Human-inspired, Task-Dimension-Guided Exploration for Efficient Learning in High Dimensions

Fanyu Zhu[1,2,3] **Jiahui An**[1,2,3,4] **Ni Ji**[1,2,3,*]
[1]Beijing Institute for Brain Research,
Chinese Academy of Medical Sciences & Peking Union Medical College
[2]Chinese Institute for Brain Research, Beijing
[3]Beijing Key Laboratory of Brain Science and Brain-Machine Interface
[4]State Key Laboratory of Cognitive Neuroscience and Learning, Beijing Normal University
{zhufanyu,anjiahui,niji}@cibr.ac.cn

Abstract

Efficient exploration in high-dimensional decision spaces remains a central challenge for decision-making systems. Humans, in contrast, can navigate large decision spaces with remarkable efficiency. Recent behavioral studies suggest that humans reduce dimensionality in large decision spaces by probing candidate feature dimensions, identifying reward-relevant ones, and restricting the effective decision space. Inspired by this mechanism, we propose **TDGE** (*Task-Dimension-Guided Exploration*), a human-inspired, model-agnostic algorithm with an automatically constructed task-dimension–feature–item hierarchy. TDGE follows a top-down exploration strategy: it first selects task-relevant feature dimensions, then identifies informative features within those dimensions, and finally recommends concrete items based on the selected features. Experiments on MovieLens-20M, Last.fm, and Amazon recommendation datasets show that TDGE substantially improves exploration efficiency and cold-start adaptation over baseline algorithms. Comparisons with other structured algorithms and ablation studies attribute these gains to TDGE's hierarchical structure and semantic feature-space exploration, with robust results across clustering methods and hierarchy depths. Recommendation-trajectory visualizations also show exploration patterns similar to human dimension-guided behavior. Code is available at `https://github.com/LordSlime/Human-inspired-Task-Dimension-Guided-Exploration-for-Efficient-Learning`.

1 Introduction

Exploration in large decision spaces remains a central challenge in reinforcement learning and other sequential decision-making settings. Classical strategies such as ϵ-greedy [32, 29], upper confidence bound (UCB) [2], and Thompson sampling [31] provide established approaches to balancing exploration and exploitation. However, efficient exploration becomes challenging when the action space is large and reward-relevant structure is unknown. Agents may spend many interactions evaluating low-reward actions while learning the context–reward relationship under partial feedback. Work on high-dimensional Bayesian optimization suggests that exploiting structured low-dimensional decompositions can improve search efficiency [11, 24]. These findings motivate exploration strategies that identify and exploit task-relevant structure in large decision spaces.

Humans often navigate large decision spaces efficiently. For example, when choosing a movie, a person may first narrow the search along abstract feature dimensions, such as genre, mood, or period, and then select specific titles within the reduced space. Recent behavioral work [1] suggests that

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

humans actively reduce the effective dimensionality of a task by probing candidate feature dimensions, identifying those predictive of reward, and restricting subsequent exploration accordingly. Applying this principle to real-world decision problems requires constructing meaningful feature dimensions and learning their relevance from feedback.

This problem is especially important in online recommendation, where an agent selects items from a large catalog for each arriving user and receives click or rating feedback as reward. This setting is naturally formulated as a contextual bandit [15], with neural variants extending reward estimation to nonlinear context–reward relationships [37, 35]. A further challenge is the user cold-start problem, where the system must adapt to new users with limited interaction feedback [25, 18]. Together, online learning and user cold-start adaptation provide a practical setting for evaluating exploration methods in large decision spaces.

Prior work has explored various approaches to exploration in large decision spaces. Representative examples in reinforcement learning include continuous action embeddings with nearest-neighbor search [6], action elimination using auxiliary environmental signals [34], and policy learning in a learned low-dimensional action space to enable generalization across similar actions [5]. In contextual bandits, CoFineUCB [33] uses a coarse parameter subspace to guide learning in the full linear reward-parameter space, while H_2N-Bandit [20] filters candidate arms through a predefined item-category hierarchy. These methods illustrate different ways of incorporating structure into exploration. TDGE constructs semantic dimensions from item-descriptive features and treats dimensions and features as explicit intermediate actions for exploration and candidate generation.

Motivated by human dimension-level exploration, we propose **TDGE** (*Task-Dimension-Guided Exploration*), a model-agnostic exploration framework for contextual-bandit recommendation. TDGE automatically groups item-descriptive features into semantic task dimensions using clustering and a KGS-based selection criterion [12]. It then organizes exploration in a top-down manner: selecting task-relevant dimensions, selecting features within those dimensions, and recommending an item from the induced candidate pool. Selected feature scores and item–feature relevance jointly determine candidate ranking. The observed item reward updates the item-level agent and is propagated to matched selected features and their source dimensions through relevance-weighted updates. This coupling of hierarchical selection, candidate generation, and credit assignment enables feedback-driven exploration in the semantic feature space.

Our contributions are as follows.

- Inspired by **human dimension-level exploration**, we propose **TDGE**, a **model-agnostic** framework that makes explicit online dimension–feature–item decisions over a semantic hierarchy **automatically constructed from item metadata**, without substantially increasing inference time. Feature-level scores guide candidate generation, while item feedback updates matched features and their source dimensions.
- We evaluate TDGE with **three contextual-bandit backbones** on MovieLens-20M, Last.fm, and Amazon. TDGE significantly reduces both online and cold-start adaptation regret on all three datasets, demonstrating **high sample efficiency** and **rapid adaptation to unseen users**.
- Comparisons with structured contextual-bandit baselines and ablation studies support the effectiveness of **semantic feature-space exploration and dimension-level routing**. Additional experiments demonstrate that TDGE is **robust to the choice of clustering method and hierarchy depth** within the evaluated settings.

2 Background and Related Work

2.1 Human exploration in high-dimensional environments

A substantial body of work in cognitive science has studied how humans learn and make decisions in environments with many candidate features. A recurring finding is that humans selectively attend to reward-relevant features, thereby reducing the effective dimensionality of the learning problem [17, 21, 1]. Behavioral and neuroimaging studies of multidimensional learning tasks provide evidence that attention supports the identification of reward-relevant dimensions [17]. Using eye tracking and fMRI, Leong et al. [14] further demonstrated a bidirectional interaction: attention influences value

estimation and updating, while feedback guides subsequent allocation of attention. These findings support the role of dimension-level selection in learning under uncertainty. TDGE draws on this principle by constructing semantic feature dimensions and learning which dimensions to explore through recommendation feedback.

2.2 Exploration in large decision spaces

Efficient exploration in large action spaces remains a central challenge in reinforcement learning and bandit learning. Representative reinforcement learning approaches include continuous action embeddings with nearest-neighbor search [6], action representations learned from state transitions to support generalization across similar actions [5], and action elimination using auxiliary environmental signals [34]. These methods improve learning or action selection by exploiting relationships among actions or restricting the available choices.

In bandit learning, zooming algorithms exploit similarity information over arms or context–arm pairs to adaptively refine exploration [13, 27], while hierarchical optimistic optimization guides exploration through hierarchical partitions of the search space [4]. Other methods introduce structure into reward estimation or candidate selection. CoFineUCB [33] uses a coarse parameter subspace to guide linear reward estimation in the full feature space. Taxonomy-based bandits exploit known relationships among arms [19], and H_2N-Bandit [20] uses available item categories in a bi-level neural architecture for category and item selection. Zhu et al. [38] further develop an efficient contextual-bandit algorithm for continuous, linearly structured action spaces using supervised-learning and action-optimization oracles. Bio-inspired algorithm applying a special type of stochastic continuous Hopfield network [10] managed to conduct uncertainty-directed exploration in ways similar to humans and animals.

Related ideas appear in high-dimensional Bayesian optimization. Representative approaches include additive models [11, 24], tree-structured additive decompositions [7], random tree-based decompositions [39], and group testing to identify active variables [8]. These methods improve optimization efficiency through objective-function decomposition or input-variable selection. TDGE instead explicitly models user preferences across semantic feature space and item space, allowing item feedback to inform preference learning over shared features and thereby supporting sample-efficient exploration and rapid user adaptation.

2.3 Contextual bandits and reinforcement learning for recommendation

Online recommendation provides a natural setting for exploration algorithms, as the agent must select items from large catalogs and adapt to users with limited feedback. Contextual bandits model each recommendation as a one-step decision and capture the exploration–exploitation trade-off [15]. LinUCB uses linear reward estimation with upper-confidence-bound exploration, while NeuralUCB and NeuralTS combine nonlinear reward models with uncertainty-based action selection [37, 35]. Neural approaches to Thompson sampling also include approximate Bayesian methods evaluated by Riquelme et al. [23]. Beyond these approaches, EE-Net [3] uses separate exploitation and exploration networks, with the latter learning potential gains relative to the current reward estimate. Bandit methods have also been studied for personalized recommendation and the user cold-start problem [30, 26].

Reinforcement learning methods extend recommendation to sequential decisions that optimize long-term user engagement [36, 9, 16]. For example, SlateQ decomposes the long-term value of a recommendation slate into item-level quantities under assumptions on user choice, making learning tractable in a combinatorial action space [9]. TDGE focuses on a complementary aspect of recommendation: organizing candidate generation through semantic dimension–feature decisions before item-level selection. We evaluate this mechanism with linear and neural contextual-bandit backbones in online learning and user cold-start adaptation.

3 Task-Dimension-Guided Exploration

We begin with the behavioral observation that motivates TDGE, and then present its three main components: feature-dimension construction from item metadata, multi-level value and uncertainty estimation with standard bandit backbones, and top-down recommendation through dimension-, feature-, and item-level selection. Full mathematical details are deferred to Appendix C.

3.1 Inspiration from human exploration

A recent behavioral study [1] examined how humans explore high-dimensional decision spaces using a food recommendation task. As shown in Figure 1(a), participants were asked to infer a customer's preferences by arranging food combinations to maximize reward. Each food item was described by multiple feature dimensions, but only two dimensions were reward-relevant in each session. Participants were not told which dimensions were relevant or how many dimensions mattered.

Participants adopted a structured exploration strategy consistent with dimension-guided exploration (DGE). Specifically, they first **selected one feature dimension for exploratory processing**. Within the chosen dimension, they **repeatedly sampled different feature combinations** by rearranging items across rows, allowing them to assess the relationship between that dimension and reward outcomes. After sufficiently probing one dimension, they **switched to another untested dimension** and repeated the same process. This sequential, dimension-wise exploration continued until the candidate dimensions had been evaluated, after which participants shifted from exploration to exploitation to maximize reward. Figure 1(b,d) illustrates this dimension-wise exploration process.

Figure 1(c) shows that DGE is strongly associated with task performance: as the proportion of DGE rounds increases, so does the probability of achieving a full score. This dimension-wise exploration pattern motivates TDGE, which extends the same principle into a scalable exploration framework for contextual bandit recommendation.

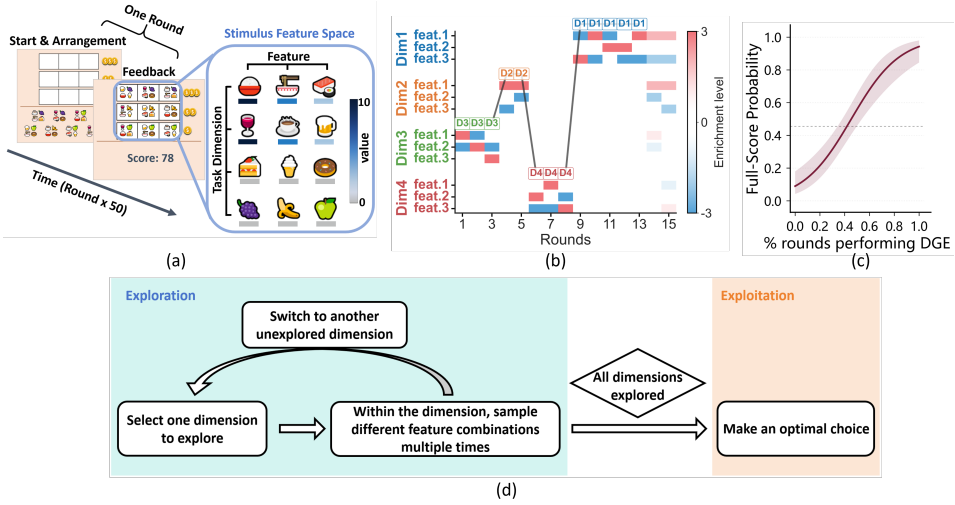


Figure 1: Human exploration through dimension-level search. (a) Behavioral task setup. (b) Reward-relevant dimensions reduce the effective action space. (c) Association between the proportion of DGE rounds and the probability of achieving a full score. (d) Schematic illustration of the observed dimension-wise exploration strategy. Adapted from An et al. [1] with permission.

3.2 Problem formulation

We formulate recommendation as a contextual bandit problem with partial feedback. Let $\mathcal{I}$ denote the full item catalog. At round t, the agent observes a user representation u_t, recommends one item $a_t \in \mathcal{I}$, and observes reward only for that item. For item i, the item-level context is

$$x_{i,t}^{\text{item}} = \text{norm}(u_t \oplus v_i \oplus b),$$

where v_i is the item representation, b is a constant bias feature, $\oplus$ denotes concatenation, and $\text{norm}(\cdot)$ denotes ℓ_2 normalization. The observed reward is

$$r_t = h(x_{a_t,t}^{\text{item}}) + \xi_t,$$

where $h : \mathbb{R}^d \to [0, 1]$ is an unknown expected reward function and ξ_t is zero-mean conditionally ν-sub-Gaussian noise. The objective is to minimize cumulative regret,

$$R_T = \sum_{t=1}^{T} \Big[h(x^{\text{item}}_{a^*_t,t}) - h(x^{\text{item}}_{a_t,t}) \Big], \qquad a^*_t \in \arg\max_{i \in \mathcal{I}} h(x^{\text{item}}_{i,t}),$$

where a^*_t is an optimal item for the current user over the full catalog, independent of candidate sampling or filtering. Higher-level contexts for feature dimensions and features are defined analogously; full details and the offline candidate-sampling and reward protocol are given in Appendix A.

Dimension-guided exploration objective The key distinction from a standard contextual bandit lies in how exploration is structured. A standard item-level policy explores directly over candidate items, without explicitly restricting search through intermediate semantic dimensions. TDGE instead first selects a small set of feature dimensions, then selects candidate features within those dimensions, and finally performs item-level selection within the candidate pool $\mathcal{P}_t \subseteq \mathcal{I}$:

$$a_t = \arg\max_{i \in \mathcal{P}_t} s^{\text{item}}_{i,t},$$

where $s^{\text{item}}_{i,t}$ is the item-level selection score. TDGE does not change the final reward objective; instead, it improves exploration by using feature-dimension selection to construct a task-adaptive candidate pool before item-level scoring. In this sense, its benefit comes from better candidate-space restriction as well as more efficient item-level search within that restricted space.

3.3 Task-dimension extraction

TDGE requires a dimension-level exploration space that groups item-descriptive features into semantically coherent dimensions. Let $\mathcal{F} = \{f_1, \ldots, f_M\}$ denote the feature set, such as item tags in recommendation datasets. We seek a partition $\mathcal{C} = \{C_1, \ldots, C_{k^*}\}$ of $\mathcal{F}$, where each cluster corresponds to one task dimension used by the dimension-level agent.

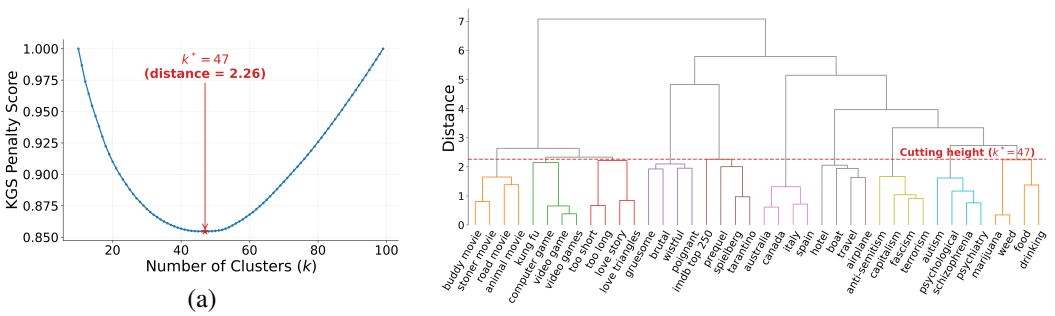
We embed each feature f_j as a normalized vector $e_j \in \mathbb{R}^{768}$ using a pretrained Sentence-BERT model [22, 28], and cluster the embeddings to obtain $\mathcal{C}$. We use agglomerative hierarchical clustering with Ward's linkage as the default method. Its dendrogram allows feature dimensions to be extracted at different resolutions without rerunning clustering and supports extensions with nested dimension levels. The standard three-level TDGE framework requires only one feature partition and can also use other clustering methods.

We determine the number of task dimensions using a KGS-based criterion that balances cluster compactness against over-partitioning. Each cut of the dendrogram defines a candidate partition. Among partitions satisfying a minimum cluster-size constraint, we select the lowest-scoring partition and denote its number of clusters by k^* (Figure 2(a)). Full details of the criterion and constrained selection rule are given in Appendix B. The resulting clusters capture related semantic aspects of items, such as genre, mood, topic, or style. Choosing a task dimension therefore directs exploration toward a coherent group of descriptive features. Figure 2(b) shows representative tag clusters extracted from MovieLens-20M. Clustering is performed once during preprocessing and reused throughout training and evaluation.

3.4 Multi-Level Exploration Schedule

TDGE performs top-down exploration over three levels: task dimension, feature, and item. The dimension-level agent identifies promising semantic aspects of the current user's preferences and restricts feature exploration to the selected dimensions. The feature-level agent identifies specific attributes within those dimensions and uses their scores to guide item candidate generation. The item-level agent evaluates concrete items in the resulting pool and makes the final recommendation. Each level maintains a separate contextual-bandit agent instantiated with the same backbone.

All three levels score arms using the current user representation and the corresponding arm representation. Let $x^{(\ell)}_{z,t}$ be the context of arm z at level $\ell \in \{\text{dim}, \text{feat}, \text{item}\}$. The scoring and selection



(a) (b)

Figure 2: (a) Normalized KGS penalty versus cluster count on MovieLens-20M; k^* minimizes the penalty among candidate partitions satisfying the cluster-size constraint. (b) Representative portion of the feature dendrogram: semantically related tags of the same color form a task dimension.

procedures are

$$\begin{cases} s_{c,t}^{(\mathrm{dim})} = \mathcal{A}_t^{(\mathrm{dim})}\big(x_{c,t}^{(\mathrm{dim})}\big), & \mathcal{D}_t = \mathrm{TopK}_{C_c \in \mathcal{C}}\big(s_{c,t}^{(\mathrm{dim})}, K_1\big), \\ s_{j,t}^{(\mathrm{feat})} = \mathcal{A}_t^{(\mathrm{feat})}\big(x_{j,t}^{(\mathrm{feat})}\big), & \mathcal{F}_t = \mathrm{TopK}_{f_j \in \bigcup_{C_c \in \mathcal{D}_t} C_c}\big(s_{j,t}^{(\mathrm{feat})}, K_2\big), \\ s_{i,t}^{(\mathrm{item})} = \mathcal{A}_t^{(\mathrm{item})}\big(x_{i,t}^{(\mathrm{item})}\big), & a_t \in \arg\max_{i \in \mathcal{P}_t} s_{i,t}^{(\mathrm{item})}. \end{cases} \tag{1}$$

Here, $\mathcal{A}_t^{(\ell)}$ denotes the arm-scoring rule of the corresponding agent, incorporating reward estimation and the backbone's exploration mechanism. TopK retains up to the specified number of highest-scoring arms. Context construction and backbone-specific scoring rules are given in Appendices A and C.

The feature level connects semantic exploration to item selection. Among items supplied by the candidate-sampling procedure, TDGE ranks those sharing at least one selected feature using

$$S_t(i) = \sum_{j:\, f_j \in \mathcal{F}_t \cap \mathrm{feat}(i)} s_{j,t}^{(\mathrm{feat})} \rho_{i,j}, \tag{2}$$

where $\mathrm{feat}(i)$ denotes the descriptive features of item i and $\rho_{i,j}$ is its relevance to feature f_j. The top K matching items form $\mathcal{P}_t$, from which the item-level agent selects a_t according to Eq. 1.

Hierarchical update After observing reward r_t, TDGE updates the item agent with unit sample weight and propagates feedback through the matched features

$$\mathcal{H}_t = \mathcal{F}_t \cap \mathrm{feat}(a_t). \tag{3}$$

Each $f_j \in \mathcal{H}_t$ updates the feature agent using r_t with sample weight $\rho_{a_t,j}$. Each source dimension C_c containing matched features is updated using the same reward with weight

$$\omega_{a_t,c} = \max_{j:\, f_j \in C_c \cap \mathcal{H}_t} \rho_{a_t,j}. \tag{4}$$

The intersection determines which selected features and dimensions receive feedback, while relevance weights determine their contribution to learning. This connects item-level outcomes with preference learning over shared semantic features, allowing each interaction to inform subsequent recommendations.

The full procedure is illustrated in Figure 3. Detailed update equations and pseudocode are provided in Appendix C.

4 Experiments

We evaluate TDGE by addressing four questions: (1) whether it improves sample efficiency in online learning and cold-start adaptation across different contextual-bandit backbones; (2) how it compares with other structured exploration methods; (3) which components account for its gains; and (4) how robust it is to the choice of clustering method and hierarchy depth.

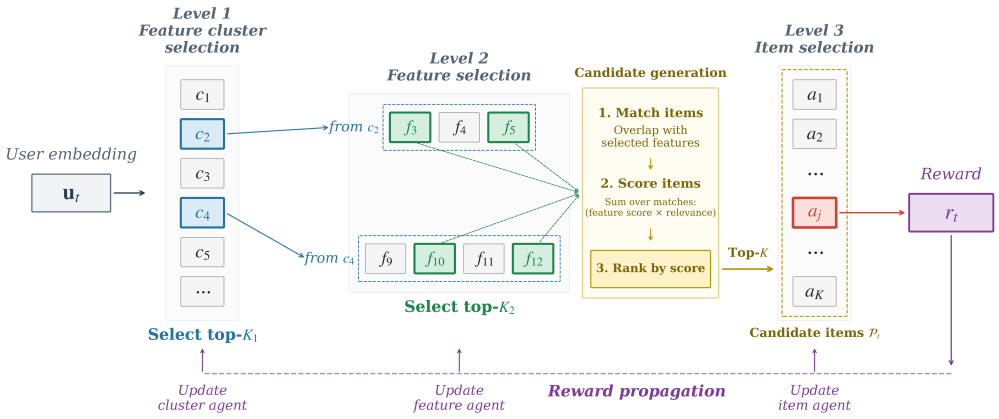


Figure 3: The TDGE exploration schedule. The dimension agent selects promising semantic aspects, the feature agent identifies specific attributes and guides candidate generation, and the item agent makes the final recommendation. The observed reward updates the item agent and the matched selected features and their source dimensions.

4.1 Experiment Setup

Datasets We evaluate TDGE on MovieLens-20M, Last.fm, and the Beauty and Personal Care subset of Amazon. For each dataset, item tags are extracted from metadata as descriptive features. Dataset statistics, tag-selection details, and preprocessing procedures are provided in Appendix D. Users are split into an online training pool and a user cold-start pool with a 9:1 ratio.

Baselines We compare TDGE-LinUCB, TDGE-NeuralUCB, and TDGE-NeuralTS with their corresponding item-level backbones: LinUCB [15], NeuralUCB [37], and NeuralTS [35]. We also compare with two structured contextual-bandit methods: CoFineUCB [33], which uses a prior coarse parameter subspace to guide linear reward estimation, and H_2N-Bandit [20], which filters candidate arms through a predefined item-category hierarchy.

Evaluation Protocol All methods follow the same offline evaluation protocol, detailed in Appendix D. Within each matched TDGE–baseline pair, we hold fixed the dataset preprocessing, user splits, item-level contexts, interaction horizons, item-level agent configuration, and final item budget ($K = 10$); only the exploration and candidate-generation mechanism differs. We report online cumulative regret (CReg) and cold-start final-round regret as mean $\pm$ standard deviation over five seeds, with significance assessed using two-sided Welch t-tests:$^{*}p < 0.05$, $^{**}p < 0.01$, and $^{***}p < 0.001$; n.s. denotes $p \geq 0.05$.

4.2 Online and Cold-Start Recommendation Results

For online evaluation, each method interacts with training users for $T = 10{,}000$ rounds. For cold-start evaluation, we randomly sample 100 held-out users, initialize them with the same mean training-user representation, and run an independent 10-step interaction trajectory for each user.

Table 1 summarizes both settings. With neural backbones, TDGE significantly reduces online cumulative regret by 4.5%–31.0% across all six dataset–backbone comparisons, demonstrating improved online sample efficiency. TDGE also lowers final-round cold-start regret by 10.5%–74.9% in all six comparisons, with statistically significant gains in five. With LinUCB backbone, TDGE significantly reduces online cumulative regret and final-round cold-start regret on MovieLens-20M by 31.6% and 65.8%, respectively, while differences on Last.fm and Amazon are not statistically significant. The limited gains on these datasets may reflect the linear reward model's restricted ability to capture complex user–item interactions, limiting the benefit of TDGE's exploration framework. These results show that TDGE's semantic hierarchy and learned routing policy support more sample-

efficient exploration and adaptation to unseen users without user-specific historical contexts, with the most consistent gains obtained using neural reward models.

Table 1: Online cumulative regret at $T = 10{,}000$ and final-round cold-start regret at $t = 10$ for TDGE and its corresponding item-level baselines. Lower is better. Results are mean $\pm$ sample standard deviation over five seeds. Superscripts compare TDGE with its corresponding backbone using two-sided Welch t-tests.

Metric	Dataset	Backbone	Baseline	TDGE
Online CReg ↓	MovieLens-20M	LinUCB	2299.697±20.439	**1573.142±43.499*****
		NeuralUCB	2417.798±15.161	**1669.303±91.162*****
		NeuralTS	2418.077±15.964	**1705.542±52.769*****
	Last.fm	LinUCB	**2438.154±43.877**	2453.269±27.544 n.s.
		NeuralUCB	2233.387±13.230	**1960.563±57.620*****
		NeuralTS	2234.352±16.528	**1975.365±81.538****
	Amazon	LinUCB	1314.280±30.398	**1303.841±29.050** n.s.
		NeuralUCB	1245.925±22.689	**1169.567±32.945****
		NeuralTS	1233.443±30.148	**1177.886±26.246***
Cold-start Regret ($t = 10$) ↓	MovieLens-20M	LinUCB	0.234±0.004	**0.080±0.013*****
		NeuralUCB	0.239±0.013	**0.060±0.015*****
		NeuralTS	0.236±0.014	**0.078±0.026*****
	Last.fm	LinUCB	0.260±0.006	**0.260±0.005** n.s.
		NeuralUCB	0.239±0.007	**0.214±0.007*****
		NeuralTS	0.240±0.010	**0.212±0.024** n.s.
	Amazon	LinUCB	**0.137±0.023**	0.140±0.028 n.s.
		NeuralUCB	0.099±0.012	**0.065±0.022***
		NeuralTS	0.104±0.010	**0.061±0.012*****

4.3 Comparison with Structured Exploration Methods

CoFineUCB structures exploration in the reward-parameter space, while H_2N-Bandit uses a predefined item-category hierarchy. TDGE instead explores automatically constructed semantic feature dimensions. We compare CoFineUCB with LinUCB and TDGE-LinUCB as linear methods, and H_2N-Bandit with NeuralUCB and TDGE-NeuralUCB as neural methods. We evaluate these methods on MovieLens-20M using the same protocol as in Section 4.2.

As shown in Table 2, TDGE-LinUCB reduces online cumulative regret and final-round cold-start regret relative to CoFineUCB by 30.7% and 66.4%, respectively. TDGE-NeuralUCB reduces the corresponding regret relative to H_2N-Bandit by 29.9% and 74.6%. All four comparisons are significant at $p < 10^{-4}$, showing that TDGE also improves over methods pursuing related structured-exploration objectives.

Table 2: Online cumulative regret at $T = 10{,}000$ and final-round cold-start regret at $t = 10$ on MovieLens-20M. Lower is better. Results are mean $\pm$ standard deviation over five seeds. Superscripts compare TDGE-LinUCB with CoFineUCB and TDGE-NeuralUCB with H_2N-Bandit using two-sided Welch t-tests.

Metric	LinUCB	CoFineUCB	TDGE-LinUCB	NeuralUCB	H_2N-Bandit	TDGE-NeuralUCB
Online CReg ↓	2299.697±18.281	2271.580±34.295	**1573.142±38.906*****	2417.798±13.561	2379.700±17.122	**1669.303±81.538*****
Cold-start Regret ($t = 10$) ↓	0.234±0.004	0.237±0.010	**0.080±0.011*****	0.239±0.011	0.235±0.008	**0.060±0.014*****

4.4 Ablation Studies

All ablations are conducted on MovieLens-20M with the five-run protocol used for the main experiments.

4.4.1 Semantic Feature-Space Structure

We compare TDGE's feature-space hierarchy with a two-level item-space alternative that clusters semantic item embeddings and selects item-cluster–item routes. We reconstruct the feature dimen-

sions with Ward hierarchical clustering, k-means, spectral clustering, and Gaussian mixture models. Each feature-clustering method operates on the same tag embeddings and uses the constrained KGS criterion to choose its number of dimensions.

Table 3: Feature-space and clustering-method ablations on MovieLens-20M. Lower regret is better. The superscript indicates that hierarchical feature-space TDGE outperforms the corresponding item-clustering alternative under a two-sided Welch t-test.

Metric	Backbone	Item clustering	Hierarchical	k-means	Spectral	GMM
Online CReg	LinUCB	$2101.500 \pm 37.535^{***}$	1573.142 ± 38.906	1608.404 ± 49.511	1639.962 ± 135.799	1558.282 ± 63.691
	NeuralUCB	$2223.900 \pm 45.983^{***}$	1669.303 ± 81.538	1839.836 ± 124.657	1854.616 ± 42.444	1786.975 ± 35.025
	NeuralTS	$2295.800 \pm 13.952^{***}$	1705.542 ± 47.198	1815.254 ± 112.595	1792.321 ± 180.387	1709.182 ± 94.690
Cold-start Regret ($t = 10$)	LinUCB	$0.219 \pm 0.011^{***}$	0.080 ± 0.011	0.127 ± 0.005	0.118 ± 0.016	0.133 ± 0.024
	NeuralUCB	$0.210 \pm 0.014^{***}$	0.060 ± 0.014	0.075 ± 0.011	0.072 ± 0.015	0.072 ± 0.015
	NeuralTS	$0.212 \pm 0.018^{***}$	0.078 ± 0.023	0.077 ± 0.007	0.065 ± 0.007	0.069 ± 0.014

Feature-space TDGE substantially outperforms item-space clustering for every backbone and metric, while the four feature-clustering algorithms yield comparable results. The gains therefore arise from exploring the semantic feature space and do not depend on a particular clustering method.

4.4.2 Hierarchy Depth

The main model uses the three-level route *fine dimension* $\rightarrow$ *feature* $\rightarrow$ *item*. We compare it with a four-level route, *coarse dimension* $\rightarrow$ *fine dimension* $\rightarrow$ *feature* $\rightarrow$ *item*. Both dimension levels are obtained from the same feature dendrogram. The four-level model retains two coarse dimensions, five fine dimensions, ten features, and at most ten items.

Table 4: Online cumulative regret and cold-start final-round regret for the hierarchy-depth ablation on MovieLens-20M. Lower is better.

Metric	Backbone	TDGE (3 levels)	TDGE (4 levels)
Online CReg	LinUCB	1573.142 ± 38.906	1586.160 ± 69.115
	NeuralUCB	1669.303 ± 81.538	1768.740 ± 142.438
	NeuralTS	1705.542 ± 47.198	1606.220 ± 90.449
Cold-start Regret ($t = 10$)	LinUCB	0.080 ± 0.011	0.090 ± 0.022
	NeuralUCB	0.060 ± 0.014	0.078 ± 0.017
	NeuralTS	0.078 ± 0.023	0.060 ± 0.017

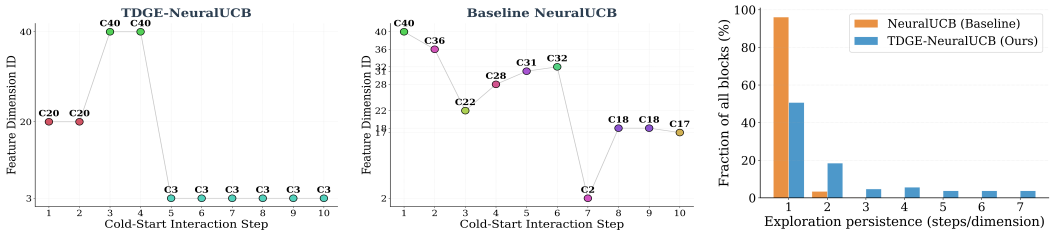
None of the differences between three- and four-level TDGE is statistically significant under two-sided Welch t-tests over five runs. Adding a coarse dimension level therefore provides no consistent improvement on MovieLens-20M. We use the three-level hierarchy because it preserves explicit dimension–feature–item semantics with lower routing and inference complexity. The effective hierarchy depth may depend on the semantic structure of the dataset, and deeper hierarchies may be beneficial for data with more complex multi-level structure.

4.5 TDGE exhibits human-like dimension-persistent probing

The behavioral study of An et al. [1] found that participants often explored several features within one task dimension before switching to another. We examine whether TDGE produces a similar trajectory pattern in cold-start recommendation, without claiming equivalence to human cognitive mechanisms.

Figure 4(a,b) compares TDGE-NeuralUCB and NeuralUCB for the same cold-start user. Because the baseline does not explicitly select dimensions, both trajectories are represented by the dominant feature dimension of the final recommended item, determined by majority vote over its tags. TDGE exhibits more persistent within-dimension exploration, whereas the baseline switches more frequently.

We quantify this behavior using the run lengths of consecutive same-dimension recommendations across all MovieLens-20M cold-start users. Figure 4(c) shows that TDGE-NeuralUCB places more probability mass on run lengths 2–5, while NeuralUCB is concentrated at length 1. Thus, TDGE produces exploration trajectories resembling the dimension-persistent pattern observed in human exploration.



(a) (b) (c)

Figure 4: Feature-dimension trajectories over 10 cold-start interactions and the distribution of consecutive same-dimension run lengths on MovieLens-20M. (a) TDGE-NeuralUCB. (b) NeuralUCB. Each step in (a,b) is represented by the dominant feature dimension of the recommended item. (c) Same-dimension run-length distributions aggregated across cold-start users.

5 Conclusion

We presented TDGE, a model-agnostic exploration framework inspired by human dimension-level exploration. TDGE explicitly models user preferences at the semantic dimension, feature, and item levels, coupling exploration in semantic feature space with selection in item space. Experiments on MovieLens-20M, Last.fm, and Amazon show significant reductions in online cumulative regret and cold-start final-round regret, particularly with the neural backbones. Ablations support the effectiveness of semantic feature-space exploration and dimension-level routing, and demonstrate robustness to different clustering methods.

However, the current version of TDGE still has several limitations. First, the retention sizes K_1 and K_2 are fixed. Ideally, they could be selected adaptively or learned online to better match different datasets and user regimes. Second, the limited effect of hierarchy depth in our ablations may be due to the structure of the evaluated data. Experiments on datasets with more complex hierarchical structure could help determine when deeper hierarchies are beneficial. Third, we evaluate TDGE only with contextual bandit backbones; extending the same dimension-guided exploration principle to general reinforcement learning is an important direction for future work.

In conclusion, these findings support explicit exploration over semantic dimensions as an effective way to improve sample efficiency in contextual-bandit recommendation.

6 Acknowledgments

This study was funded by Project (2025ZD0217400) supported by Brain Science and Brain-Like Intelligence Technology —National Science and Technology Major Project; and the CAMS Innovation Fund for Medical Sciences (CIFMS), 2024-RC180-02. We acknowledge Beijing Key Laboratory of Brain Science and Brain-Machine Interface; and Fundamental and Interdisciplinary Disciplines Breakthrough Plan of the Ministry of Education of China (JYB2025XDXM504).

References

[1] Jiahui An, Jiewen Hu, Yilin Elaine Wu, Siyu Ning, Chun Liu, Yuhang Pan, Fanyu Zhu, Ruosi Wang, and Ni Ji. Task-space dimensions guide human exploration in complex environments. *bioRxiv*, 2026. doi: 10.64898/2026.04.29.720265. URL `https://www.biorxiv.org/content/early/2026/05/04/2026.04.29.720265`.

[2] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2):235–256, 2002.

[3] Yikun Ban, Yuchen Yan, Arindam Banerjee, and Jingrui He. EE-net: Exploitation-exploration neural networks in contextual bandits. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=X_ch3VrNSRg`.

[4] Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12(5), 2011.

[5] Yash Chandak, Georgios Theocharous, James Kostas, Scott Jordan, and Philip Thomas. Learning action representations for reinforcement learning. In *International conference on machine learning*, pages 941–950. PMLR, 2019.

[6] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

[7] Eric Han, Ishank Arora, and Jonathan Scarlett. High-dimensional bayesian optimization via tree-structured additive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7630–7638, 2021.

[8] Erik Orm Hellsten, Carl Hvarfner, Leonard Papenmeier, and Luigi Nardi. High-dimensional bayesian optimization with group testing. *arXiv preprint arXiv:2310.03515*, 2023.

[9] Eugene Ie, Vihan Jain, Jing Wang, Sanmit Narvekar, Ritesh Agarwal, Rui Wu, Heng-Tze Cheng, Tushar Chandra, and Craig Boutilier. Slateq: A tractable decomposition for reinforcement learning with recommendation sets. In *IJCAI*, volume 19, pages 2592–2599, 2019.

[10] Chen Jiang, Jiahui An, Yating Liu, and Ni Ji. Brain bandit: A biologically grounded neural network for efficient control of exploration. In *International Conference on Learning Representations*, volume 2025, pages 64153–64184, 2025.

[11] Kirthevasan Kandasamy, Jeff Schneider, and Barnabás Póczos. High dimensional bayesian optimisation and bandits via additive models. In *International conference on machine learning*, pages 295–304. PMLR, 2015.

[12] Lawrence A Kelley, Stephen P Gardner, and Michael J Sutcliffe. An automated approach for clustering an ensemble of nmr-derived protein structures into conformationally related subfamilies. *Protein Engineering, Design and Selection*, 9(11):1063–1065, 1996.

[13] Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 681–690, 2008.

[14] Yuan Chang Leong, Angela Radulescu, Reka Daniel, Vivian DeWoskin, and Yael Niv. Dynamic interaction between reinforcement learning and attention in multidimensional environments. *Neuron*, 93(2):451–463, 2017.

[15] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[16] Yuanguo Lin, Yong Liu, Fan Lin, Lixin Zou, Pengcheng Wu, Wenhua Zeng, Huanhuan Chen, and Chunyan Miao. A survey on reinforcement learning for recommender systems. *IEEE Transactions on Neural Networks and Learning Systems*, 35(10):13164–13184, 2023.

[17] Yael Niv, Reka Daniel, Andra Geana, Samuel J Gershman, Yuan Chang Leong, Angela Radulescu, and Robert C Wilson. Reinforcement learning in multidimensional environments relies on attention mechanisms. *Journal of Neuroscience*, 35(21):8145–8157, 2015.

[18] Deepak Kumar Panda and Sanjog Ray. Approaches and algorithms to mitigate cold start problems in recommender systems: a systematic literature review. *Journal of Intelligent Information Systems*, 59(2): 341–366, 2022.

[19] Sandeep Pandey, Deepak Agarwal, Deepayan Chakrabarti, and Vanja Josifovski. Bandits for taxonomies: A model-based approach. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 216–227. SIAM, 2007.

[20] Yunzhe Qi, Yao Zhou, Yikun Ban, Allan Stewart, Chuanwei Ruan, Jiachuan He, Shishir Kumar Prasad, Haixun Wang, and Jingrui He. Bi-level hierarchical neural contextual bandits for online recommendation. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=k3XsA75SGv. J2C Certification.

[21] Angela Radulescu, Yeon Soon Shin, and Yael Niv. Human representation learning. *Annual review of neuroscience*, 44(1):253–273, 2021.

[22] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[23] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyYe6k-CW.

[24] Paul Rolland, Jonathan Scarlett, Ilija Bogunovic, and Volkan Cevher. High-dimensional bayesian optimization via additive models with overlapping groups. In *International conference on artificial intelligence and statistics*, pages 298–307. PMLR, 2018.

[25] 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.

[26] Nicollas Silva, Thiago Silva, Heitor Werneck, Leonardo Rocha, and Adriano Pereira. User cold-start problem in multi-armed bandits: When the first recommendations guide the user's experience. *ACM Transactions on Recommender Systems*, 1(1):1–24, 2023.

[27] Aleksandrs Slivkins. Contextual bandits with similarity information. In *Proceedings of the 24th annual Conference On Learning Theory*, pages 679–702. JMLR Workshop and Conference Proceedings, 2011.

[28] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867, 2020.

[29] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[30] Liang Tang, Yexi Jiang, Lei Li, Chunqiu Zeng, and Tao Li. Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 323–332, 2015.

[31] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[32] C. J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, 1989.

[33] Yisong Yue, Sue Ann Hong, and Carlos Guestrin. Hierarchical exploration for accelerating contextual bandits. In *Proceedings of the 29th International Conference on Machine Learning*, ICML'12, page 979–986, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

[34] Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

[35] Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural thompson sampling. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=tkAtoZkcUnm.

[36] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 167–176, 2018.

[37] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International conference on machine learning*, pages 11492–11502. PMLR, 2020.

[38] Yinglun Zhu, Dylan J Foster, John Langford, and Paul Mineiro. Contextual bandits with large action spaces: Made practical. In *International Conference on Machine Learning*, pages 27428–27453. PMLR, 2022.

[39] Juliusz Krzysztof Ziomek and Haitham Bou Ammar. Are random decompositions all we need in high dimensional bayesian optimisation? In *International Conference on Machine Learning*, pages 43347–43368. PMLR, 2023.

A Full Problem Formulation, Exploration Schedule, and Evaluation Protocol

A.1 Recommendation Formulation and Context Construction

We formulate recommendation as a contextual-bandit problem with partial feedback. Let $\mathcal{I}$ be the item catalog and $\mathcal{U}$ the user set. At round t, the agent observes a user representation $u_t \in \mathbb{R}^{50}$, recommends one item a_t, and observes feedback only for that item.

For offline evaluation, let $\mathcal{I}_t^{\text{eval}} = \mathcal{I}_t^+$ contain the items with recorded feedback for the current user. We do not add unobserved user–item pairs to this set or impute rewards for missing feedback. Before selecting an item, an agent observes the item contexts but not their logged rewards; after selecting a_t, only its normalized logged reward is revealed. Both the flat baselines and TDGE make the final decision over at most $K = 10$ item arms. Items remain available after selection, as in the standard repeated-arm contextual-bandit setting.

The representation of an action z—a feature dimension, feature, or item—is denoted by $v_z \in \mathbb{R}^{50}$. Its contextual representation is

$$x_{z,t} = \text{norm}\left(u_t \oplus v_z \oplus 0.01\right) \in \mathbb{R}^{101}, \tag{5}$$

where $\oplus$ denotes concatenation and norm denotes ℓ_2 normalization. Item and feature representations are obtained by projecting their semantic embeddings to 50 dimensions. A dimension representation is the normalized centroid of the projected features assigned to that dimension. This common context construction allows the same contextual-bandit backbone to operate at every TDGE level.

A.2 Multi-Level Selection and Candidate Construction

Let $\mathcal{C}$ be the set of feature dimensions and C_c the features in dimension c. At round t, the dimension-level agent retains

$$\mathcal{D}_t = \text{TopK}_{c\in\mathcal{C}}\left(s_{c,t}^{(\text{dim})}, K_1\right). \tag{6}$$

The feature-level agent then forms

$$\mathcal{G}_t = \bigcup_{c\in\mathcal{D}_t} C_c \tag{7}$$

and retains

$$\mathcal{F}_t = \text{TopK}_{j\in\mathcal{G}_t}\left(s_{j,t}^{(\text{feat})}, K_2\right). \tag{8}$$

Let $q_{t,j} = s_{j,t}^{(\text{feat})}$ be the score of selected feature j, $\rho_{i,j}$ the relevance of feature j to item i, and $\text{feat}(i)$ the feature set of item i. If continuous relevance is unavailable, $\rho_{i,j} = 1$ for $j \in \text{feat}(i)$. The eligible item set is

$$\mathcal{E}_t = \left\{i \in \mathcal{I}_t^{\text{eval}} : \mathcal{F}_t \cap \text{feat}(i) \neq \emptyset\right\}. \tag{9}$$

TDGE scores each eligible item by

$$S_t(i) = \sum_{j\in\mathcal{F}_t\cap\text{feat}(i)} q_{t,j}\rho_{i,j}, \tag{10}$$

and retains

$$\mathcal{P}_t = \text{TopK}_{i\in\mathcal{E}_t}\left(S_t(i), K\right). \tag{11}$$

If $\mathcal{E}_t$ is empty, TDGE resamples the dimension–feature route before making a recommendation rather than introducing an item outside the selected semantic route. The item-level bandit then selects a_t from $\mathcal{P}_t$.

A.3 Reward, Regret, and Hierarchical Credit Assignment

The selected item receives reward

$$r_t = h(x_{a_t,t}) + \xi_t, \tag{12}$$

where $h : \mathbb{R}^{101} \to [0, 1]$ is the unknown expected reward and ξ_t is zero-mean conditionally sub-Gaussian noise. In the offline evaluation, the oracle is the highest-reward observed item,

$$a_t^* = \arg\max_{i\in\mathcal{I}_t^+} h(x_{i,t}), \tag{13}$$

and cumulative regret is

$$R_T = \sum_{t=1}^{T} \left[h(x_{a_t^*,t}) - h(x_{a_t,t}) \right]. \tag{14}$$

The item agent is updated at every completed round. Credit is propagated to the upper levels only through features shared by the selected route and the recommended item:

$$\mathcal{H}_t = \mathcal{F}_t \cap \mathrm{feat}(a_t). \tag{15}$$

Each $j \in \mathcal{H}_t$ is updated with sample weight $\rho_{a_t,j}$, and each source dimension c is updated with

$$\omega_{a_t,c} = \max_{j \in \mathcal{H}_t \cap C_c} \rho_{a_t,j}. \tag{16}$$

Thus, the hard intersection determines which upper-level actions receive credit, while ρ provides graded credit among matched actions. For MovieLens, ρ is the genome relevance; for Last.fm and Amazon it is one. Because the item pool contains only items matched by at least one selected feature, $\mathcal{H}_t$ is nonempty for every completed recommendation round.

B Task-Dimension Extraction with KGS-Based Clustering

Let $\mathcal{F} = \{f_1, \ldots, f_M\}$ be the descriptive feature set and $e_j \in \mathbb{R}^{768}$ the normalized Sentence-BERT embedding of feature f_j. Ward hierarchical clustering produces a dendrogram over $\mathcal{F}$. A cut yielding k clusters defines $\Pi_k = \{C_1^{(k)}, \ldots, C_k^{(k)}\}$. We select k with the constrained Kelley–Gardner–Sutcliffe (KGS) criterion.

For partition Π_k, its within-cluster sum of squares is

$$\mathcal{W}(k) = \sum_{c=1}^{k} \sum_{f_j \in C_c^{(k)}} \left\| e_j - \mu_c^{(k)} \right\|_2^2, \qquad \mu_c^{(k)} = \frac{1}{|C_c^{(k)}|} \sum_{f_j \in C_c^{(k)}} e_j. \tag{17}$$

Over candidate values $\mathcal{K}$, the KGS penalty is

$$\mathrm{KGS}(k) = \frac{\mathcal{W}(k) - \mathcal{W}_{\min}}{\mathcal{W}_{\max} - \mathcal{W}_{\min}} + \frac{k - k_{\min}}{k_{\max} - k_{\min}}. \tag{18}$$

The first term favors compact clusters and the second penalizes excessive fragmentation. A cut is valid only if every cluster contains at least $\eta = 5$ features. The selected number of dimensions is

$$k^* = \arg\min_{k \in \Omega} \mathrm{KGS}(k), \qquad \Omega = \{k \in \mathcal{K} : |C_c^{(k)}| \geq 5 \ \forall c\}. \tag{19}$$

We search $k \in \{10, \ldots, 99\}$ for all three datasets. This procedure yields 47, 37, and 35 fine-grained task dimensions for MovieLens-20M, Last.fm, and Amazon, respectively.

C Value Estimators and Full TDGE Procedure

TDGE uses the same top-down routing procedure with different contextual-bandit backbones. For level $\ell \in \{\mathrm{dim}, \mathrm{feat}, \mathrm{item}\}$, let (x, r, ω) denote an update observation. Item updates use $\omega = 1$; matched feature and dimension updates use the relevance weights defined above. It should be noted that our NeuralUCB and NeuralTS implementations estimate uncertainty using penultimate-layer features rather than full-network gradients. The same variants are used for the item-level TDGE and baselines.

LinUCB At level ℓ, LinUCB maintains $P_t^{(\ell)} = (A_t^{(\ell)})^{-1}$ and $b_t^{(\ell)}$, with $\widehat{\theta}_t^{(\ell)} = P_t^{(\ell)} b_t^{(\ell)}$. The score of action z is

$$s_{z,t}^{(\ell)} = (x_{z,t}^{(\ell)})^\top \widehat{\theta}_t^{(\ell)} + \alpha_\ell \sqrt{(x_{z,t}^{(\ell)})^\top P_t^{(\ell)} x_{z,t}^{(\ell)}}. \tag{20}$$

For (x, r, ω), the weighted update is

$$P^{(\ell)} \leftarrow P^{(\ell)} - \frac{\omega P^{(\ell)} x x^\top P^{(\ell)}}{1 + \omega x^\top P^{(\ell)} x}, \qquad b^{(\ell)} \leftarrow b^{(\ell)} + \omega r x. \tag{21}$$

Within each round, the feature agent is updated once per matched feature, and the dimension agent once per matched source dimension, with updates applied sequentially within each agent.

NeuralUCB NeuralUCB maintains a reward network $f_{\theta_t^{(\ell)}}$ and an inverse uncertainty matrix $P_t^{(\ell)}$. Let $\phi_{\theta_t^{(\ell)}}(x)$ be its penultimate-layer feature. The action score is

$$s_{z,t}^{(\ell)} = f_{\theta_t^{(\ell)}}(x_{z,t}^{(\ell)}) + \alpha_\ell \sqrt{\phi_{\theta_t^{(\ell)}}(x_{z,t}^{(\ell)})^\top P_t^{(\ell)} \phi_{\theta_t^{(\ell)}}(x_{z,t}^{(\ell)})}. \tag{22}$$

Observations are stored in a level-specific replay buffer and optimized with weighted squared loss,

$$\mathcal{L}^{(\ell)}(\theta) = \frac{1}{|\mathcal{B}^{(\ell)}|} \sum_{(x_\tau, r_\tau, \omega_\tau) \in \mathcal{B}^{(\ell)}} \omega_\tau \big(f_\theta(x_\tau) - r_\tau\big)^2. \tag{23}$$

After updating the network, $P_t^{(\ell)}$ is updated by the same weighted Sherman–Morrison rule using the new penultimate-layer feature.

NeuralTS NeuralTS uses the same reward network, replay training, and uncertainty matrix, but samples

$$s_{z,t}^{(\ell)} \sim \mathcal{N}\Big(f_{\theta_t^{(\ell)}}(x_{z,t}^{(\ell)}), \alpha_\ell^2 \phi_{\theta_t^{(\ell)}}(x_{z,t}^{(\ell)})^\top P_t^{(\ell)} \phi_{\theta_t^{(\ell)}}(x_{z,t}^{(\ell)})\Big) \tag{24}$$

and selects the action with the largest sampled value.

Algorithm 1 Three-level TDGE used in the main recommendation experiments

Require: Dimensions $\mathcal{C} = \{C_1, \ldots, C_{k^*}\}$; agents $\mathcal{A}^{(\mathrm{dim})}$, $\mathcal{A}^{(\mathrm{feat})}$, $\mathcal{A}^{(\mathrm{item})}$; budgets K_1, K_2, K.

for $t = 1, \ldots, T$ **do**
  Observe u_t and construct $\mathcal{I}_t^{\mathrm{eval}}$.
  Construct dimension contexts using (5) and score them with the dimension-level agent; retain the top K_1 dimensions $\mathcal{D}_t$.
  Score the features in $\bigcup_{c \in \mathcal{D}_t} C_c$; retain the top K_2 features $\mathcal{F}_t$ and their scores $q_{t,j}$.
  Compute $S_t(i)$ by (10) for evaluation candidates having nonempty overlap with $\mathcal{F}_t$.
  if at least one overlapping candidate exists **then**
    Retain the top K scored items as $\mathcal{P}_t$.
  else
    Resample the dimension–feature route before making a recommendation; do not add an item outside the selected route.
  end if
  The item agent recommends $a_t = \arg\max_{i \in \mathcal{P}_t} s_{i,t}^{(\mathrm{item})}$ and observes r_t.
  Update the item agent with $(x_{a_t,t}^{(\mathrm{item})}, r_t, 1)$.
  Set $\mathcal{H}_t = \mathcal{F}_t \cap \mathrm{feat}(a_t)$.
  for $j \in \mathcal{H}_t$ **do**
    Update the feature agent with $(x_{j,t}^{(\mathrm{feat})}, r_t, \rho_{a_t,j})$.
  end for
  for $c \in \mathcal{D}_t$ such that $\mathcal{H}_t \cap C_c \neq \varnothing$ **do**
    Update the dimension agent with $(x_{c,t}^{(\mathrm{dim})}, r_t, \max_{j \in \mathcal{H}_t \cap C_c} \rho_{a_t,j})$.
  end for
end for

D Recommendation Datasets and Preprocessing

Table 5: Recommendation-dataset statistics after preprocessing.

Dataset	# Users	# Items	# Features	# Dimensions (k^*)	# Interactions	Sparsity
MovieLens-20M	5,000	10,000	1,128	47	5,068,810	89.86%
Last.fm	1,892	10,000	2,074	37	84,365	99.55%
Amazon	5,000	43,724	5,743	35	494,083	99.77%

MovieLens-20M We retain the 10,000 most frequently rated movies with tag-genome coverage and the 5,000 most active users. Ratings are divided by five. For hierarchical routing and credit assignment, each movie is associated with its ten highest-relevance genome tags. Its semantic item representation is the relevance-weighted average of its 50 highest-relevance tag embeddings before PCA projection.

Last.fm We retain the 10,000 most frequently interacted artists with tag annotations and up to 2,000 active users, resulting in 1,892 users after preprocessing. Listening counts are transformed by $\log(1 + x)$ and scaled by each user's maximum to $[0.1, 1.0]$. Tags used at most five times globally are removed. Artist–tag relevance is binary, and up to ten associated tags define the route and credit set of each artist. The semantic artist representation averages up to 50 associated tag embeddings.

Amazon We use the Beauty and Personal Care subset, retain the 5,000 most active users, and keep products with at least five interactions among those users. We extract informative metadata key–value pairs and convert each pair into a textual feature such as "Skin Type: Oily." Features occurring on fewer than five products are removed, and each product retains up to five features. Ratings are divided by five, and a product representation is the unweighted average of its retained feature embeddings.

Shared semantic preprocessing Feature strings are encoded with sentence-transformers(all-mpnet-base-v2 [28]) and ℓ_2 normalized. Clustering is performed in the original 768-dimensional embedding space. Item embeddings are then reduced to 50 dimensions with PCA; the same PCA transform is applied to feature embeddings, and dimension arms are normalized feature-centroid vectors in this shared space. Users are split before factorization, and 50-dimensional truncated-SVD representations are fitted using only the online-training users. Each evaluated cold-start user is initialized with the same normalized mean training-user representation, so no user-specific interaction history is included in its context.

Offline evaluation-environment construction. For each seed, retained users are split into online-training and held-out cold-start pools with a 9:1 ratio before fitting the user representation model. The truncated-SVD model is fitted only to the online-training users. For cold-start evaluation, 100 held-out users are sampled, and each begins an independent 10-step trajectory with the same normalized mean training-user representation. The held-out users' logged interactions are used only by the offline evaluator to define the recorded-feedback decision sets and rewards; they are not used to fit the representation model or construct user contexts.

For user u_t, the evaluation decision set $\mathcal{I}_t^{\mathrm{eval}}$ contains the items with recorded feedback for that user. Missing user–item pairs are excluded rather than treated as negative feedback. The logged reward of an item is hidden until that item is selected. All compared methods use the same preprocessing, user splits, recorded-feedback decision sets, reward definition, random seeds, interaction horizons, and final item budget.

Fair comparison of candidate generation. All methods use the same evaluation decision sets, contexts, rewards, interaction horizons, and final item-level budget of $K = 10$. The item-level agent in each TDGE variant has exactly the same architecture and hyperparameters as its corresponding flat baseline. A flat baseline uniformly samples up to K items from $\mathcal{I}_t^{\mathrm{eval}}$ for item-level scoring, whereas TDGE uses its learned dimension–feature route to construct the K-item pool. Candidate generation is part of the TDGE algorithm and is therefore included in end-to-end regret. The NoFD ablation provides a feature-only semantic retrieval control with the same downstream retrieval, item-level selection, and update rules, thereby isolating the additional contribution of learned dimension-level routing.

This design directly evaluates TDGE's central advantage. In a large item catalog, uniform sampling is unlikely to retrieve high-reward items within a limited item-scoring budget. TDGE instead narrows the catalog through learned user preferences at the semantic dimension and feature levels, enabling the item-level agent to evaluate more relevant candidates and improving sample efficiency.

Asset provenance and usage terms. MovieLens-20M is used under the GroupLens research-use terms, and the HetRec 2011 Last.fm dataset is used under its non-commercial research terms. Amazon Reviews 2023 is obtained from the official McAuley Lab research release. We do not redistribute the raw third-party datasets. The `sentence-transformers/all-mpnet-base-v2` model is released

under the Apache License 2.0. All baseline methods are properly cited. CoFineUCB is independently implemented from its published algorithmic description, whereas our H_2N-Bandit baseline is adapted from the implementation publicly released by its original authors. All external assets are credited and used in accordance with their applicable terms.

E Experimental Configuration

All main recommendation results use five independent runs with seeds 2026–2030. Statistical significance is assessed on run-level final metrics using two-sided Welch t-tests.

Table 6: Shared recommendation hyperparameters.

Parameter	Value
Online rounds (T)	10,000
Independent runs	5
Final item budget (K)	10
Train/cold-start user split	90%/10%
Cold-start users per run	100
Cold-start steps per user	10
Recommendation context dimension	101
Semantic embedding dimension after PCA	50
Base random seeds	2026–2030

Hyperparameter selection Hyperparameters are selected separately for each dataset–backbone pair. For the flat backbones, we tune the exploration coefficient α and regularization parameter λ by grid search over

$$\alpha \in \{0.01, 0.1, 1, 10\}, \qquad \lambda \in \{0.01, 0.1, 1, 10\}.$$

For TDGE, the dimension- and feature-level values of α and λ are selected from the same grids. The retention budgets are selected from $K_1 \in \{1, 2, 3, 5, 8\}$ and $K_2 \in \{4, 6, 10, 16, 24\}$.

To ensure a controlled comparison at the item level, we first tune each flat backbone and then fix its selected (α, λ) pair for the item-level agent of the corresponding TDGE variant. The item-level agent also uses exactly the same architecture, optimizer, learning rate, batch size, replay-buffer size, and update schedule as the corresponding flat baseline. Only the dimension- and feature-level agents and their retention budgets are specific to TDGE. Thus, performance differences cannot be attributed to a stronger item-level reward estimator in TDGE.

Table 7: Selected dimension and feature retention budgets (K_1, K_2).

Method	MovieLens-20M	Last.fm	Amazon
TDGE-LinUCB	$(5, 10)$	$(2, 4)$	$(1, 6)$
TDGE-NeuralUCB	$(5, 10)$	$(3, 10)$	$(3, 24)$
TDGE-NeuralTS	$(5, 10)$	$(1, 24)$	$(5, 24)$

Table 8: Selected exploration and regularization parameters (α, λ) for the flat backbones.

Backbone	MovieLens-20M	Last.fm	Amazon
LinUCB	$(0.01, 1)$	$(0.01, 1)$	$(1, 0.01)$
NeuralUCB	$(0.01, 0.1)$	$(0.01, 0.01)$	$(1, 0.1)$
NeuralTS	$(0.1, 10)$	$(0.01, 10)$	$(0.1, 10)$

Neural network configuration All neural agents use a two-hidden-layer feedforward network $d \to 128 \to 128 \to 1$ with ReLU activations, Adam with learning rate 10^{-3}, ten gradient steps per update, and a replay buffer containing at most 2,000 observations. The dimension- and feature-level agents use a batch size of 128, whereas the flat baselines and the item-level agents of TDGE use a

Table 9: Selected TDGE parameters. Each cell lists $(\alpha, \lambda)_{\mathrm{dim}}/(\alpha, \lambda)_{\mathrm{feat}}/(\alpha, \lambda)_{\mathrm{item}}$. For each dataset–backbone pair, the item-level parameters are identical to those of the corresponding flat backbone in Table 8.

Method	MovieLens-20M	Last.fm	Amazon
TDGE-LinUCB	$(1, 1)/(1, 1)/(0.01, 1)$	$(0.1, 1)/(1, 0.1)/(0.01, 1)$	$(1, 1)/(0.1, 1)/(1, 0.01)$
TDGE-NeuralUCB	$(0.1, 1)/(0.1, 0.1)/(0.01, 0.1)$	$(1, 10)/(0.1, 10)/(0.01, 0.01)$	$(0.1, 10)/(0.1, 10)/(1, 0.1)$
TDGE-NeuralTS	$(0.1, 1)/(0.1, 1)/(0.1, 10)$	$(0.1, 10)/(0.1, 10)/(0.01, 10)$	$(0.1, 0.1)/(0.1, 0.1)/(0.1, 10)$

batch size of 64. When the replay buffer contains fewer observations than the specified batch size, all available observations are used. Therefore, within each dataset–backbone pair, TDGE and its corresponding baseline use the same item-level model and training configuration.

Structured baselines The matched MovieLens comparison includes CoFineUCB and H_2N-Bandit. CoFineUCB uses a five-dimensional coarse parameter subspace learned only from training-user preference profiles; its fine and coarse exploration coefficients and regularization parameters match the LinUCB configuration. H_2N-Bandit uses the raw MovieLens genre taxonomy, retains its top four genre categories, and applies its neural item selector to at most ten items. All compared methods use the same five seeds, user split, interaction horizon, initial candidate-sampling protocol, item-level context representations, and final item budget.

Compute resources All experiments were conducted on a single workstation equipped with a 13th Gen Intel Core i9-13900K CPU, 128 GB of system memory, and one NVIDIA GeForce RTX 4090 GPU with 24 GB of memory. No distributed or multi-GPU training was used. For an online horizon of $T = 10{,}000$, a single run required approximately one minute for LinUCB-based methods and seven minutes for NeuralUCB- and NeuralTS-based methods. Reproducing the complete set of reported experiments required approximately three hours of wall-clock time. The preprocessed datasets occupy approximately 1.2 GB of storage.

F Additional ablation studies

F.1 Importance of the Feature-Dimension Level

The NoFD ablation serves as a feature-only semantic candidate-generation control. It removes the dimension-level agent while retaining the same feature-level agent, semantic candidate scoring, item retrieval, item-level agent, and update procedure as full TDGE. At each round, it uniformly samples 50 features from the full feature vocabulary, applies the same feature-level agent to retain the top K_2 features, and then uses the same candidate-generation, item-selection, and update procedures as full TDGE.

Table 10: Online cumulative regret and cold-start final-round regret for the feature-dimension-level ablation on MovieLens-20M. Lower is better. Superscripts compare full TDGE with NoFD using two-sided Welch t-tests.

Metric	Backbone	NoFD	Full TDGE
Online CReg	LinUCB	1885.934 ± 33.885	$\mathbf{1573.142 \pm 38.906}^{***}$
	NeuralUCB	2059.076 ± 29.659	$\mathbf{1669.303 \pm 81.538}^{***}$
	NeuralTS	2074.051 ± 15.427	$\mathbf{1705.542 \pm 47.198}^{***}$
Cold-start Regret ($t = 10$)	LinUCB	0.170 ± 0.014	$\mathbf{0.080 \pm 0.011}^{***}$
	NeuralUCB	0.173 ± 0.015	$\mathbf{0.060 \pm 0.014}^{***}$
	NeuralTS	0.175 ± 0.008	$\mathbf{0.078 \pm 0.023}^{***}$

Removing dimension-level routing consistently increases both online cumulative regret and cold-start final-round regret across all three backbones, showing that TDGE's gains cannot be explained by feature-level semantic retrieval alone and specifically support the contribution of learned dimension-level routing.

G Regret Trajectories

Figure 5 reports the complete online cumulative-regret trajectories. TDGE reduces cumulative regret for all three backbones on MovieLens-20M and for both neural backbones on Last.fm and Amazon. TDGE-LinUCB remains comparable to LinUCB on Amazon but does not improve over it on Last.fm, consistent with the endpoint comparisons in Table 1. On MovieLens-20M, the corresponding TDGE variants also maintain lower regret than CoFineUCB and H_2N-Bandit throughout online training.

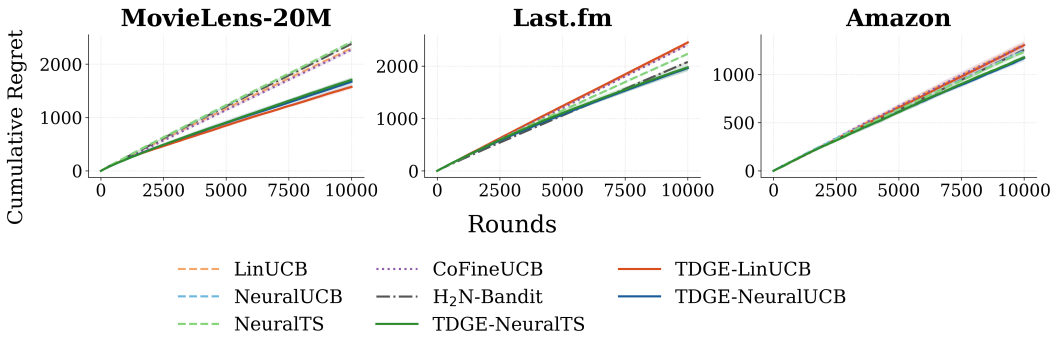


Figure 5: Online cumulative regret over 10,000 interaction rounds. Curves show the mean over five runs, and shaded regions denote one standard deviation. Dashed curves are the item-level baselines, and solid curves are the corresponding TDGE variants. Dotted and dash-dotted curves denote CoFineUCB and H_2N-Bandit, respectively, where results are available. Lower is better.

Figure 6 shows cumulative regret over the 10-step zero-history cold-start trajectory. TDGE accumulates substantially less regret for all three backbones on MovieLens-20M. On Last.fm, all TDGE variants also end with numerically lower cumulative regret. On Amazon, the neural TDGE variants reduce cumulative regret, while TDGE-LinUCB is comparable to LinUCB, likely due to the limited expressiveness of the linear reward model for capturing complex user–item interactions, as we mentioned in previous Section 4.2. For completeness, we also report the regret trajectories of CoFineUCB and H_2N-Bandit, further supporting the structured-baseline results reported in Section 4.3. The main text reports regret at the tenth interaction to measure recommendation quality at the end of adaptation; the cumulative trajectories additionally show the regret incurred throughout the adaptation process.

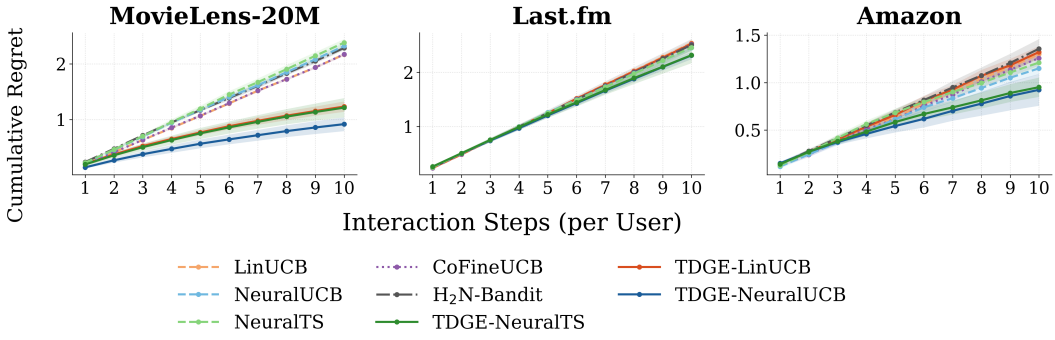


Figure 6: Cold-start cumulative regret over 10 interactions per held-out user. Within each run, regret at each step is averaged over 100 users before accumulation. Curves show the mean over five runs, and shaded regions denote one standard deviation. Dashed curves are the item-level baselines, and solid curves are the corresponding TDGE variants. Dotted and dash-dotted curves denote CoFineUCB and H_2N-Bandit, respectively, where results are available. Lower is better.

Figure 7 provides the corresponding per-step view of cold-start adaptation. Each point is the average instantaneous regret at that interaction step, and the point at step 10 is exactly the final-round regret reported in Table 1. The trajectories verify that the final-round differences appear directly in

instantaneous regret rather than being induced by cumulative aggregation. The reduction is most pronounced across all three backbones on MovieLens-20M and for the neural backbones on Amazon; the Last.fm endpoints also favor the corresponding TDGE variants.

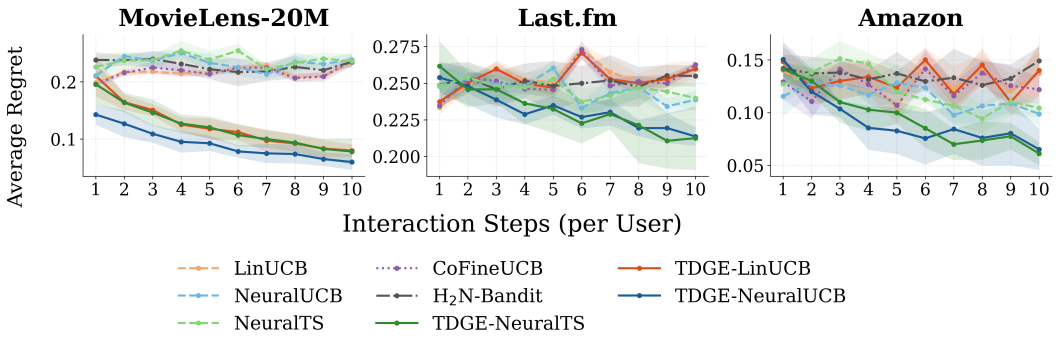


Figure 7: Average instantaneous regret at each step of the 10-step cold-start trajectory. At each step, regret is averaged over 100 held-out users within each run. Curves show the mean over five runs, and shaded regions denote one standard deviation. The values at step 10 correspond to the final-round regret reported in Table 1. Lower is better.

H Inference-Time Analysis

TDGE introduces dimension- and feature-level scoring before item selection. To quantify the resulting overhead at a large arm-set size, we benchmark neural inference on an NVIDIA GeForce RTX 4090 using 101-dimensional synthetic contexts. Both the flat backbone and its TDGE variant score 100,000 item arms per decision; TDGE additionally scores 5,000 dimension arms and 5,000 feature arms. We use 50 warm-up passes followed by 150 timed passes and report median and 95th-percentile latency. Context tensors are constructed before timing and retained on the GPU, so the benchmark measures model scoring and action selection rather than data loading.

Table 11: Per-decision neural inference latency with 100,000 item arms. TDGE additionally scores 5,000 dimension arms and 5,000 feature arms. Measurements use an NVIDIA GeForce RTX 4090 after 50 warm-up passes; 150 passes are timed.

Method	Median (ms)	P95 (ms)
NeuralUCB	1.6716	1.7666
TDGE-NeuralUCB	1.9353	1.9538
NeuralTS	1.7218	1.7504
TDGE-NeuralTS	2.0404	2.0606

Despite the two additional routing stages, TDGE adds only 0.2637 ms for NeuralUCB and 0.3186 ms for NeuralTS at the median, while the complete scored decision in this benchmark remains close to 2 ms. This benchmark is conservative with respect to TDGE's intended deployment: it assigns the same 100,000 item arms to the item-level model for both methods and therefore does not credit TDGE for the computation saved by narrowing the item pool before final scoring. Moreover, the dimension and feature arm sets are determined by the metadata vocabulary and do not scale directly with the catalog size. These results show that TDGE introduces modest scoring overhead even for a large item set and support the practical scalability of its semantic routing in large recommendation systems.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction describe TDGE as a human-inspired, model-agnostic framework that performs dimension–feature–item exploration and improves online training and cold-start adaptation. Section 3 specifies the framework; Tables 1 and 2, Section 4.4, and Appendix H support the performance, robustness, and efficiency claims; and Section 4.5 reports dimension-persistent trajectories while explicitly avoiding claims of equivalence to human cognitive mechanisms.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 5 discusses three limitations: the fixed retention sizes K_1 and K_2, the possibility that the observed effect of hierarchy depth depends on dataset structure, and the current restriction of the evaluation to contextual-bandit backbones rather than general reinforcement learning.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not claim a formal regret bound or convergence theorem. Section 3 and Appendix C use mathematical expressions to define TDGE's selection, scoring, and update rules rather than to state theoretical results requiring proof.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 and Appendix C provide the selection and update equations and full pseudocode in Algorithm 1. Appendix D documents preprocessing and the offline evaluation environment, while Appendix E reports the random seeds, grid-search spaces, selected retention budgets, model configurations, and selected backbone parameters in Tables 6–9.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: MovieLens-20M, Last.fm-2K, and Amazon Reviews are publicly accessible datasets, and Appendix D documents their preprocessing. The repository at `https://github.com/LordSlime/Human-inspired-Task-Dimension-Guided-Exploration-for-Efficient-Learning` provides preprocessing, training, metric-computation, and plotting code together with reproduction instructions.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 specifies the datasets, baselines, evaluation metrics, and paired-comparison protocol. Appendices D and E give the user split, zero-history cold-start construction, interaction horizons, candidate budgets, representations, network and optimizer settings, random seeds, grid-search spaces, and selected hyperparameters.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Tables 1 and 2 report mean $\pm$ sample standard deviation over five independent seeds, and significance is assessed on run-level final metrics using two-sided Welch t-tests. Appendix G additionally plots mean regret trajectories with one-standard-deviation bands.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix E reports the CPU, GPU, system memory, and GPU memory used for the experiments, together with separate per-run wall-clock training times for the reported experiments. Appendix H separately reports the hardware and measurement protocol used for the inference-time benchmark.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: The experiments use established public recommendation datasets described in Appendix D and do not collect new human-subject data. Section 3.1 uses a separately published behavioral study only as scientific motivation; no participant-level data from that study are used or analyzed in this work.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: TDGE may improve sample efficiency and reduce low-quality recommendations during new-user adaptation (Section 4.2); its explicit dimension-level routing may also aid interpretation. As with other personalized recommenders, deployment could inherit bias from interaction data and metadata or reinforce narrow content exposure, so fairness, diversity, and privacy should be evaluated in the intended application.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release a pretrained generative model, a newly scraped dataset, or another asset with elevated misuse potential. It releases algorithmic code and uses an existing Sentence-BERT/MPNet encoder only to embed item-descriptive features, as documented in Appendices D and B.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper and supplementary materials identify the sources and applicable usage terms of the datasets and pretrained encoder. All baseline methods are cited. CoFineUCB is independently implemented from its published description, while H_2N-Bandit is adapted from the implementation publicly released by its original authors, whose source is explicitly provided. Raw third-party datasets are not redistributed.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The submission releases the TDGE implementation as a new code asset through the anonymous repository linked in the abstract. Its README documents the environment, dataset preparation, training commands, evaluation scripts, result aggregation, and figure generation; no new dataset or pretrained model is released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not conduct crowdsourcing or new human-subjects research and does not use participant-level data from the behavioral study cited for motivation in Section 3.1. All experiments reported in this paper are computational evaluations on existing recommendation datasets.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not conduct human-subjects research or analyze participant-level data. The behavioral study cited in Section 3.1 is prior work used only to motivate the algorithm and is outside the experimental scope of this paper.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are not an important, original, or non-standard component of TDGE. The paper uses the off-the-shelf all-mpnet-base-v2 sentence encoder to embed feature strings during preprocessing (Appendices D and B), rather than using an LLM as part of the proposed learning or decision-making method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.