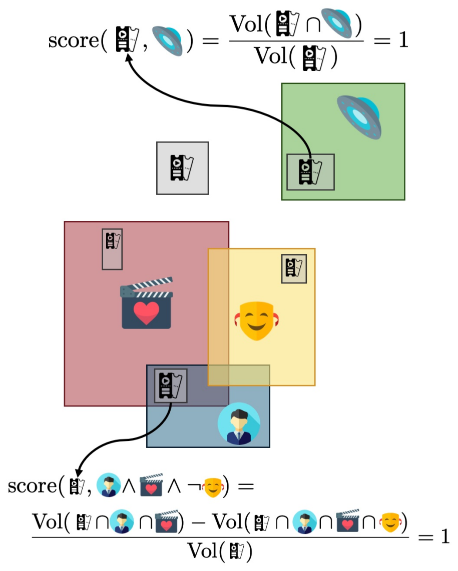
Recommendation systems are a standard component of most online platforms, providing personalized suggestions for products, movies, articles, and more. In addition to generic recommendation, these platforms often present the option for the user to search for items, either via natural language or structured queries. While collaborative filtering methods like matrix factorization have proven successful in addressing data sparsity for unconditional generation, they often fall short when attempting to combine them with more complicated queries. This is not unexpected, as vector embeddings, while effectively capturing linear relationships, are ill-equipped to handle the complex set-theoretic relationships. Even advanced neural network-based approaches, which are designed to capture intricate relationships, have been shown to struggle with set-theoretic compositionally that underlie many real-world preferences.

Let us consider an example where a user named Bob wants to watch a comedy which is not a romantic comedy. Assuming we have a prior watch history for users, standard collaborative filtering techniques (e.g. low-rank matrix factorization) would yield a learned score function $\mathrm{score}(m, \mathtt{Bob})$ for each movie m. If we also have movie-attribute annotations, we could form the set of comedies C and set of romance movies R and simply filter to those movies in $C \setminus R$, however this assumes that the movie-attribute annotations are complete, which is rarely the case.

A standard approach in a setting with sparse data is to learn a low-rank approximation for the attribute $\times$ movie matrix $\mathbf{A}$, yielding a dense matrix $\hat{\mathbf{A}}$. We can then form sets of movies based on this dense matrix using an (attribute-specific) threshold, *e.g.* $\hat{C} \coloneqq \{m \mid \hat{A}_{\mathtt{comedy},m} > \tau_{\mathtt{comedy}}\}$ and $\hat{R} \coloneqq \{m \mid \hat{A}_{\mathtt{romance},m} > \tau_{\mathtt{romance}}\}$, and then rank movies $m \in \hat{C} \setminus \hat{R}$ according to $\mathrm{score}(m, \mathtt{Bob})$. While this approach does allow for performing the sort of queries we are after, it suffers from three fundamental issues:



(a) Standard matrix completion assumes you are given partial information about the user $\times$ movie matrix $\mathbf{U}$, and potentially incomplete information about the attribute $\times$ movie matrix $\mathbf{A}$, and asks you to recover any unobserved entries. The task of set-theoretic matrix completion extends this to being able to predict the entries of arbitrary set-theoretic combinations of these rows.



(b) Box embeddings represent the movies, users, and attributes as "boxes" (Cartesian products of intervals) in $\mathbb{R}^n$. The score for a specific movie in relation to a given query is determined by the proportion of the movie box's volume that falls within the corresponding region. During training, this membership score for a movie, w.r.t the U and A are optimized, creating a set-geometric representation of the matrix.

Figure 1: Set-theoretic matrix completion for movies, users, and attributes, illustrating how box embeddings, trained in a set-theoretic manner, address this task.

1. Limited user-attribute interaction: Since the attribute classification is done independently from the user, any latent relationships between the user and attribute cannot be taken into account.
2. Error compounding: Errors in the completion of attribute sets accumulate as the number of sets involved in the query increase.
3. Mismatched inductive-bias: Our queries can be viewed as set-theoretic combinations of the rows, not linear combinations. As such, using a low-rank approximation of the matrix may be misaligned with the eventual use.

In this paper, we formulate the problem of attribute-specific recommendation as matrix completion where rows are not necessarily *linear combinations* of each other but, rather, are *set-theoretic combinations* of each other. More precisely, given some user $\times$ movie interaction matrix $\mathbf{U}$ and attribute $\times$ movie matrix $\mathbf{A}$, the queries we are considering are set-theoretic combinations of these rows (see Figure 1a). For example, the ground-truth data for comedies which are not romance movies which Bob likes would be the vector $x \in \{0,1\}^{|M|}$, where $x_m = 1$ if and only if $\mathbf{U}_{\texttt{Bob},m} = 1$ and $\mathbf{A}_{\texttt{comedy},m} = 1$ and $\mathbf{A}_{\texttt{romance},m} = 0$. Note that this is not a linear combination of the previous rows, and so while the inductive bias of low-rank factorization has proven immensely effective for collaborative filtering we should not expect it to be directly applicable in this setting.

Instead, we propose to learn representations for the users and attributes that are consistent with specific set-theoretic axioms. These representations must also be compactly parameterizable in a lower-dimensional space, differentiable with respect to some appropriate score function, and allow for efficient computation of various set operations. Box Embeddings (Vilnis et al., 2018; Dasgupta et al., 2020), which are axis-parallel n-dimensional hyperrectangles, meet these criteria (see Figure 1b). The volume of a box is easily calculated as the product of its side-lengths. Furthermore, box embeddings are closed under intersection (*i.e.* the intersection of two boxes is another box). Inclusion-exclusion thus allows us to calculate the volume of arbitrary set-theoretic combinations of boxes.

The contributions of our paper are as follows -

1. We model the problem of attribute-specific query recommendation as "set-theoretic matrix completion", where attributes and users are treated as sets of items. We discuss the challenges faced by existing machine-learning approaches for this problem setup.
2. We demonstrate the inconsistency of existing vector embedding models for this task. Additionally, we establish box embeddings as a suitable embedding method for addressing such set-theoretic problems.
3. We conduct an extensive empirical study comparing various vector and box embedding models for the task of set-theoretic query recommendation.

Box embeddings, with their geometric set operations, significantly outperform all vector-based methods. We also evaluate score multiplication and threshold-based prediction for both vector and box embedding models, and find that performing set operations directly on the box embeddings performs best, solidifying our claim that the inductive bias of box embeddings provides the necessary generalization capabilities to address set-theoretic queries.

2 Task Formulation

2.1 Background

Matrix completion is a fundamental problem in machine learning, and arises in a wide array of tasks, from recommender systems to image reconstruction. Formally, this problem is typically modeled as follows: Given a matrix $X \in \mathbb{R}^{m \times n}$ where only a subset of the entries are observed, find a complete matrix $\hat{X} \in \mathbb{R}^{m \times n}$ which closely approximates X on the observed entries. For the task of recommendation, this involves predicting user interactions with items they have not previously interacted with, and a common assumption is that the preferences of users and characteristics of the items can be expressed by a small number of latent factors, with the alignment of these latent factors captured via dot-product. This justifies the search for a low-rank approximation $\hat{X} = BC$, where $B \in \mathbb{R}^{m \times D}$ and $C \in \mathbb{R}^{D \times n}$. In the case where the original matrix is binary, $X \in \{0,1\}^{m \times n}$, it is common to perform *logistic matrix factorization*, where an elementwise sigmoid is applied after the dot-product of latent factors, which we denote (with slight abuse of notation) as $\hat{X} = \sigma(BC)$.
Many recent advanced methods for this matrix completion task utilize sophisticated neural networks (He et al., 2017; 2020; Liang et al., 2018), that capture more expressive interactions among users and items. However, they are not particularly recognized for their set-theoretic compositionality, as previously described.

2.2 Set-Theoretic Matrix Completion

We will describe the task of set-theoretic matrix completion on the setting of movies, users, and attributes, though the formulation and our proposed model can be generalized to arbitrary domains. We are given a set $\mathcal{D}_U \subseteq U \times M$ of user-movie interactions, and a set $\mathcal{D}_A \subseteq A \times M$ of attribute-movie pairs. We assume both of these sets are incomplete.

Our goal is to eventually be able to recommend movies based on some query, for example "comedy and not romance". Such a query for a particular user can be represented as $u \wedge a_1 \wedge \neg a_2$, where u is the user, $a_1 = \texttt{comedy}$ and $a_2 = \texttt{romance}$. We let Q be the set of all queries of interest, which depends on which queries we anticipate evaluating at inference time. In this work, we will take Q to be queries of the form u, a_1, $u \wedge a_1$, $u \wedge a_1 \wedge a_2$, and $u \wedge a_1 \wedge \neg a_2$, where $u \in U$ and $a_1, a_2 \in A$.

With this formulation, we can view our task as matrix completion for a matrix $X \in \{0,1\}^{|Q| \times |M|}$, where the rows are derived by applying bitwise operators on the rows of user and attribute data. While we could, in theory, proceed directly with logistic matrix factorization on this matrix, there are both practical and theoretical reasons to search for an alternative. First, the number of rows of this matrix is very large relative to the original data - in our case we have $|Q| = \mathcal{O}(|U||A|^2)$, but in general $|Q| = \mathcal{O}(3^{|U||A|})$. This poses practical issues, both at training time (as there are an exponential number of elements of X to traverse) and inference time (storing the low-rank approximations requires $\mathcal{O}(|Q|)$ memory, which is much larger than $|U| + |A|$). There are also theoretical issues with the underlying assumption, as it is no longer reasonable to assume the rows of $\sigma^{-1}(X)$ are linear combinations of some latent factors.

3 Method

Our proposed solution to address these issues starts by defining the sets of movies which comprise the queries of interest. Let, $\mathcal{P}(M)$ be the power set of movies M. Specifically, for each user u we can define the set $M_u = \{m \mid (u, m) \in \mathcal{D}_U\}$, and for each attribute a we can define the set $M_a = \{m \mid (a, m) \in \mathcal{D}_A\}$. If we let $\mathcal{M} \subseteq \mathcal{P}(M)$ be the collection of all such sets, then the set of movies corresponding to a given query q are direct set-theoretic combinations of elements in $\mathcal{M}$. Hence, the reasonable underlying assumption, in this case, is to model the elements of $\mathcal{M}$ as sets via a map $f : \mathcal{M} \to R$ where R is also a set of sets, and the map f respects set-theoretic operations, *i.e.* $f(S \cap T) = f(S) \cap f(T)$ and $f(S \setminus T) = f(S) \setminus f(T)$, etc. Such a map is referred to as a *homomorphism of Boolean algebras*, and the problem of learning such a function was explored in general in Boratko et al. (2022), wherein the authors proposed the use of box embeddings as the function f which is then trained to obey the homomorphism constraints.

3.1 Set-theoretic Representation Box Embeddings

As introduced in Vilnis et al. (2018), box embeddings represent entities by a hyperrectangle in $\mathbb{R}^D$, *i.e.* a Cartesian product of intervals. Let the box embedding for user u be:

$$\mathrm{Box}(u) = \prod_{d=1}^{D} [u^{\llcorner}_d, u^{\urcorner}_d] = [u^{\llcorner}_1, u^{\urcorner}_1] \times \ldots \times [u^{\llcorner}_D, u^{\urcorner}_D] \subseteq \mathbb{R}^D,$$

where $[u^{\llcorner}_d, u^{\urcorner}_d]$ is the interval for d-th dimension, $u^{\llcorner}_d < u^{\urcorner}_d$ for $d \in \{1, \ldots, D\}$.
The volume of an interval is defined as the length of the interval $\mathrm{Vol}((u^{\llcorner}_d, u^{\urcorner}_d)) = \max(u^{\urcorner}_d - u^{\llcorner}_d, 0)$. Let, $\mathrm{Box}(m) = \prod_{d=1}^{D} [m^{\llcorner}_d, m^{\urcorner}_d]$ be the box embeddings for a movie m. At dimension d, the volume of intersection between user u and movie m is defined as -

$$\mathrm{VolInt}((u^{\llcorner}_d, u^{\urcorner}_d), (m^{\llcorner}_d, m^{\urcorner}_d)) = \max(\min(u^{\urcorner}_d, m^{\urcorner}_d) - \max(u^{\llcorner}_d, m^{\llcorner}_d), 0)$$

. When the movie interval $[m^{\llcorner}_d, m^{\urcorner}_d]$ is completely contained by user interval $[u^{\llcorner}_d, u^{\urcorner}_d]$, then $\frac{\mathrm{VolInt}((u^{\llcorner}_d, u^{\urcorner}_d), (m^{\llcorner}_d, m^{\urcorner}_d))}{\mathrm{Vol}((u^{\llcorner}_d, u^{\urcorner}_d))} = 1$. This objective creates a set-theoretic interpretation with box embeddings, where user $\mathrm{Box}(u)$ contains all the movie boxes related to u (fig. 1b). The score for containment for a single dimension d is formulated as:

$$F_{\mathrm{Box}}((u^{\llcorner}_d, u^{\urcorner}_d), (m^{\llcorner}_d, m^{\urcorner}_d)) := \frac{\mathrm{VolInt}((u^{\llcorner}_d, u^{\urcorner}_d), (m^{\llcorner}_d, m^{\urcorner}_d))}{\mathrm{Vol}((u^{\llcorner}_d, u^{\urcorner}_d))} := \frac{\max(\min(u^{\urcorner}_d, m^{\urcorner}_d) - \max(u^{\llcorner}_d, m^{\llcorner}_d), 0)}{\max(u^{\urcorner}_d - m^{\llcorner}_d, 0)}.$$

The overall containment score is the multiplication of F_{Box} for each dimension. The log of this score is referred to as the energy function as given:

$$\mathrm{E}_{\mathrm{Box}}(u, m) := -\log \prod_{d=1}^{D} F_{\mathrm{Box}}((u^{\llcorner}_d, u^{\urcorner}_d), (m^{\llcorner}_d, m^{\urcorner}_d)). \tag{1}$$

This energy function is minimized when the user $\mathrm{Box}(u)$ contains the movie $\mathrm{Box}(m)$. Previous works have highlighted the difficulty of optimizing an objective including these hard $\min$ and $\max$ functions (Li et al., 2019; Dasgupta et al., 2020). In our work, we use the latter solution, termed GumbelBox, which treats the endpoints $x^{\llcorner}$ and $x^{\urcorner}$ as mean of GumbelMax and GumbelMin random variables, respectively. Given 1-dimensional box parameters $\{[x^{\llcorner}_n, x^{\urcorner}_n]\}_{n=1}^{N}$, we define the associated GumbelMax random variables $X^{\llcorner}_n$ with mean $x^{\llcorner}_n$ and scale β, as well as the GumbelMin random variables $X_n^\urcorner$ with mean $x_n^\urcorner$ and scale β. Dasgupta et al. (2020) calculates that the expected volume of intersection of intervals $\{[X_n^\llcorner, X_n^\urcorner]\}$ can be approximated by

$$\mathbb{E}\left[\max\left(\min_n X_n^\urcorner - \max_n X_n^\llcorner, 0\right)\right] \approx \mathrm{LSE}_\beta(\mathrm{LSE}_{-\beta}(x_1^\urcorner, \dots, x_N^\urcorner) - \mathrm{LSE}_\beta(x_1^\llcorner, \dots, x_N^\llcorner), 0),$$

essentially replacing the hard $\min$ and $\max$ operators with a smooth approximation, $\mathrm{LSE}_t(\mathbf{x}) \coloneqq t\log(\sum_i e^{x_i/t})$. Expected intersection volume in higher dimensions is just a product of the preceding equation, as the random variables are independent. We use this GUMBELBOX (abbrev GB) formulation in our work changing the notations F_{Box}, Vol, VolInt to F_{GB}, Vol_{GB}, VolInt_{GB}. We modify the per-dimension score function F_{Box} in equation 1 by replacing the ratio of hard volume calculations with the approximation to the expected volume,

$$F_{\mathrm{GB}}((u_d^\llcorner, u_d^\urcorner), (m_d^\llcorner, m_d^\urcorner); (\tau, \nu)) \coloneqq \frac{\mathrm{LSE}_\nu(\mathrm{LSE}_{-\tau}(u_d^\urcorner, m_d^\urcorner) - \mathrm{LSE}_\tau(u_d^\llcorner, m_d^\llcorner), 0)}{\mathrm{LSE}_\nu(m_d^\urcorner - m_d^\llcorner, 0)} \tag{2}$$

$$\eqqcolon \frac{\mathrm{VolInt}_{\mathrm{GB}}((u_d^\llcorner, u_d^\urcorner), (m_d^\llcorner, m_d^\urcorner); (\tau, \nu))}{\mathrm{Vol}_{\mathrm{GB}}((m_d^\urcorner - m_d^\llcorner); \nu)}. \tag{3}$$

For more details and alternative approaches see the related work in Appendix A.

3.2 Training

We model each user, attribute, and movie as a box in $\mathbb{R}^D$, and denote the map from these entities to their associated box parameters as θ, i.e., the trainable box embedding for user u is $\theta(u) \coloneqq \mathrm{Box}(u)$. Our goal is to train these box representations to represent certain sets of movies which allow us to perform the sort of queries we are interested in. As motivated above, for a given user u, we train $\mathrm{Box}(u)$ to approximate the set M_u via a noise-contrastive estimation objective. Namely, for each $(u, m) \in \mathcal{D}_U$, we have a loss term

$$\ell_{(u,m)}(\theta) \coloneqq \mathrm{E}_{\mathrm{GB}}(u, m; \theta) - \mathbb{E}_{\tilde{m}\sim M}\left[\log(1 - \exp(-\mathrm{E}_{\mathrm{GB}}(u, \tilde{m}; \theta))\right].$$

The first term is minimized when $\mathrm{Box}(u)$ contains $\mathrm{Box}(m)$. We approximate the second term via sampling, which encourages $\mathrm{Box}(u)$ to be disjoint from $\mathrm{Box}(\widetilde{m})$ for a uniformly randomly sampled movie $\widetilde{m}$. We define an analogous loss function $\ell_{(a,m)}(\theta)$ for attribute-movie interactions, which trains $\mathrm{Box}(a)$ to contain the box $\mathrm{Box}(m)$ for each m such that $(u, m) \in \mathcal{D}_U$.

The overall loss function is a convex combination of these loss terms:

$$\mathcal{L}(\theta; \mathcal{D}_U, \mathcal{D}_A) \coloneqq w \sum_{(u,m)\in\mathcal{D}_U} \ell_{(u,m)}(\theta) + (1 - w) \sum_{(a,m)\in\mathcal{D}_A} \ell_{(a,m)}(\theta)$$

for a hyperparameter $w \in [0, 1]$. This optimization ensures that the movie boxes are contained within the corresponding user and attribute boxes, thereby establishing a set-theoretic inductive bias. Both numbers of negative samples and w are hyperparameters for training (Please Refer to section 4, appendix B.2) for further details.

3.3 Inference

During inference, given the trained embedding model θ and a user u we determine the user's preference for the movie m by negating and exponentiating the energy function,

$$\mathrm{score}(m, u; \theta) \coloneqq \exp\left(-\mathrm{E}_{\mathrm{GB}}(u, m; \theta)\right) = \prod_{d=1}^{D} F_{\mathrm{GB}}(\theta(u)_d, \theta(m)_d; (\tau, \nu)) \in \mathbb{R}_{\geq 0},$$

where $\theta(x)_d = (x_d^\llcorner, x_d^\urcorner)$. Since the calculation is simply a product over dimensions, for notational clarity we will restrict our discussion for more complex queries to the one-dimensional case, and omit the explicit dependence on temperature hyperparameters, so

$$\mathrm{score}(m, u; \theta) \coloneqq \frac{\mathrm{VolInt}_{\mathrm{GB}}\left(\theta(m), \theta(u)\right)}{\mathrm{Vol}_{\mathrm{GB}}\left(\theta(m)\right))}$$

which is the *proportion of* $\theta(m)$ *which is contained within* $\theta(u)$ (see Figure 1b). It achieves it's maximum at 1 if $\theta(u)$ contains $\theta(m)$, and is minimized at 0 when they are disjoint, corresponding to the motivation that $\theta(u)$ represents the set of movies that user u has interacted with.

Given a query with a conjunction between attributes (*e.g.* "comedy and action") we denote the attributes involved a_1 and a_2. Similarly to the score for a single user query, we define the score for these attributes as the proportion of the movie box $\theta(m)$ which is contained inside of the (soft) intersection of boxes $\theta(u)$, $\theta(a_1)$, and $\theta(a_2)$, *i.e.*

$$\text{score}(m, u \wedge a_1 \wedge a_2; \theta) := \frac{\text{VolInt}_{\text{GB}}\left(\theta(m), \theta(u), \theta(a_1), \theta(a_2)\right)}{\text{Vol}_{\text{GB}}\left(\theta(m)\right))}$$

Again, this score is maximized if $\theta(m)$ is contained inside $\theta(u), \theta(a_1)$, and $\theta(a_2)$, and minimized when it is disjoint.

In order to address queries with set differences, recall that, given two measurable sets S and T, we can compute the volume of $S \setminus T$ as $\text{Vol}(S \setminus T) = \text{Vol}(S) - \text{Vol}(S \cap T)$. Thus, if the query involves a negated attribute (*e.g.* "comedy and not action"), we define

$$\text{score}(m, u \wedge a_1 \wedge \neg a_2; \theta) := \frac{\text{VolInt}_{\text{GB}}\left(\theta(m), \theta(u), \theta(a_1)\right) - \text{VolInt}_{\text{GB}}\left(\theta(m), \theta(u), \theta(a_1), \theta(a_2)\right)}{\text{Vol}_{\text{GB}}\left(\theta(m)\right))}$$

This score is maximized when $\theta(m)$ is contained inside $\theta(u)$ and $\theta(a_1)$ while being disjoint from $\theta(a_2)$, and decreases when these conditions are not met.

4 Experiments

In our experiments, we evaluate all the models on item recommendation across three domains: movies, songs, and restaurants. (4.1). We systematically generate queries of varying complexity from these datasets to evaluate performance on set-theoretic tasks (4.2.1, 4.2.2). We train and select models based on the performance of the traditional personalized item prediction (4.3). Finally, we demonstrate that our set-based representation method is better suited for handling set-theoretic constraints in recommendation tasks (5.1, 5.2).

4.1 Dataset

The datasets used in our study must contain two primary components: **Item-User interactions** $\mathcal{D}_U$ and **Item-Attribute interactions** $\mathcal{D}_A$. We select datasets that offer rich ground truth annotations for both components. We utilize the MovieLens 1M and 20M datasets for personalized movie recommendations (Harper & Konstan, 2015). For the song domain, we employ a subset of the Last-FM dataset, which is the official song tag dataset of the Million Song Dataset (Bertin-Mahieux et al., 2011). In the restaurant domain, we use the NYC-R dataset introduced by Wang et al. (2018).

We utilize the data curated by Dasgupta et al. (2023) to construct $\mathcal{D}_A$ for the Movielens data. This dataset employs Wikidata (Vrandečić & Krötzsch, 2014) to generate ground truth attribute labels for movies[1]. For the Last-FM dataset, the authors use the Last.fm API ('getTopTags')[2] to create attribute tags. Likewise, the authors in Wang et al. (2018) crawl restaurant review data from TripAdvisor[3] to curate tags and ratings for restaurants in NYC. The sparsity of D_A and D_U is comparable in the Movielens datasets. In contrast, the Last.fm and NYC-R datasets, designed with tag annotations in mind, exhibit much denser attribute-movie interaction. Thus, the selection of these three datasets not only encompasses diverse domains but also offers varying ground-truth distributions for our experiments.

We use the binarized implicit feedback data Hu et al. (2008), indicating whether the user or the attribute has been associated with the specific item. To ensure the quality of the data, we retain users/items with 5 or more interactions and attributes with frequency 20 or more in all the datasets. Refer to Table 1 for a detailed description of the dataset statistics.

[1]https://github.com/google-research-datasets/genre2movies

[2]https://www.last.fm/

[3]https://www.tripadvisor.com

Table 1: Dataset Statistics, the Item-User interaction $\mathcal{D}_U$ & the Item-Attribute interaction $\mathcal{D}_A$. The Train/Test split is created using algorithm 1 to test set-theoretic generalization.

Dataset	#Users	#Items	#Attributes	#Train $\mathcal{D}_U$	#Eval $\mathcal{D}_U$	#Train $\mathcal{D}_A$	#Eval $\mathcal{D}_A$
Last-FM	1,872	2417	490	60,497	8,857	34,374	4,240
NYC-R	9,597	3764	579	82,734	8,502	34,908	4,376
MovieLens 1M	6,040	3,705	57	963,554	36,655	10,273	1,545
MovieLens-20M	138,493	26,744	95	19,722,646	277,617	80,178	1,734

4.2 Dataset Splits & Query Generation

To select models for each method, we train on a dataset split D_U^{train} & D_A^{train} while evaluating on a held-out set D_U^{eval} & D_A^{eval}. However, we use these eval set pairs to construct compositional queries. Simple random sampling or leave-one-out data splits do not ensure a substantial number of these queries. Therefore, we devise a data splitting technique closely linked to query generation, which we discuss next.

4.2.1 Personalized Simple Query

This type of query corresponds to a single attribute for a particular user, *e.g. Bob wants to watch a comedy movie.* More formally, given a user u and an attribute a, the query type would be - $u \cap a$. Note that, these simple queries are set-theoretic combinations between the item sets corresponding to the users and the attributes. Let us denote the data corresponding to these queries as $Q_{U \cap A}$.

While constructing the $Q_{U \cap A}$ pairs we need to ensure that - if an item is held out for evaluation for a simple query, the individual user-item and attribute-item pair should belong to the evaluation set as well. More formally, $(u, a, i) \in Q_{U \cap A} \iff (u, i) \in \mathcal{D}_U^{\text{eval}} \wedge (a, i) \in \mathcal{D}_A^{\text{eval}}$. To ensure this train/test isolation, we use the sampling algorithm 1 that takes in D_U and D_A and outputs $Q_{U \cap A}$, $\mathcal{D}_U^{\text{train}}, \mathcal{D}_A^{\text{train}}, \mathcal{D}_U^{\text{eval}}, \mathcal{D}_A^{\text{eval}}$ (Refer to Appendix B.1 for more details). The detailed statistics for the splits are provided in Table 1. Also, the statistics for the $Q_{U \cap A}$ are present in Table 2

4.2.2 Personalized Complex Query

The set-theoretic compositions that we consider here are the intersection and negation of attributes for a particular user. Given a user u and attributes a_1 and a_2, we consider the query types- $u \cap a_1 \cap a_2$ and $u \cap a_1 \cap \neg a_2$, e.g, *Bob want to watch an Action Comedy movie, Alice want to watch a Children but not Monster movie*. Creating meaningful attribute compositions requires careful consideration, as not all combinations make sense. For instance, 'Sci-Fi' & 'Documentary' might not be a meaningful combination, whereas 'Sci-Fi' & 'Time-Travel' is. Similarly, 'Sci-Fi' ¬' Fiction' doesn't make sense, but 'Fiction' ¬ 'Sci-Fi' does. Sometimes, even if the intersection is valid, it could be trivial and non-interesting, e.g., 'Fiction' & 'Sci-Fi'.
Intuitively, for two attributes a_1 & a_2, their intersection is interesting if $|a_1 \cap a_2|$ is greater than combining any two random items set. Also, for their intersection to be non-trivial the size of the intersection $|a_1 \cap a_2|$ must be less than the individual sizes of the attributes i.e., $\alpha|a_1|$ and $\alpha|a_2|$. Here,$|.|$ denotes the size of the item set corresponding to the attributes. $\alpha \in [0, 1]$ is a design parameter, dedicated after manual inspection of the quality of the item sets for the combinations . In case of difference queries such as $a_1 \cap \neg a_2$, we consider $\neg a_2$ to be the second attribute and carry out the same filtering strategy as done for the intersection queries.
We denote the set of non-trivial and viable attribute pairs for the intersection to be $\mathcal{A}_\cap = \{(a_1, a_2) | |a_1 \cap a_2| > \epsilon, |a_1 \cap a_2| < \alpha|a_1|, |a_1 \cap a_2| < \alpha|a_2|\}$, and for the difference to be $\mathcal{A}_\setminus = \{(a_1, a_2) | |a_1 \cap \neg a_2| > \epsilon, |a_1 \cap \neg a_2| < \alpha|a_1|, |a_1 \cap \neg a_2| < \alpha|\neg a_2|\}$. Using the above formulation, we generate the test set for the personalized complex queries $Q_{U \cap A_1 \cap A_2}$ and $Q_{U \cap A_1 \cap \neg A_2}$ using algorithm 2. Please refer to Table 2 for the detailed statistics.

4.3 Training Details & Evaluation Criteria

We train all the methods on users and attributes jointly using $\mathcal{D}^{\text{train}} = \mathcal{D}_U^{\text{train}} \cup \mathcal{D}_A^{\text{train}}$. We use dimensions $d = 128$ for vector-based models, and $d = 64$ for box models so that the number of parameters per user, attribute, and movie is equal.[4] We perform extensive hyperparameter tuning for the learning rate, batch size, volume and intersection temperature of boxes, loss combination constant, etc. Please refer to the Appendix B.2 for details. We follow the standard sampled evaluation procedure described in Rendle et al. (2020), only for model selection purpose. For each user-item tuple (u, m) in $\mathcal{D}_U^{\text{eval}}$, the model ranks m amongst a set of items consisting of the m together with 100 other true negative items w.r.t the user. Then we report on two different evaluation metrics namely Hit Ratio@k (HR@k) and NDCG. (a) HitRatio@k: If the rank of m is less than or equals to k then the value of HR@k is 1 or 0 otherwise. (2) NDCG: if r is the rank of m, then $1/\log(r+1)$ is the NDCG.

The model is selected based on the best-performing model on NDCG for the item prediction over the user-item validation set $\mathcal{D}_U^{\text{eval}}$, with the best-performing checkpoint saved for further evaluation on compositional queries. We follow the same evaluation protocol for the compositional queries as well, except, we rank m amongst all items in the vocabulary rather than a sampled subset.

4.4 Baselines

The recommendation systems literature offers a wide range of methods that represent users, and items in $\mathbb{R}^d$. These methods then propose a compatibility score function between the user and item, $\phi : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. A common and effective choice for ϕ is the dot product, which underpins matrix factorization (Rendle et al., 2020; Koren & Bell, 2015). To capture more complex interactions among users, items, and attributes, He et al. (2017) extend matrix factorization by replacing the dot product with a neural network-based similarity function. This method, called Neural Matrix Factorization (NeuMF), combines the dot product with an MLP. Similarly, He et al. (2020) propose LightGCN (LGCN) to captures the user, items, and attribute interaction using Graph Convolution Network Kipf & Welling (2017) over a joint graph of user-item-attribute. We use MF, and, NeuMF LGCN as our baselines.

For a personalized query, be it simple or complex, we need to devise a method to combine the individual scores of the user and the attributes involved in the query. In this work, we compare three approaches to obtain an aggregated score:

1. Filter: In this approach, we retrieve a list of items corresponding to the attributes based on the scores provided by the embedding models. The list is generated by thresholding the scores, where the threshold is optimized by minimizing the F1 score between the training data and predicted scores. We refer to the methods using this aggregation technique as Box-Filter for box embeddings and MF-Filter, NeuMF-Filter, LGCN-Filterfor vector-based methods.
2. Product: In this method, the compositional score is computed by multiplying the scores for the individual queries. For vector-based embeddings, the scores for each movie related to a user or attribute are normalized using the *sigmoid* function. For box embeddings, the energy function is normalized by conditioning on the movie box volume (see Section 3.3). The score for negation is calculated by subtracting the normalized score from 1. The three methods using this technique are referred to as Box-Product, MF-Product, NeuMF-Product, and LGCN-Product.
3. Geometric: This approach leverages the geometry of the embedding space. For vector-based embeddings, learned through Matrix Factorization, addition, and subtraction are often used for query composition (Mikolov et al., 2013). Box embeddings, on the other hand, naturally represent intersection operations, allowing us to compute scores for any set-theoretic combination using box intersection and inclusion-exclusion principles. We refer to these methods as Box-Geometric and MF-Geometric.

5 Results

After conducting an extensive hyper-parameter search on D_U^{eval}, we select the best-performing model for each method based on NDCG scores. This ensures the chosen model is optimal for set-theoretic query inference. The results for the best models for all the methods on all datasets are reported in 3.

[4]Recall that box embeddings are parameterized with two vectors, one for each min and max coordinate.

Table 2: Compositional Query Statistics

Dataset	Personalized Simple Query $u \cap a$	Personalized Complex Query $u \cap a_1 \cap a_2$	$u \cap a_1 \cap \neg a_2$
Last-FM	9,867	45,142	10,814
NYC-R	9,482	7,460	2,369
ML-1M	21,392	51,299	37,769
ML-20M	35,368	42,355	47,374

Table 3: Test NDCG on D_U^{eval} for selected models.

Dataset	MF	NeuMF	LGCN	Box
Last-FM	0.51	0.52	0.56	0.65
NYC-R	0.31	0.33	0.37	0.39
ML-1M	0.51	0.53	0.55	0.58
ML-20M	0.71	0.70	-	0.73

5.1 Set-Theoretic Generalization

We test the selected models for each method with the curated set-theoretic personalized queries (Detailed stats for the queries in Table 2). We report the ranking performance in terms of Hit Rates at 10, 20, and 50. Please refer to 4 for the results.

Table 4: Hit Rate(%)↑ on Set-theoretic queries for datasets Last-FM, MovieLens 1M, NYC-R.

Methods	$U \cap A$			$U \cap A_1 \cap A_2$			$U \cap A_1 \cap \neg A_2$		
	h@10	h@20	h@50	h@10	h@20	h@50	h@10	h@20	h@50
				Last-FM					
MF-Filter	14.8	25.1	37.4	26.8	46.8	62.8	15.2	24.4	35.5
MF-Product	9.0	21.7	48.0	14.3	36.8	73.2	4.8	14.8	43.4
MF-Geometric	6.1	12.2	29.7	3.4	7.6	27.5	1.7	4.8	15.9
NeuMF-Filter	13.5	21.9	32.3	20.0	19.6	55.7	11.3	18.8	28.7
NeuMF-Product	13.6	25.6	47.6	19.5	35.7	63.3	9.0	16.8	40.5
LGCN-Filter	20.4	28.5	39.1	$\underline{42.4}$	54.2	67.4	15.8	21.5	27.6
LGCN-Product	20.5	31.0	48.6	**43.8**	$\underline{58.0}$	80.7	0.8	1.3	3.5
Box-Filter	22.9	31.5	39.0	32.7	46.5	55.9	**22.0**	32.1	40.3
Box-Product	$\underline{27.9}$	$\underline{44.5}$	$\underline{68.0}$	38.2	57.7	$\underline{82.7}$	$\underline{17.8}$	$\underline{32.4}$	**60.3**
Box-Geometric	**28.3**	**44.8**	**68.3**	38.8	**58.3**	**83.1**	17.5	**32.5**	$\underline{60.0}$
				MovieLens-1M					
MF-Filter	5.0	10.2	22.3	11.4	17.9	27.5	4.7	9.8	22.5
MF-Product	4.3	8.5	20.4	5.1	10.6	26.1	3.4	7.3	19.3
MF-Geometric	0.4	0.9	3.0	0.1	0.2	0.8	0.5	1.0	2.7
NeuMF-Filter	9.3	15.5	28.5	13.3	21.5	35.9	8.8	14.7	26.7
NeuMF-Product	10.3	16.8	31.4	$\underline{15.3}$	$\underline{24.5}$	$\underline{43.5}$	5.7	9.7	20.2
LGCN-Filter	8.2	12.3	20.9	11.4	15.6	24.0	9.9	13.8	21.9
LGCN-Product	5.9	9.0	14.9	7.6	11.7	20.1	5.5	8.6	14.1
Box-Filter	**11.7**	**19.1**	$\underline{32.3}$	14.5	20.5	28.6	**11.4**	**19.5**	**34.0**
Box-Product	9.95	16.7	$\underline{31.5}$	10.6	17.8	34.2	$\underline{8.9}$	15.1	29.4
Box-Geometric	$\underline{11.0}$	$\underline{18.3}$	**34.2**	**16.9**	**26.6**	**46.1**	8.6	$\underline{15.2}$	$\underline{31.0}$
				NYC-R					
MF-Filter	1.4	2.4	4.6	2.7	4.8	8.0	2.1	3.5	6.3
MF-Product	1.1	2.9	8.6	3.7	8.2	23.3	8.9	13.1	17.6
MF-Geometric	0.5	1.5	4.3	0.2	0.8	3.5	0.5	1.2	3.7
NeuMF-Filter	3.8	5.6	9.2	2.5	3.2	4.5	4.2	6.3	10.8
NeuMF-Product	4.6	7.3	13.7	6.6	11.2	20.8	2.7	5.2	11.2
LGCN-Filter	4.8	7.8	17.2	$\underline{12.7}$	16.9	21.8	5.4	8.6	16.4
LGCN-Product	5.0	$\underline{8.7}$	**18.1**	12.1	17.6	35.1	4.9	8.0	13.2
Box-Filter	4.9	7.8	13.4	9.9	13.5	20.4	4.4	7.1	12.5
Box-Product	**5.0**	**8.9**	**17.9**	$\underline{10.9}$	$\underline{19.5}$	$\underline{37.3}$	$\underline{5.3}$	$\underline{9.1}$	$\underline{18.8}$
Box-Geometric	$\underline{4.9}$	$\underline{8.7}$	$\underline{17.6}$	**12.2**	**21.5**	**39.2**	**5.5**	**9.2**	**19.2**

The Box Embedding-based method outperforms vector-based methods by a significant margin, showing on average 30% improvement when comparing the aggregated HR@50 performance of the best vector model (MF-PRODUCT/NeuMF-PRODUCT/LGCN-FILTER) to the box model (BOX-GEOMETRIC) across all the three different domains.

The $U \cap A_1 \cap A_2$ query is the most challenging, as it requires accuracy in all three individual queries. For this difficult query, BOX-GEOMETRIC shows the largest performance gap compared to other methods. Additionally, using vector addition and subtraction as geometric proxies for intersection and difference performs significantly worse than all other vector-based methods, while geometric operations in the box embedding space outperform even other box embedding methods. This validates the set-theoretic inductive bias of box embeddings and confirms that geometric operations in this space provide valid set-theoretic operations, unlike vectors.

The FILTER aggregation technique performs similarly to or better than other methods only for Hits@10. However, as k increases, its performance declines across all model types (Box, MF, NeuMF) and datasets. This observation highlights the limitation of a fixed threshold filter and advocates smoother aggregation techniques like PRODUCT and GEOMETRIC.

5.2 SPECTRUM OF GENERALIZATION

The query generation process (refer Section 4.2.1) ensures that for the target item m corresponding to a query involving user u and attribute a, the pair (u, m) and (a, m) must not be in the training set $(u, m) \notin \mathcal{D}_U^{\text{train}}$ and $(a, m) \notin \mathcal{D}_A^{\text{train}}$. The set-theoretic evaluation weakens when such pairs are added back to the training set. There are three different weakening settings applicable here, which we refer to as a spectrum – WEAKEST GENERALIZATION ($(u, m) \in \mathcal{D}_U^{\text{train}}$ and $(a, m) \in \mathcal{D}_A^{\text{train}}$), WEAK GENERALIZATION-USER ($(u, m) \in \mathcal{D}_U^{\text{eval}}$ and $(a, m) \notin \mathcal{D}_A^{\text{train}}$), WEAK GENERALIZATION-ATTRIBUTE ($(u, m) \notin \mathcal{D}_U^{\text{train}}$ and $(a, m) \in \mathcal{D}_A^{\text{eval}}$). We report HitRate@50 performance on query type $U \cap A_1 \cap A_2$ for the MovieLens-1M dataset in Table 5 (More query types in Appendix - Table 9, 8). The weaker the generalization setting the easier it is for the models to achieve higher performance on the test set. Indeed, we observe that this is true across all the methods w.r.t each of the aggregation settings, validating the correctness of the trained models.
However, we are interested in observing the performance gap when we go from the weakest to the strongest set-theoretic generalization. We refer to the percentage gap *Generalization Spectrum Gap* (hr(Weakest) - hr(Set-theoretic) / hr(Weakest) %). From Table 5 we observe that the best-performing box model BOX-GEOMETRIC achieves the best *Generalization Spectrum Gap* for HR@50.

Table 5: *Generalization Spectrum Gap* for PERSONALIZED COMPLEX QUERY $U \cap A_1 \cap A_2$

Methods	Hit Rate @50 ↑				*Spectrum Gap* ↓
	Weakest (W)	Weak-User (W-U)	Weak-Attribute (W-A)	Set-Theoretic (S)	(W − S) / W
MF-FILTER	55.2	41.9	30.5	27.5	50.2%
MF-PRODUCT	**67.4**	38.5	39.3	26.1	61.2 %
MF-GEOMETRIC	18.5	12.9	1.8	0.8	95.6%
NEUMF-FILTER	48.4	33.1	40.4	35.9	38.5%
NEUMF-PRODUCT	67.8	48.7	40.6	43.5	35.9%
BOX-FILTER	52.7	44.5	30.3	28.5	45.9%
BOX-PRODUCT	64.6	52.8	39.0	34.2	47.1%
BOX-GEOMETRIC	62.6	53.3	50.1	**46.1**	**26.4%**

6 CONCLUSION

In this work we presented the task of personalized recommendation with set-theoretic queries. We discussed how this problem can be viewed as set-theoretic matrix completion, and why the common approach of logistic matrix factorization is not aligned with the set-theoretic operations we wish to perform at inference time. We observed substantial improvements over the vector/neural baselines when using box embeddings as the representation, validating our intuition regarding the necessary set-theoretic bias. Our empirical results confirm that box embeddings are ideally suited to the task of recommendation with set-theoretic queries.

REFERENCES

Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, Oct. 2011. doi: 10.1609/aimag.v32i3.2364. URL `https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2364`.

Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawlat, Gautam Das, and Cong Yu. Group recommendation: semantics and efficiency. *Proc. VLDB Endow.*, 2(1):754–765, August 2009. ISSN 2150-8097. doi: 10.14778/1687627.1687713. URL `https://doi.org/10.14778/1687627.1687713`.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495, 2018.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Michael Boratko, Dhruvesh Patel, Shib Sankar Dasgupta, and Andrew McCallum. Measure-theoretic set representation learning. preprint from `https://www.mboratko.com/mtsrl.pdf`, 2022.

Lei Chen, Jie Cao, Youquan Wang, Weichao Liang, and Guixiang Zhu. Multi-view graph attention network for travel recommendation. *Expert Systems with Applications*, 191:116234, 2022. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2021.116234. URL `https://www.sciencedirect.com/science/article/pii/S0957417421015402`.

Shib Dasgupta, Andrew McCallum, Steffen Rendle, and Li Zhang. Answering compositional queries with set-theoretic embeddings, 2023.

Shib Sankar Dasgupta, Michael Boratko, Dongxu Zhang, Luke Vilnis, Xiang Lorraine Li, and Andrew McCallum. Improving local identifiability in probabilistic box embeddings. In *Advances in Neural Information Processing Systems*, 2020.

Yashar Deldjoo, Maurizio Ferrari Dacrema, Mihai Gabriel Constantin, Hamid Eghbal-Zadeh, Stefano Cereda, Markus Schedl, Bogdan Ionescu, and Paolo Cremonesi. Movie genome: alleviating new item cold start in movie recommendation. *User Modeling and User-Adapted Interaction*, 29 (2):291–343, April 2019. ISSN 0924-1868. doi: 10.1007/s11257-019-09221-y. URL `https://doi.org/10.1007/s11257-019-09221-y`.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015. ISSN 2160-6455. doi: 10.1145/2827872. URL `http://doi.acm.org/10.1145/2827872`.

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*, WWW '17, pp. 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee. ISBN 9781450349130. doi: 10.1145/3038912.3052569. URL `https://doi.org/10.1145/3038912.3052569`.

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*, SIGIR '20, pp. 639–648, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401063. URL `https://doi.org/10.1145/3397271.3401063`.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pp. 263–272, 2008.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Yehuda Koren and Robert Bell. *Advances in Collaborative Filtering*, pp. 77–118. Springer US, Boston, MA, 2015. ISBN 978-1-4899-7637-6. doi: 10.1007/978-1-4899-7637-6_3. URL https://doi.org/10.1007/978-1-4899-7637-6_3.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. *Advances in neural information processing systems*, 27, 2014.

Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. Smoothing the geometry of probabilistic box embeddings. *ICLR*, 2019.

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*, WWW '18, pp. 689–698, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356398. doi: 10.1145/3178876.3186150. URL https://doi.org/10.1145/3178876.3186150.

Tingting Liang, Yuanqing Zhang, Qianhui Di, Congying Xia, Youhuizi Li, and Yuyu Yin. Contrastive box embedding for collaborative reasoning. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 38–47, 2023.

Chaitanya Malaviya, Peter Shaw, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Quest: A retrieval dataset of entity-seeking queries with implicit set operations, 2023. URL https://arxiv.org/abs/2305.11694.

Lang Mei, Jiaxin Mao, Gang Guo, and Ji-Rong Wen. Learning probabilistic box embeddings for effective and efficient ranking. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 473–482, New York, NY, USA, 2022a. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512073. URL https://doi.org/10.1145/3485447.3512073.

Lang Mei, Jiaxin Mao, Gang Guo, and Ji-Rong Wen. Learning probabilistic box embeddings for effective and efficient ranking. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pp. 473–482, New York, NY, USA, 2022b. Association for Computing Machinery. ISBN 9781450390965. doi: 10.1145/3485447.3512073. URL https://doi.org/10.1145/3485447.3512073.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Hongyu Ren, Weihua Hu, and Jure Leskovec. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. In *8th International Conference on Learning Representations*. OpenReview.net, 2020.

Steffen Rendle, Walid Krichene, Li Zhang, and John Anderson. Neural collaborative filtering vs. matrix factorization revisited. In *Proceedings of the 14th ACM Conference on Recommender Systems*, RecSys '20, pp. 240–248, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450375832. doi: 10.1145/3383313.3412488. URL https://doi.org/10.1145/3383313.3412488.

Anna Rogers, Aleksandr Drozd, and Bofang Li. The (too many) problems of analogical reasoning with word vectors. In *Proceedings of the 6th Joint Conference on Lexical and Computational Semantics (*SEM 2017)*, pp. 135–148, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/S17-1017. URL `https://aclanthology.org/S17-1017`.

Haitian Sun, Andrew O. Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W. Cohen. Faithful embeddings for knowledge base queries. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020a. Curran Associates Inc. ISBN 9781713829546.

Haitian Sun, Andrew O Arnold, Tania Bedrax-Weiss, Fernando Pereira, and William W Cohen. Guessing what's plausible but remembering what's true: Accurate neural reasoning for question-answering. 2020b.

Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. Probabilistic embedding of knowledge graphs with box lattice measures. In *Association for Computational Linguistics*, 2018.

Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pp. 1543–1552, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356398. doi: 10.1145/3178876.3186066. URL `https://doi.org/10.1145/3178876.3186066`.

Cheng Wu, Shaoyun Shi, Chaokun Wang, Ziyang Liu, Wang Peng, Wenjin Wu, Dongying Kong, Han Li, and Kun Gai. Enhancing recommendation accuracy and diversity with box embedding: A universal framework. In *Proceedings of the ACM on Web Conference 2024*, WWW '24, pp. 3756–3766, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400701719. doi: 10.1145/3589334.3645577. URL `https://doi.org/10.1145/3589334.3645577`.

Le Wu, Yonghui Yang, Kun Zhang, Richang Hong, Yanjie Fu, and Meng Wang. Joint item recommendation and attribute inference: An adaptive graph convolutional network approach. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pp. 679–688, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450380164. doi: 10.1145/3397271.3401144. URL `https://doi.org/10.1145/3397271.3401144`.

Yikun Xian, Tong Zhao, Jin Li, Jim Chan, Andrey Kan, Jun Ma, Xin Luna Dong, Christos Faloutsos, George Karypis, S. Muthukrishnan, and Yongfeng Zhang. Ex3: Explainable attribute-aware item-set recommendations. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, pp. 484–494, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384582. doi: 10.1145/3460231.3474240. URL `https://doi.org/10.1145/3460231.3474240`.

Zezhong Xu, Yincen Qu, Wen Zhang, Lei Liang, and Hua zeng Chen. Inbox: Recommendation with knowledge graph using interest box embedding. *ArXiv*, abs/2403.12649, 2024. URL `https://api.semanticscholar.org/CorpusID:268532286`.

Shuai Zhang, Huoyu Liu, Aston Zhang, Yue Hu, Ce Zhang, Yumeng Li, Tanchao Zhu, Shaojian He, and Wenwu Ou. Learning user representations with hypercuboids for recommender systems. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, pp. 716–724, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450382977. doi: 10.1145/3437963.3441768. URL `https://doi.org/10.1145/3437963.3441768`.

A Related Work

A.1 Box Embeddings

Vilnis et al. (2018) also proposed a probabilistic interpretation of box embeddings, where the volume of a box could be trained to represent the probability of an associated binary random variable, and intersections of boxes would be trained to capture the joint probabilities. Some of the recent works have tried to incorporate box embeddings in a recommendation systems setup.Xu et al. (2024); Wu et al. (2024); Zhang et al. (2021) use the side-length of the box embeddings as a preference range to obtain diverse set recommendations for users, Mei et al. (2022a) utilizes the axis parallel nature of the box embeddings for faster retrieval for recommendation systems. Sun et al. (2020a;b); Ren et al. (2020) are some of the recent works that focus on logical query over knowledge bases (KB).Liang et al. (2023) uses Ren et al. (2020) and add a volumetric contrastive loss as a regularizer. However, in this work, we frame collaborative filtering as a set-theoretic matrix completion problem, which helps us to achieve better generalization for the composition of personalized queries.

A.2 Context Aware Recommendation

The concept of context-aware recommendation, as introduced in Adomavicius et al. (2011), provides a general framework where "context" is broadly defined as any auxiliary information. This framework emphasizes that user preferences for items can vary based on the context in which interactions occur, reflecting a user-centric view of contextual information.
Building on this foundation, recent works have explored specific instances of context-aware recommendation, such as "attribute-aware recommendation." These approaches often leverage item or user attributes as contextual information to address various goals, including improving user profiling (Adomavicius et al., 2011), predicting missing item attributes (Wu et al., 2020; Chen et al., 2022), enhancing recommendations for cold-start scenarios(Deldjoo et al., 2019), or providing attribute-based explanations for recommendations (Xian et al., 2021).

Our work differs significantly in its focus and objectives. we term "attribute-constrained recommendation," which involves generating recommendations explicitly constrained by logical combinations of attributes. Unlike attribute-aware approaches, which aim to improve recommendation quality by incorporating attribute information as auxiliary data, our work directly targets the task of satisfying explicit attribute-based constraints posed by users.

A.3 Compositional Queries with Vector Embeddings

It is common in machine learning to represent discrete entities such as items or attributes by vectors Bengio et al. (2013) and to learn them by fitting the training data. Besides semantic similarity, some have claimed that learned vectors have compositional properties through vector arithmetic, for example in the empirical analysis of word2vec Mikolov et al. (2013) and GLOVE Pennington et al. (2014), and some theoretical analysis Levy & Goldberg (2014); Arora et al. (2018). However, anecdotally, many have found that the compositional behavior of vectors is far from reliable Rogers et al. (2017). Our paper provides a comprehensive evaluation of vector embeddings on compositional queries and compares the results to a region-based alternative.

A.4 Set-based queries in Search and group recommendation systems.

While set-theoretic queries are commonplace in search, popular question-answering (QA) benchmarks often do not include them. We found QUEST (Malaviya et al., 2023) the most closely related study, introducing a benchmark for entity-seeking queries with implicit set-based semantics. However, QUEST does not focus on explicit constraints or personalization, which are central to our work. Additionally, related studies in group recommendation systems (Amer-Yahia et al., 2009) touch on explicit constraint-based personalization, where preferences of multiple users are explicitly aggregated into a coherent recommendation.

B Experiment Details

B.1 Data Splits & Query Generation

Algorithm 1 Personalised Simple Query ($u \cap a$) generation algorithm $u \cap a$

```
 1: Let the set of users, attributes, and movies be $\mathcal{U}, \mathcal{A}, \mathcal{M}$
 2: Marginal probability of an attribute $a$ in $A$, $P(a) = \sum_m A_{a,m} / \sum_{a'} \sum_m A_{a',m}$
 3: Marginal probability of an user $u$ in $U$, $P(u) = \sum_m U_{u,m} / \sum_{u'} \sum_m U_{u',m}$
 4: Marginal probability of an movie $m$ in $U$, $P(m) = \sum_u U_{u,m} / \sum_u \sum_{m'} U_{u,m'}$
 5: Let $U$ be the User $\times$ Item matrix and $A$ be the Attribute $\times$ Item matrix.
 6: $U^{Train} \leftarrow U$, $A^{Train} \leftarrow A$
 7: $U^{Eval} \leftarrow \mathbf{0}$, $A^{Eval} \leftarrow \mathbf{0}$
 8: Set of simple personalized queries, $Q_{U \cap A} \leftarrow \phi$
 9: while $|Q_{U \cap A}|$ < MAX SAMPLE SIZE do
10:     Sample an attribute $a$ from $\mathcal{A}$ according to $P(a)$.
11:     Sample a movie $m$ from for the attribute $a$, i.e., Sample from $\{m' | A_{a,m'} = 1\}$, according to
        $P(m)$
12:     Sample a user $u$ from who has rated movie $m$, i.e., Sample from $\{u' | U_{m,u'} = 1\}$, according
        to $P(u)$
13:     $U^{Train}_{u,m} = 0$, $A^{Train}_{a,m} = 0$, $U^{Eval}_{u,m} = 1$, $A^{Eval}_{a,m} = 1$
14:     $Q_{U \cap A}$.INSERT($(u, a, m)$)
15: end while
```

Algorithm 2 Personalised Complex Query Generation Algorithm

```
 1: Compositional Query sets $Q_{U \cap A_1 \cap A_2}$, $Q_{U \cap A_1 \cap \neg A_2}$
 2: Non-Trivial attribute combination set $\mathcal{A}_\circ$
 3: for each user-movie tuple in Eval set, i.e., $(u, m) \in \{(u, m) | U^{Eval}_{u,m} = 1\}$ do
 4:     for each pair of attributes $(a_1, a_2) \in \{(a_1, a_2) | A^{Eval}_{a_1,m} = 1$ and $A^{Eval}_{a_2,m} = 1\}$ do
 5:         if the pair is viable and non-trivial, i.e., $(a_1, a_2) \in \mathcal{A}_\cap$ then
 6:             $Q_{U \cap A_1 \cap A_2}$.INSERT($(u, a_1, a_2, m)$)
 7:         end if
 8:     end for
 9:     for each pair of attributes $(a_1, a_2) \in \{(a_1, a_2) | A^{Eval}_{a_1,m} = 1$ and $A_{a_2,m} = 0\}$ do
10:         if the pair is viable and non-trivial, i.e., $(a_1, a_2) \in \mathcal{A}_\setminus$ then
11:             $Q_{U \cap A_1 \cap \neg A_2}$.INSERT($(u, a_1, a_2, m)$)
12:         end if
13:     end for
14: end for
```

B.2 Training Details

Table 6: Hyper Parameter range for all the dataset. We run 100 runs for both models and select the best model on User-Movie validation set NDCG metric

Hyperparameters	Range Box	Best Value Box	Range Vector	Best Value Vector
Embedding dim	64	64	128	128
Learning Rate	1e-1, 1e-2, 1e-3, 1e-4, 1e-5	0.001	1e-1, 1e-2, 1e-3, 1e-4, 1e-5	0.001
Batch Size	64, 128, 256, 512, 1024	128	64, 128, 256, 512, 1024	128
# Negatives	1, 5, 10, 20	20	1, 5, 10, 20	5
Intersection Temp	10, 2, 1, 1e-1, 1e-2, 1e-3, 1e-5	2.0	-	-
Volume Temp	10, 5, 1, 0.1, 0.01, 0.001	0.01	-	-
Attribute Loss const	0.1, 0.3, 0.5, 0.7, 0.9	0.7	0.1, 0.3, 0.5, 0.7, 0.9	0.5

Hyperparameters are reported in Table 6. Best parameter values are reported for Box Embeddings and MF method.

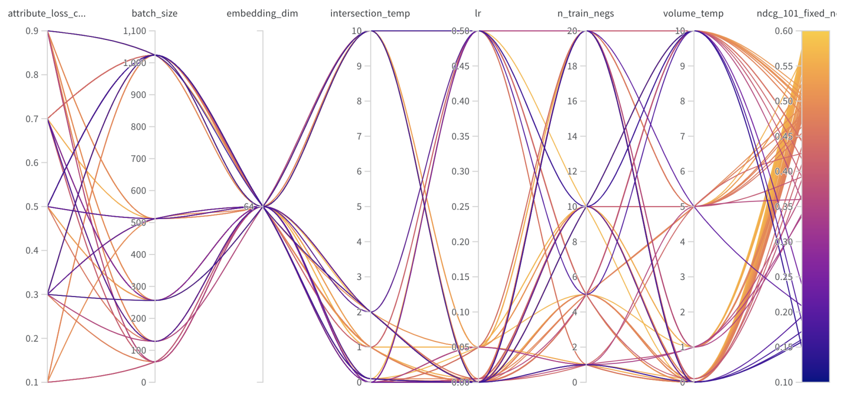


Figure 2: Parallel Co-ordinate plot for different hyperparameters vs model performance. Lighter the color, better the model's performance.

B.3 Set-Theoretic Generalization

Table 7: Hit Rate(%)↑ for Set-theoretic queries for dataset ML-20M.

Methods	$U \cap A$			$U \cap A_1 \cap A_2$			$U \cap A_1 \cap \neg A_2$		
	h@10	h@20	h@50	h@10	h@20	h@50	h@10	h@20	h@50
MF-Filter	4.6	8.1	16.1	0.4	1.0	2.9	3.7	6.6	13.7
MF-Product	4.1	7.5	15.6	3.3	6.6	16.4	2.7	5.1	11.4
MF-Geometric	0.1	0.3	0.6	0.0	0.0	0.0	0.3	0.6	1.4
NeuMF-Filter	4.6	8.2	16.1	1.1	5.6	6.4	4.9	7.3	13.9
NeuMF-product	4.6	8.2	16.1	4.1	8.5	22.1	4.3	6.9	12.0
Box-Filter	4.6	8.1	16.1	11.0	21.8	42.3	4.6	7.7	16.3
Box-Product	4.5	8.2	16.1	11.1	21.8	42.5	4.3	7.1	15.1
Box-Geometric	4.5	8.1	16.2	11.0	21.8	42.4	**6.4**	**12.8**	**25.9**

B.4 Spectrum of Weak Generalization

Table 8: The spectrum of generalization for Simple Personalized query $U \cap A$. W: Weakest Generalization, W-U: Weak Generalization-User, W-A: Weak Generalization-Attribute, S: Set Theoretic Generalization

Methods	Hit Rate @10	Hit Rate @ 20	Hit Rate @ 50
	W | W-U | W-A | S	W | W-U | W-A | S	W | W-U | W-A | S
MF-Filter	24.7 | 6.7 | 13.0 | 5.0	36.3 | 13.3 | 20.7 | 10.2	54.2 | 30.1 | 33.3 | 22.3
MF-Product	23.3 | 5.7 | 13.1 | 4.3	35.0 | 10.8 | 21.4 | 8.5	54.7 | 24.2 | 38.8 | 20.4
MF-Geometric	4.9 | 0.9 | 1.8 | 0.4	7.9 | 1.7 | 3.3 | 0.9	15.1 | 4.5 | 7.4 | 3.0
Box-Filter	24.1 | 13.0 | 16.4 | 11.7	34.5 | 22.3 | 24.6 | 19.1	50.5 | 40.5 | 37.6 | 32.3
Box-Product	25.2 | 13.6 | 13.9 | 10.0	35.2 | 21.5 | 21.9 | 16.7	52.2 | 38.4 | 38.3 | 31.5
Box-Geometric	25.4 | 14.7 | 14.8 | 11.0	35.6 | 23.3 | 23.5 | 18.3	**52.2 | 40.8 | 40.5 | 34.1**

Table 9: The spectrum of generalization for COMPLEX PERSONALIZED QUERY $U \cap A_1 \cap \neg A_2$. W: WEAKEST GENERALIZATION, W-U: WEAK GENERALIZATION-USER, W-A: WEAK GENERALIZATION-ATTRIBUTE, S: SET THEORETIC GENERALIZATION

Methods	Hit Rate @10	Hit Rate @ 20	Hit Rate @ 50
	W I W-U I W-A I S	W I W-U I W-A I S	W I W-U I W-A I S
MF-FILTER	25.5 I 13.0 I 12.4 I 4.7	34.9 I 14.1 I 19.5 I 9.8	54.7 I 29.5 I 37.1 I 22.5
MF-PRODUCT	23.5 I 7.0 I 10.4 I 3.4	34.9 I 12.8 I 18.0 I 7.3	54.5 I 27.5 I 35.0 I 19.3
MF-GEOMETRIC	5.2 I 2.0 I 1.7 I 0.5	8.8 I 3.5 I 1.9 I 1.0	17.4 I 8.8 I 6.5 I 2.7
BOX-FILTER	24.1 I 15.3 I 15.0 I 11.4	35.5I 22.7 I 21.1 I 19.5	**54.1 I 39.2 I 37.3 I 34.0**
BOX-PRODUCT	21.1 I 13.7 I 12.0 I 8.9	30.5 I 21.7 I 19.3 I 15.2	47.4 I 38.0 I 35.0 I 29.4
BOX-GEOMETRIC	21.1 I 13.2 I 10.8 I 8.6	30.4 I 20.8 I 17.7 I 15.1	**47.3 I 36.6 I 33.2 I 31.0**

Table 10: The spectrum of generalization for COMPLEX PERSONALIZED QUERY $U \cap A_1 \cap A_2$. W: WEAKEST GENERALIZATION, W-U: WEAK GENERALIZATION-USER, W-A: WEAK GENERALIZATION-ATTRIBUTE, S: SET THEORETIC GENERALIZATION

Methods	Hit Rate @10	Hit Rate @ 20	Hit Rate @ 50
	W I W-U I W-A I S	W I W-U I W-A I S	W I W-U I W-A I S
MF-FILTER	35.3 I 17.6 I 16.9 I 11.4	45.0 I 27.3 I 23.3 I 17.9	55.2 I 41.9 I 30.5 I 27.5
MF-PRODUCT	34.0 I 11.0 I 11.6 I 5.1	47.3 I 19.6 I 20.1 I 10.6	67.4 I 38.5 I 39.3 I 26.1
MF-GEOMETRIC	6.13 I 3.1 I 0.3 I 0.1	9.90 I 5.8 I 0.6 I 0.2	18.5 I 12.9 I 1.8 I 0.8
BOX-FILTER	30.8 I 21.5 I 17.3 I 14.5	41.1 I 31.2 I 23.3 I 20.5	52.7 I 44.5 I 30.3 I 28.5
BOX-PRODUCT	35.4 I 23.8 I 13.4 I 10.6	47.0 I 34.5 I 21.7 I 17.8	64.6 I 52.8 I 39.0 I 34.2
BOX-GEOMETRIC	34.6 I 25.2 I 20.0 I 16.8	45.7 I 35.7 I 30.5 I 26.6	**62.6 I 53.3 I 50.1 I 46.1**

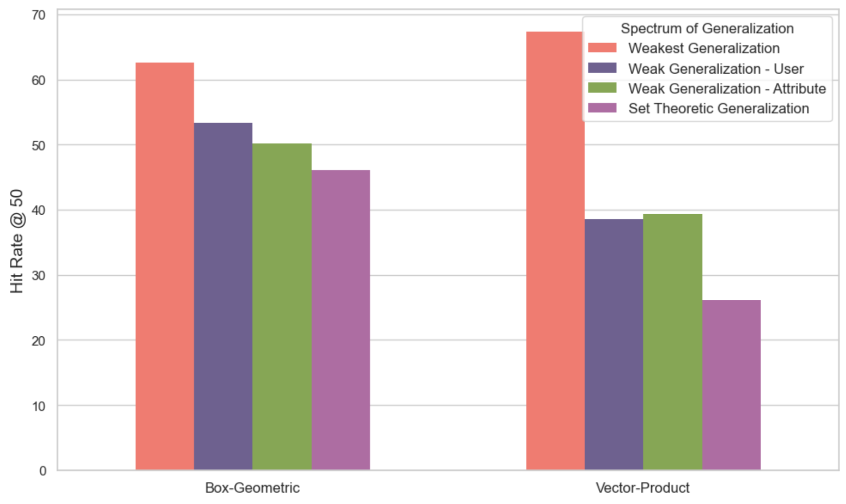
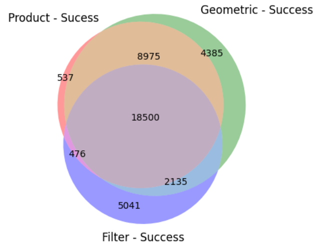


Figure 3: Spectrum of Set-theoretic Generalization.

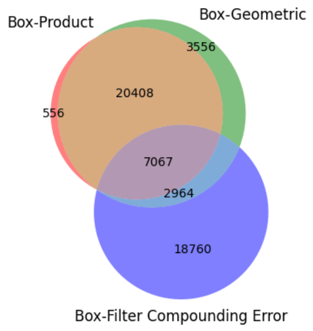
The BOX-GEOMETRIC achieves the best *Generalization Spectrum Gap* for all types of queries.

C ERROR COMPOUNDING ANALYSIS

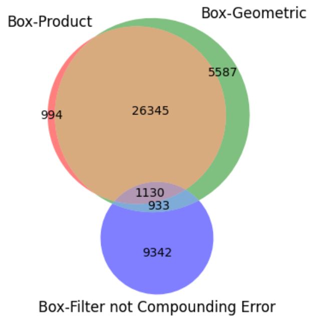
We further perform more granular analysis amongst the BOX based methods with complex query type $U \cap A_1 \cap A_2$. As claimed in our initial hypothesis, the FILTER method suffers from error compounding. If the target movie m is in the model's prediction list for A_1 but not for A_2 or the other way round, we denote this error as *compounding error*. In figure 4b, out of the compounding errors, 34% is solved by the BOX-GEOMETRIC method and 26% by the BOX-PRODUCT method. However, in figure 4c, for the error that is not due to compounding (where the model gets both A_1



(a) Relationships of correct answers by the three box models on $u \wedge a_1 \wedge a_2$ queries.



(b) The Geometric method subsumes the benefit of the product in compounding error.



(c) The effect is less for the non-compounding error.

and A_2 prediction wrong), only 18% are corrected by the BOX-GEOMETRIC method and a mere 10% of them are corrected by BOX-PRODUCT. Refer to figure 4a 4b 4c for details. This demonstrates that the BOX-GEOMETRIC significantly contributes to the correction of error compounding.

D TIME EFFICIENCY ANALYSIS

Table 11: Training time (*mm:ss*) for a single epoch are measured for different batch sizes with 5 negative samples on Movielens-1M dataset. Experiments are conducted on Nvidia GTX 1080Ti gpus

Batch Size	MF	NEUMF	LIGHTGCN	BOX
64	08:37	17:00	70:30	19:32
128	04:32	09:46	38:40	11:40
256	02:29	04:40	20:55	05:28
512	01:18	02:23	10:47	02:54
1024	00:40	01:20	05:24	01:12

In Table 11, we observe that the MF, being the simplest approach with minimal computational requirements, is consistently the fastest across all batch sizes. At the largest batch size (1024), it achieves the shortest training time of just 00:40. The BOX-based method exhibits training times comparable to NEUMF. However, it is significantly faster than LIGHTGCN, which relies on graph convolutional computations. The iterative message-passing operations required by LIGHTGCN result in considerably higher training times, particularly at smaller batch sizes (e.g., 70:30 at a batch size of 64). As the batch size increases, the training time for BOX embeddings becomes almost as efficient as MF. For instance, at a batch size of 1024, BOX achieves a training time of 01:12, compared to 00:40 for MF. This demonstrates that the computational complexity of box embeddings is of the same order as MF, making it a scalable and efficient choice.

Box embeddings are generally quite fast because the computation of box intersection volumes can be parallelized over dimensions. Note that the training times above use GumbleBox embeddings, which involve log-sum-exp calculations. However, this could be improved even further at inference time by replacing these soft min and max approximations with hard operators. If such an optimized approach is desired, then training can accommodate this by regularizing temperature. For deployment in industrial set-up, we could take additional steps with Box Embeddings as outlined in Mei et al. (2022b).