Learning Probabilistic Box Embeddings for Effective and Efficient Ranking

Lang Mei
Beijing Key Laboratory of Big Data Management and Analysis Methods, Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
meilang2013@ruc.edu.cn

Jiaxin Mao*
Beijing Key Laboratory of Big Data Management and Analysis Methods, Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
maojiaxin@gmail.com

Gang Guo
Beijing Key Laboratory of Big Data Management and Analysis Methods, Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
guogang@ruc.edu.cn

Ji-Rong Wen
Beijing Key Laboratory of Big Data Management and Analysis Methods, Gaoling School of Artificial Intelligence, Renmin University of China
Beijing, China
jrwen@ruc.edu.cn

ABSTRACT

Ranking has been one of the most important tasks in information retrieval. With the development of deep representation learning, many researchers propose to encode both the query and items into embedding vectors and rank the items according to the inner product or distance measures in the embedding space. However, the ranking models based on vector embeddings may have shortages in effectiveness and efficiency. For effectiveness, they lack the intrinsic ability to model the diversity and uncertainty of queries and items in ranking. For efficiency, nearest neighbor search in a large collection of item vectors can be costly. In this work, we propose to use the recently proposed probabilistic box embeddings for effective and efficient ranking, in which queries and items are parameterized as high-dimensional axis-aligned hyper-rectangles. For effectiveness, we utilize probabilistic box embeddings to model the diversity and uncertainty with the overlapping relations of the hyper-rectangles, and prove that such overlapping measure is a kernel function which can be adopted in other kernel-based methods. For efficiency, we propose a box embedding-based indexing method, which can safely filter irrelevant items and reduce the retrieval latency. We further design a training strategy to increase the proportion of irrelevant items that can be filtered by the index. Experiments on public datasets show that the box embeddings and the box embedding-based indexing approaches are effective and efficient in two ranking tasks: ad hoc retrieval and product recommendation.

*Corresponding author.

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512073

KEYWORDS

Embeddings, Dense Retrieval, Probabilistic Box Embedding, Indexing, Ranking, Recommendation

ACM Reference Format:
Lang Mei, Jiaxin Mao, Gang Guo, and Ji-Rong Wen. 2022. Learning Probabilistic Box Embeddings for Effective and Efficient Ranking. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3485447.3512073

1 INTRODUCTION

The ranking task, such as document ranking, recommendation systems, is one of the basic tasks in information retrieval (IR). In general, the ranking model aims to match queries with items (i.e. documents, products) from a large set of candidate collections, and produce a ranking list by computing numeric scores to measure the relevance between the queries and items.

Recently, with the development of deep representation learning [8, 34], many researchers propose to encode queries and items into dense vector embeddings in a low-dimensional Euclidean space to better model the semantic relations between them. Based on the learned vector embeddings, the Euclidean distances or inner products between vector embeddings of queries and items can be utilized to measure their relevance in ranking. Previous work has demonstrated that dense vector embeddings-based methods achieve convincing performance on many IR-related tasks [21]. Furthermore, ANNS (Approximate Nearest Neighbor Search) algorithms [12, 17, 18] can leverage a precomputed ANNS index to efficiently retrieve the approximate top K items that are similar to a given query, from a large collection of items. However, vector embedding-based learning and indexing methods may have some problems in both effectiveness and efficiency, respectively. For effectiveness, vector embeddings (i.e. a single point in the embedding space) may be a suboptimal choice to model the semantic diversity and uncertainty of queries and items. Figure 1 shows an example in recommendation where the "queries" and "items" correspond to the user preference and products respectively. From this figure,

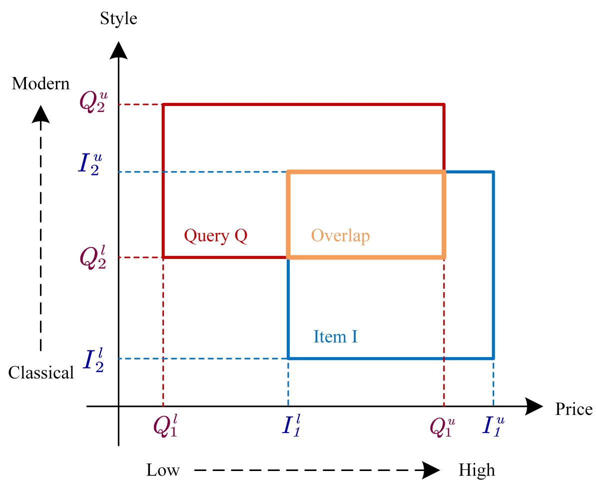

Figure 1: Two dimensional examples in recommendation to explain the diversity and uncertainty of queries and items.

we can see that a user may have some acceptable ranges of price and style for the products, and a product may also be associated with the price fluctuation and several styles due to its versatile attributes. The user usually chooses a product based on the partial matching of her preference and the item's attributes. Similarly, for ad hoc search, the short query can be ambiguous and be relevant to several subtopics, a document may cover diverse topics and thus be relevant to queries with disparate information needs. It is hard to model the semantic diversity and uncertainty of queries and items with a single point in the vector space. For efficiency, the exact nearest neighbor search in a high-dimensional vector space can be prohibitively time-consuming for a large corpus. Therefore, a variety of ANNS algorithms [7, 18, 25, 32] are utilized to improve the retrieval efficiency. However, as the ANNs algorithms can only retrieve the approximate top-k nearest neighbors, there is a trade-off between the speedup brought by the ANNS and the retrieval accuracy.

Recently, a series of geometric-inspired embedding models [6, 9, 22, 23, 28, 33, 35–37] have been proposed and drawn much attention from researchers. Probabilistic box embeddings [6, 23, 33, 36], which is a typical variants of these models, has demonstrated its superior representation ability to model the hierarchical [9, 28] and transitive relations [6, 22, 23, 33, 35, 36], entailments [23], and the uncertainty [37]. As the probabilistic box embedding model represents objects (i.e. queries, items) as multi-dimensional axis-aligned hyper-rectangles, we believe that it can improve the effectiveness and efficiency for ranking tasks. For effectiveness, from the example shown in Figure 1, we can see that after mapping the queries and items to box representation, probabilistic box embeddings can naturally model the semantic diversity and uncertainty of queries or items. The volume of the overlapping box between query box and item box can be use as an intuitive similarity measure for the queries and items. For efficiency, as shown in Figure 1, we can easily judge whether a query box and an item box have overlapping in space. If two boxes are disjoint with each other, it means that they are not relevant. After representing the query and documents as boxes, we can leverage this property to efficiently filter irrelevant items, which will reduce the time needed in searching for relevant items.

Therefore, in this work, we propose to improve effectiveness and efficiency for ranking by probabilistic box embeddings. First, we utilize the box embeddings to better represent queries and items in ranking tasks, and propose to use the volume of the overlapping box between the query and item as a similarity measure. We further prove that this similarity measure is in fact a kernel function which can be adopted in other kernel-based methods. Second, we incorporate two efficient-oriented constraints in the training process of box embeddings. These two constraints will encourage the learned box representations of the irrelevant items to be disjoint with the representation of the query. Third, after obtaining the learned box embeddings, we propose a box embedding-based indexing method, which can filter irrelevant items and reduce the visiting times without sacrificing the retrieval accuracy. We conduct experiments on real-world datasets of both passage ranking and recommendation tasks. Experimental results show that the proposed method can outperform existing vector embedding-based approaches in both effectiveness and efficiency.

2 RELATED WORK

In passage retrieval tasks, conventional algorithms such as BM25 [31] utilize the features from exact keyword matching, which can lead to the vocabulary mismatch problem if there exist different terms sharing the same meaning. Dense Retrieval model represents queries and items as vector embeddings during the online stage, and builds the document index during the offline stage. For the training process in the online stage, negative sampling methods are used to train dense retrieval models. Huang et al. [15] randomly samples negative documents from the whole corpus. Karpukhin et al. [21] adopts In-Batch negatives, which use other queries' positive documents in the same mini-batch as negatives. Gao et al. [11], Karpukhin et al. [21] utilizes top-retrieval documents from BM25 model as hard negatives. Xiong et al. [38] retrieve the top documents as hard negatives by using a warm-up dense retrieval model, and during training they refresh the document index to update the hard negatives, which are inferred by the current parameters of the model. Zhan et al. [40] proposes to consider both hard negatives and random negatives, which can better optimize for the ranking metrics. For the indexing process in the offline stage, ANNS (Approximate Nearest Neighbor Search) algorithms [12, 17] are utilized to build document index as pre-computed form, which can efficiently retrieve the approximate top K items given a query.

In recommendation tasks, two-tower architectures based on deep neural networks have been widely adopted in industrial recommender systems to capture personalized information [14, 16, 20]. After obtaining the learned vector embedding of user preferences and item features, the index can be constructed and the inner product is utilized to perform efficient searching. Huang et al. [16] learns the relevance based on the inner product between user features and item features. Hidasi et al. [14] applies GRU network to model the user interaction sequence. Kang and McAuley [20] adopts the self-attention mechanism to better capture the users' dynamic interests.

Recently, probabilistic box embeddings [6, 23, 33, 36] are proposed to model objects by high dimensional axis-aligned hyper-rectangles. While box embeddings have a good representation capacity, especially for transitive relations, it is difficult to optimize the box embeddings with the standard gradient descent approach. To improve the optimization of box embeddings, Li et al. [23] use Gaussian convolution to smooth the edges and thus avoid the zero gradient problem. Dasgupta et al. [6] utilizes Gumbel distribution to alleviate the problems of local identifiability. Motivated by the box embeddings, real-world applications, such as knowledge graphs [30], recommendation [26, 41], have been proposed. Ren et al. [30] adopt box embeddings for logical reasoning in knowledge graphs, and encode queries and entities as boxes. For the recommendation task Zhang et al. [41] proposes to represent the user as a multi-dimensional hypercuboid, the edges of the hypercuboid is used to describe the ranges of preferences, which enhance the representation capacity in capturing the diversity of preferences. Mei et al. [26] propose to embed users and items as high dimensional latent space, in which hypercuboid representation is a simplified variant.

3 BACKGROUND

In this section, we give a brief introduction of vector embeddings and probabilistic box embeddings.

3.1 Vector Embeddings

In the general settings of vector embedding, given n arbitrary objects $X_1, X_2, ..., X_n$, they are embedded as d-dimensional vectors $\boldsymbol{Y_1}, \boldsymbol{Y_2}, ..., \boldsymbol{Y_n}$ in space. When optimizing the vector embedding-based models, we require the pairwise distances $||\boldsymbol{Y_i} - \boldsymbol{Y_j}||^2$ or the inner product (or cosine similarity $\frac{\boldsymbol{Y_i^T Y_j}}{||\boldsymbol{Y_i}||||\boldsymbol{Y_j}||}$) $\boldsymbol{Y_i^T Y_j}$ to be approximately equal to the ground truth distances $d_X\left(X_i, X_j\right)$ or the similarity $sim_X(X_i, X_j)$ between X_i and X_j, respectively. After training the vector embedding model (i.e. a mapping function from X_i to $\boldsymbol{Y_i}$), we can use the distances $||\boldsymbol{Y_i} - \boldsymbol{Y_j}||^2$ or inner products $\boldsymbol{Y_i^T Y_j}$ between vector embeddings to model the similarity relation of between the objects X_i and Y_i, and then build the index based on these learned vector embeddings.

3.2 Probabilistic Box Embeddings

3.2.1 *Notion.*

In probabilistic box embeddings, given an object X, an d-dimensional axis-aligned hyper-rectangle (or box) is used to represent it, in which the parameters contain two vectors that correspond to the lower and upper boundaries of the box in d dimensions.

$$box\left(X\right) = \left\langle \left[X_1^l, X_1^u\right], ..., \left[X_d^l, X_d^u\right] \right\rangle \tag{1}$$

After associating the object X with a d-dimensional box, $box(X)$, the interval lengths of the d-dimensional boundaries are utilized to compute the volume of $box\left(X\right)$,

$$V\left(box\left(X\right)\right) = \prod_{k=1}^{d} \left(X_k^u - X_k^l\right) \tag{2}$$

Furthermore, given the box representations $box\left(A\right)$ and $box\left(B\right)$ of any two objects A and B, we can obtain the overlapping region, a d-dimensional box, $box\left(A \wedge B\right)$. Similarly, the lower and upper boundaries of $box\left(A \wedge B\right)$ can be calculated by:

$$\begin{aligned} box\left(A\right) \wedge box\left(B\right) = box\left(A \wedge B\right) = \\ = \langle A_1 \wedge B_1, ..., A_d \wedge B_d \rangle \end{aligned} \tag{3}$$

Specifically, $A_k \wedge B_k = \left[\max\left(A_k^l, B_k^l\right), \min\left(A_k^u, B_k^u\right)\right]$ are the lower and upper boundaries of $box\left(A \wedge B\right)$ for dimension k. If two boxes are disjoint, it means there always exist at least one dimension j such that $\max\left(A_j^l, B_j^l\right) > \min\left(A_j^u, B_j^u\right)$.

3.2.2 *Overlapping Volume.*

Based on the boundaries of the overlapping region $box\left(A \wedge B\right)$, we can also calculate the its volume. Noticing that $min(A_j^u, B_j^u)$ can be smaller than $max(A_j^l, B_j^l)$ but the volume of overlapping box should be non-negative. Therefore, we use the following formula to compute the volume of $box\left(A \wedge B\right)$:

$$\begin{aligned} V\left(box\left(A \wedge B\right)\right) = \prod_{k=1}^{d} \max\left(0, |A_k \wedge B_k|\right) = \\ \prod_{k=1}^{d} \max\left(0, \min\left(A_k^u, B_k^u\right) - \max\left(A_k^l, B_k^l\right)\right) \end{aligned} \tag{4}$$

Where $|\cdot|$ refers to the interval length in single dimension. We adopt the overlapping volume to model the relation between the object A and B. Specially, we prove the property of such overlapping volume by the following theorem, and the details are given in Appendix A.
Theorem 1. The overlapping volume function $V\left(box\left(A \wedge B\right)\right)$ is a kernel function.

3.2.3 *Learning Box Embeddings.*

It is challenging to directly optimize the above vanilla box embeddings because, when two boxes are disjoint, it would be difficult to make them overlap with each other by a gradient-based training method (i.e. the gradients with respect to this training pair would be zero.). Dasgupta et al. [6] introduce a random latent variable approach *GumbelBox* to overcome these problems, it models the box embedding parameters based on the assumption of independent Gumbel distribution, which allows all parameters to involve the gradient updating in different training situations.

In *GumbelBox*, based on the lower and upper boundaries of $box\left(A\right)$ and $box\left(B\right)$, Gumbel distributions are utilized to generate the lower and upper boundaries $\langle A_1 \wedge B_1, ..., A_d \wedge B_d \rangle$ of the overlapping box $box\left(A \wedge B\right)$.

$$f\left(x; \mu, \beta\right) = \frac{1}{\beta} \exp\left(-\frac{x - \mu}{\beta} - e^{-\frac{x-\mu}{\beta}}\right) \tag{5}$$

Where β is the temperature parameter of Gumbel distribution. For any dimension k, $A_k \wedge B_k = \left[\min\left(A_k \wedge B_k\right), \max\left(A_k \wedge B_k\right)\right]$ is given by:

$$\min\left(A_k \wedge B_k\right) \sim Gumbel\left(-\beta ln\left(e^{-\frac{A_k^l}{\beta}} + e^{-\frac{B_k^l}{\beta}}\right), \beta\right) \tag{6}$$

$$\max\left(A_k \wedge B_k\right) \sim Gumbel\left(\beta ln\left(e^{\frac{A_k^u}{\beta}} + e^{\frac{B_k^u}{\beta}}\right), \beta\right) \tag{7}$$

Through calculating the expected volumes of overlapping box $box\,(A \wedge B)$, the expected $A_k \wedge B_k$ can be obtained, and serve as the final estimation of $A_k \wedge B_k$,

$$\mu_k^- := E\left(\min\left(A_k \wedge B_k\right)\right) = -\beta LogSumExp\left(-\frac{A_k^l}{\beta}, -\frac{B_k^l}{\beta}\right) \quad (8)$$

$$\mu_k^+ := E\left(\max\left(A_k \wedge B_k\right)\right) = \beta LogSumExp\left(\frac{A_k^u}{\beta}, \frac{B_k^u}{\beta}\right) \quad (9)$$

$$\begin{aligned} V_g\left(box\left(A \wedge B\right)\right) &:= E\left(V\left(box\left(A \wedge B\right)\right)\right) \\ &= \prod_k \beta \log\left(1 + \exp\left(\frac{\mu_k^+ - \mu_k^-}{\beta} - 2\gamma\right)\right) \end{aligned} \quad (10)$$

Where γ is the Euler-Mascheroni constant.

4 LEARNING

In this section, we first give the formulation of how to represent queries and items as box embeddings and how to compute the ranking score with the box representations. Then we introduce how to train the box embeddings in ranking tasks. Specifically, we introduce two the additional constraints in the loss function, which can enhance the model to better distinguish the relevant and irrelevant items and benefit the online retrieval efficiency.

4.1 Formulation

For a set of queries Q and a set of items I, each query $q \in Q$ usually have a relevant item subset $I_q \in I$ ($|I_q|$ is usually much smaller than $|I|$). Given a query $q \in Q$ or an item $i \in I$, we can represent them as d-dimensional axis-aligned hyper-rectangles (or boxes) $box\,(q)$ and $box\,(i)$.

$$box\,(q) = \left\langle \left[q_1^l, q_1^u\right], ..., \left[q_d^l, q_d^u\right] \right\rangle \quad (11)$$

$$box\,(i) = \left\langle \left[i_1^l, i_1^u\right], ..., \left[i_d^l, i_d^u\right] \right\rangle \quad (12)$$

Using the Gumbel distribution-based overlapping volume measure $V_g\,(box\,(q \wedge i))$ given in Eqn. (10) between $box\,(q)$ and $box\,(i)$, we can model the relevance between query q and item i.

4.2 Training for Ranking

4.2.1 ***Ranking Loss.*** For the ranking tasks, given a query $q \in Q$, the goal is optimizing model to rank the positive items i_q^+ higher than the negative items i_q^-, which are produced by the sampling strategy. Let $f\,(q, i)$ be the relevance score between query q and item i predicted by the model, which equals to $V_g\,(box\,(q \wedge i))$ mentioned above. We adopt the pairwise ranking loss as the optimization objective, which is defined as:

$$L_r = \log\left(1 + \exp^{f\left(q, i_q^-\right) - f\left(q, i_q^+\right)}\right) \quad (13)$$

4.2.2 ***Volume Regularization.*** In box embeddings, the volume of boxes can reflect the magnitude of the diversity and uncertainty of the queries and items. We introduce this regularization measure by penalizing the volumes of boxes when they become greater than a fixed value, which can give a reasonable bound for the measure of uncertainty.

$$\begin{aligned} L_v = &\sum_q 1_{\left[V_g(box(q)) > \tau\right]} V_g\left(box\left(q\right)\right) \\ &+ \sum_{i \in \{i_q^+\} \cup \{i_q^-\}} 1_{\left[V_g(box(i)) > \tau\right]} V_g\left(box\left(i\right)\right) \end{aligned} \quad (14)$$

4.3 Overlapping Constraints for More Efficient Indexing and Retrieval

Through box representations we can easily judge whether query box $box\,(q)$ and item box $box\,(i)$ have a non-empty overlapping region. If two boxes are disjoint with each other, (i.e. there exist at least one dimension j such that $\left|q_j \wedge i_j\right| < 0$.), the overlapping volume $V\,(box(q \wedge i))$ must be zero, indicating that the item is irrelevant to the query. Therefore, we can leverage this property to safely filter irrelevant items, which can reduce the visiting times on the whole set of items, and thus, improve the searching efficiency.

If most of the irrelevant boxes are indeed disjoint, then we can get a large improvement in efficiency via this filtering strategy. Therefore, we introduce two constraints to enable the model to better distinguish the positive items i_q^+ and negative items i_q^-, and most importantly, make $box\,(q)$ and $box\left(i_q^-\right)$ more likely to be disjoint with each other. In detail, given the lower and upper boundaries of $box\left(q \wedge i_q^+\right)$ and $box\left(q \wedge i_q^-\right)$,

$$box\left(q \wedge i_q^+\right) = \left\langle q_1 \wedge \left(i_q^+\right)_1, ..., q_d \wedge \left(i_q^+\right)_d \right\rangle \quad (15)$$

$$box\left(q \wedge i_q^-\right) = \left\langle q_1 \wedge \left(i_q^-\right)_1, ..., q_d \wedge \left(i_q^-\right)_d \right\rangle \quad (16)$$

We find the minimum of interval lengths of $box\left(q \wedge i_q^+\right)$ and $box\left(q \wedge i_q^-\right)$.

$$\min box\left(q \wedge i_q^+\right) = \min_k \left|q_k \wedge \left(i_q^+\right)_k\right| \quad (17)$$

$$\min box\left(q \wedge i_q^-\right) = \min_k \left|q_k \wedge \left(i_q^-\right)_k\right| \quad (18)$$

Noted that whether such minimum interval lengths are positive implies whether two boxes overlap with each other. Therefore, we require $\min box\left(q \wedge i_q^+\right) > 0$ and $\min box\left(q \wedge i_q^-\right) < 0$, which can help the model to better distinguish the negative items from the positive ones based on the disjointness of box representations. The two overlapping constraints are formalized as below,

$$\begin{aligned} L_c = &\max\left(0, \delta - \min box\left(q \wedge i_q^+\right)\right) \\ &+ \max\left(0, \delta + \min box\left(q \wedge i_q^-\right)\right) \end{aligned} \quad (19)$$

Where δ is the margin value of the constraints. Combining the ranking loss, volume regularization, and overlapping constraints, the final training loss are given by,

$$L = L_r + \lambda_v L_v + \lambda_c L_c \quad (20)$$

Where λ_v and λ_c correspond to the weights of loss L_v and L_c, respectively.

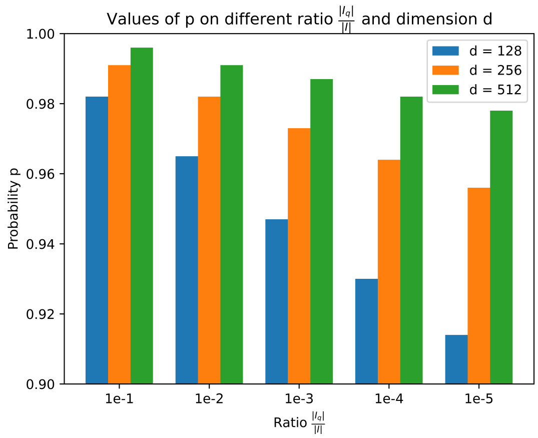

Figure 2: Values of probability p on different $\frac{|I_q|}{|I|}$ and dimension d.

5 INDEXING AND SEARCHING METHODS

In this section, we first analyze how to design an efficient indexing and searching methods based on the learned box embeddings. Then we introduce the details of index building and search algorithm (Algorithm 1).

5.1 Analysis

After the training stages, we can obtain the learned box embeddings of every query $q \in Q$ and item $i \in I$ through the model inference. Theoretically, under the assumption of independent random distribution, query box $box\,(q)$ have the average probability p to overlap with each item box $box\,(i)$ in each dimension (i.e. the average probability $1-p$ to disjoint). Through the joint filtering of all d dimension, the expected size of the remaining relevant item set I_q follows this equation:

$$|I|p^d = |I_q| \Leftrightarrow p = \left(\frac{|I_q|}{|I|}\right)^{\frac{1}{d}} \tag{21}$$

In practice, as shown in Figure 2, the embedding size in learning process usually make p close to 1, which means that in single dimension, the items disjoint with the query are the minority. Inspired by this finding, we can obtain the relevant items by filtering the irrelevant items in each dimension.

5.2 Building Indexing

As mentioned above, the box representations of irrelevant items are usually disjoint with query box $box\,(q)$ on at least one dimension. As shown in Figure 3, for a single dimension j, one of the following situations happens,

- The lower bound q_j^l of query box $box\,(q)$ is larger than the upper bound i_j^u of item box $box\,(i)$.
- The upper bound q_j^u of query box $box\,(q)$ is smaller than the lower bound i_j^l of item box $box\,(i)$.

As shown in the step 1 of Algorithm 1, for item set I and each dimension j, we sort the items in I based on the lower and upper

Figure 3: The situations of disjointedness in single dimension.

boundaries of box representation respectively. The sorted item sets $[I]_j^l$ and $[I]_j^u$ are given as,

$$[I]_j^l = \left\{i_{j,1}^l, ..., i_{j,|I|}^l\right\} \tag{22}$$

$$[I]_j^u = \left\{i_{j,1}^u, ..., i_{j,|I|}^u\right\} \tag{23}$$

The sorted lower and upper bounds are formalized as below,

$$[box\,(I)]_j^l = \left\{\left(i_{j,1}^l\right)_j^l, ..., \left(i_{j,|I|}^l\right)_j^l\right\} \tag{24}$$

$$[box\,(I)]_j^u = \left\{\left(i_{j,1}^u\right)_j^u, ..., \left(i_{j,|I|}^u\right)_j^u\right\} \tag{25}$$

In our model, these sorted sets are used to serve as box embedding-based index.

Time Complexity. By adopting the classical sorting algorithm on d dimension box embeddings of $|I|$ items, the time cost can be inferred as $O(d|I|\log|I|)$.

5.3 Searching Relevant Items

As shown in the step 2-3 of Algorithm 1, given the query box $box\,(q)$, we initialize a bitarray B, which are valued by 1 and have a length of $|I|$. Bitarray B is utilized to mark whether the item box of $i \in I$ overlaps with $box\,(q)$. $B\,[i] = 1$ indicates that item i overlaps with q and otherwise $B\,[i] = 0$.

5.3.1 ***Binary search.*** For each dimension j, the lower bound q_j^l and upper bound q_j^u of the query box are used as the key in the binary search on the sorted upper bounds $[box\,(I)]_j^u$ and lower bound $[box\,(I)]_j^l$ of the item sets I, respectively, which will return two position indexes e^+ and e^- to filter the irrelevant items before e^+ and after e^-. We can formalize the filtered irrelevant item set I_j^{q-} as below,

$$I_j^{q-} = \left\{i_{j,1}^u, ..., i_{j,e^+}^u\right\} \cup \left\{i_{j,e^-}^l, ..., i_{j,|I|}^l\right\} \tag{26}$$

Time Complexity. The general binary search algorithm on d dimension box embeddings of $|I|$ sorted items requires the time cost $O\,(d\log|I|)$.

5.3.2 ***Bit-level Operation.*** After the binary search on all d dimension, the final irrelevant item set are remained as,

$$I^{q-} = \left\{I_j^{q-}\right\}, j = 1, ..., d \tag{27}$$

As shown in the step 4-5 of Algorithm 1, for each irrelevant item in Irr^q, we check the value of corresponding position in the bitarray B, and update the value to 0 if the current value are 1. Then we can visit the whole bitarray B to return the relevant items I_q, which has the value 1.

$$I^{q+} = \left\{i \mid B[i] = 0, \forall i \in N^i < |I|\right\} \tag{28}$$

Time Complexity. The time cost contains three parts:

- Checking operation: The time cost is decided by the expected size of Irr^q, which will spend,

$$O\left(d|I|\left(1-p\right)\right) \Leftrightarrow O\left(d|I|\left(1-\left(\frac{|I_q|}{|I|}\right)^{\frac{1}{d}}\right)\right) \tag{29}$$

- Updating operation: The repeated updating are not needed, and the time cost depends on the size of irrelevant item set, which spend $O\left(|I|-|I_q|\right)$.
- Visiting operation: Visiting the $|I|$-length bitarray B and finding the position with value 1 will cost $O\left(|I|\right)$.

Note that such operations are all **bit operations**, and can be processed very fast in practice.

Algorithm 1 Indexing and searching methods based on box embeddings.

Input: $|I|$-length bitarray B with initial value 1, d-dimensional lower and upper boundaries of all $|I|$ item boxes and a query box of q.
Output: The scores of $|I_q|$ relevant items.
1: Sort the lower and upper boundaries of all $|I|$ item boxes.
2: Search the irrelevant items of query q on each dimension, based on the sorted boundaries of $|I|$ item boxes.
3: Merge the irrelevant items of query q on all d dimension.
4: Check and update the 1-value in bitarray B to 0, according to the index of the merged irrelevant items.
5: Visit bitarray B to find the $|I_q|$ relevant items valued by 1.
6: Compute the scores of the $|I_q|$ relevant items.

5.4 Calculating Scores of Relevant Items

As shown in the step 6 of Algorithm 1, after returning the relevant item set I_q, we can infer the scores $f\left(q,i\right)$ through inputing the query box $box\left(q\right)$ and each item box $box\left(i\right), i \in I_q$ to our model. The final ranking list are produced according to the scores.
Time Complexity. The above calculation on d dimension box embeddings of a query and $|I_q|$ relevant items requires $O\left(d|I_q|\right)$ time .

5.5 Discussion

We also considered using the spatial index methods that support range query (e.g. R-tree [13], R*-tree [2], X-tree [3], and Priority R-tree [1]) for this problem. These methods can find the geometry objects efficiently and achieve a sublinear time complexity. However, existing spatial index usually suffer from the **curse of dimensionality** and the searching efficiency can drop to $O\left(d|I|\right)$ with the increase of dimension ($d > 10$). Such problems limit the application of high dimensional spatial index, so we don't adopt the spatial index as the indexing methods of box embeddings.

6 EXPERIMENTS

We conduct experiments on the two tasks of document ranking, recommendation to demonstrate the improvement in both effectiveness and efficiency.

6.1 Passage Retrieval

6.1.1 ***Datasets and Metrics.*** We adopt the corpus of the passage retrieval task in TREC 2019 Deep Learning (DL) Track [4], which contains 8,841,823 passages, 502,939 training queries, and 6,980 test queries. For effectiveness, we use the widely-used metrics MRR@10 and Recall@100 to evaluate the top-ranking performance. For efficiency, we report the ratios of remained items $\frac{|I_{q^+}|}{|I|}$, and the average latency of the searching process.

6.1.2 ***Baselines.***
Sparse Retrieval & Cascade IR. We report several important results according to the TREC [4], the leaderboard on MSMARCO dataset [10, 27] and cascade systems: BM25 (classic traditional retrieval method) [39], DeepCT (BERT weighted BM25) [5], the best BERT model [29], which use BM25 as the first-stage retriever.

Dense Retrieval. There are several baselines of dense retrieval, which the difference between them mainly lie in the negative sampling strategy of training.

- Random negative sampling: Rand Neg [15] randomly samples negatives from the entire corpus, In-Batch Neg [21] utilize other queries' relevant documents in the same mini-batch as negative documents to extend the training data.
- Static hard negative sampling: BM25 Neg [11] uses the BM25 top-retrieved results as the negative documents, ANCE [38] refreshes the document index and retrieves the static hard negatives under the parameters of current model.
- Hybrid negative sampling: STAR [40] utilizes both random sampling negatives and the static hard negatives to train model, and reuses the document embeddings in the same batch.

6.1.3 ***Implementation Details.*** Following the settings in [40], all dense retrieval models adopt the RoBERTabase [24] model as the encoder of queries and documents. The inner product is used to calculate the relevance score, and we adopt the Faiss [19] to perform the efficient searching. We build the index of the accurate inner product (IndexFlatIP), and process the exact search to find the nearest vectors, which is aligned with the exact search in our methods that find the overlapped boxes. The top-200 documents are used as the hard negatives. More details have been elaborated in [40].

For box retrieval model, we adopt the same training settings as STAR, except utilizing two feed-forward neural networks to map the 768-dimensional dense vectors to 384-dimensional minimum and maximum coordinates of the box embeddings. We compute the relevance score based on box representations. We test the β in Gumbel-Box $\in [0.001, 0.01, 0.1, 1]$, λ_v weight of loss $L_v \in [0.001, 0.01, 0.1, 1]$, and λ_c weight of loss $L_c \in [0.01, 0.1, 1]$, margin value $\lambda = 1$. After obtaining the learned box embeddings, we build the index and search the relevant documents as mentioned above.

We use Numba [1] to parallelize our searching process on different dimensions, and calculate the scores of the relevant documents by Pytorch. In practice, Numba has no bit-level data type, we adopt the Numba's smallest data type np.bool (size = np.int8). As the major

[1]http://numba.pydata.org/

Table 1: Metrics of all baselines on MSMARCO Dev Passage. We use paired t-test with p-value threshold of 0.05 on the test dataset. $*$ (or #) indicates significant difference between baselines and Box+STAR (or Box+STAR+Constraint).

Models	MRR@10	Recall@100	Latency (ms/Per Query)	ratios $\frac{\|I_q\|}{\|I\|}$ (%)
Sparse Retrieval & Cascade IR			-	-
BM25	$0.187^{*\#}$	$0.670^{*\#}$	36	-
Best TREC Trad Retrieval	$0.240^{*\#}$	-	-	-
Best DeepCT	$0.243^{*\#}$	$0.760^{*\#}$	-	-
BERT Reranker	$0.365^{*\#}$	-	-	-
Dense Retrieval			293	
In-Batch Neg	$0.264^{*\#}$	$0.837^{*\#}$	-	-
Rand Neg	$0.301^{*\#}$	$0.853^{*\#}$	-	-
BM25 Neg	$0.309^{*\#}$	$0.813^{*\#}$	-	-
ANCE	0.338	0.862	-	-
STAR	0.340	**0.867**	-	-
Box Retrieval			-	-
Box + STAR	0.3407 (+0.2%)	0.859 (-0.9%)	2783	40.27
Box + STAR + Constraint	**0.3418** (+0.59%)	0.856 (-1.27%)	137	0.93

bottleneck of retrieval lies in the memory access step, we believe that much more improvement on efficiency can be brought if we use the 8 times smaller bit-level data type to process the searching step.

Noticing that in the test step, all the calculation are processed on the CPU.

6.1.4 ***Results and Analysis.*** Experimental results are given in Table 1.
Effectiveness. We can see that under the same negative sampling methods STAR, box embedding-based models achieve better ranking performance than all the vector embedding-based models, indicating that the box embedding models have a good representation ability to capture semantic diversity and uncertainty in document ranking tasks. We also observe a little drop in the Recall@100 measure. This is because that our box embedding model may filter some false negative documents in searching stage and lead to a drop in recall.
Efficiency. We can observe that compared with the dense retrieval models, we achieve the better average retrieval latency, because we can filter a large proportion of irrelevant documents (> 99%) with fast bit-level operations. It demonstrate efficiency of the proposed box embedding-based indexing methods.
Ablation Study. We investigate the effect of the overlapping constraints on filtering the irrelevant documents and improving the ranking performance. From the results we can see that , without the constraints, a substantial proportion of the documents (about 40%) will not be filtered out by the indexing and searching algorithm, which results in a unacceptable retrieval latency (2.783s per query). We also note that the constraints slightly improve the ranking performance (from a MRR@10 of 0.3407 to 0.3418). This is because the constraints can help the model better distinguish the positive and negative documents.

6.2 Recommendation

6.2.1 ***Datasets and Metrics.*** We experiment with two publicly available datasets: MovieLens-1M, and Amazon books. The statistics of the two datasets are summarized in Table 1. For effectiveness, we we report the NDCG@10, and Hit@10 of the top-retrieved results on both the 100 random sampled negative items and the full item set. For efficiency, we report the ratios of remained items $\frac{|I_{q^+}|}{|I|}$, and the average latency of the recommendation process.

Table 2: Statistics of datasets in recommendation task.

Dataset	#User	#Item	#Interaction
MovieLens-1M	6,040	3,706	1,000,209
Amazon Books	351,356	393,801	6,271,511

6.2.2 ***Baselines.*** We compare our box embedding model with the following three representative two-tower models.

- DSSM [16] employs vector embeddings to represent users and items. The relevance score is calculated based on the inner product between the user vector and the item vector.
- GRU4REC [14] applies GRU network to model user interaction sequence for session-based recommendation.
- SASREC [20] is a self-attention based sequential recommendation model, which uses the multi-head attention mechanism to recommend the next item.

6.2.3 ***Implementation Details.*** We implement the baselines and our model using the Recbole [42], a python package that contains many advanced recommended models. We follow the training settings in [42] including the automatic parameter fine-tuning. For preprocessing, we filter the two datasets that retained only users with at least 20 interactions. For training the vector embedding-based model, we encode the features of users and items to 128-dimensional vectors. We adopt the Faiss to perform the efficient searching, and build the index of the accurate inner product (IndexFlatIP). More details have been demonstrated in [42].

For box embedding-based model, we utilize two feed-forward neural networks to map the 128-dimensional vectors to 128-dimensional

Table 3: Metrics of all baselines on MovieLens by testing random 100 negative items and full items. We use paired t-test with p-value threshold of 0.05 on the test dataset. +, * and # indicate significant difference over DSSM, GRU4Rec and SASrec, respectively.

Models	random 100		full			
	NDCG@10	Hit@10	NDCG@10	Hit@10	Latency (ms)	ratios $\frac{\lvert I_q \rvert}{\lvert I \rvert}$ (%)
Vector Framework					0.24	
DSSM	0.2238	0.4142	0.0203	0.0441	-	-
GRU4Rec	0.5147	0.7760	0.0989	0.1972	-	-
SASrec	0.5347	0.7887	0.1018	0.2005	-	-
Box Framework						
Box + DSSM + Constraint	0.3652^{+} (+63.2%)	0.6368^{+} (+53.7%)	0.0374^{+} (+84.2%)	0.0709^{+} (+60.8%)	0.19	5.3
Box + GRU4rec + Constraint	0.5377^{*} (+4.7%)	0.7835^{*} (+1.0%)	0.1072^{*} (+8.4%)	0.2116^{*} (+7.3%)	0.15	4.5
Box + SASRec + Constraint	$\mathbf{0.5438}^{\#}$ (+2.6%)	$\mathbf{0.7960}^{\#}$ (+0.9%)	$\mathbf{0.1094}^{\#}$ (+7.5%)	$\mathbf{0.2139}^{\#}$ (+6.7%)	0.21	5.6

Table 4: Metrics of all baselines on Amazon Books by testing random 100 negative items and full items. We use paired t-test with p-value threshold of 0.05 on the test dataset. +, * and # indicate significant difference over DSSM, GRU4Rec and SASrec, respectively.

Models	random 100		full			
	NDCG@10	Hit@10	NDCG@10	Hit@10	Latency (ms)	ratios $\frac{\lvert I_q \rvert}{\lvert I \rvert}$ (%)
Vector Framework					2.6	
DSSM	0.3650	0.4210	0.0032	0.0064	-	-
GRU4Rec	0.6095	0.8533	0.0114	0.0302	-	-
SASrec	0.6288	0.8707	0.0119	0.0316	-	-
Box Framework						
Box + DSSM + Constraint	0.4044^{+} (+10.8%)	0.5332^{+} (+26.7%)	0.005^{+} (+56.3%)	0.0075^{+} (+17.2%)	1.2	3.4
Box + GRU4Rec + Constraint	0.6159^{*} (+1.0%)	0.8604^{*} (+0.8%)	0.0125^{*} (+9.6%)	0.0325^{*} (+7.6%)	0.9	2.6
Box + SASRec + Constraint	$\mathbf{0.6341}^{\#}$ (+0.8%)	$\mathbf{0.8797}^{\#}$ (+1.0%)	$\mathbf{0.0129}^{\#}$ (+8.4%)	$\mathbf{0.0337}^{\#}$ (+6.6%)	1.4	4.9

box embeddings. We compute the relevance score based on box representations. We test the β in GumbelBox $\in [0.001, 0.01, 0.1, 1]$, λ_v weight of loss $L_v \in [0.001, 0.01, 0.1, 1]$, and λ_c weight of loss $L_c \in [0.01, 0.1, 1]$, margin value $\lambda = 1$.

We adopt the same testing methods as document ranking tasks.

6.2.4 *Results and Analysis.*

Experimental results are given in Table 3 and Table 4.

Effectiveness. We can see that under the same baseline models, box embedding framework can enhance the ranking performance, demonstrating a superior representation ability of box embeddings in recommendation tasks. For simple models like DSSM, our framework gain more improvement. For the sequential models GRU4RREC and SASREC, we also observe a consistent improvement in ranking performance. However, as these two models already achieved a good performance, the relative improvement is not as large as the improvement over DSSM.

Efficiency. We can observe that compared with the three two-tower models, we achieve the better average latency for single query, which benefit from the filtering operation in searching step, which again demonstrate the efficiency of the box embedding models and the proposed indexing and searching algorithm.

7 CONCLUSION

In this paper, we propose to improve effectiveness and efficiency for ranking tasks by incorporating the recently proposed probabilistic box embeddings. For effectiveness, the box embeddings can better represent the semantic diversity and uncertainty. The overlapping constraints are introduced to enhance the model to better distinguish the relevant and irrelevant items. After obtaining the learning box embeddings, a box embedding-based indexing method are designed, which can filter irrelevant items and reduce the visiting times without losing the relevance accuracy in top-retrieved results. We conduct experiments on real-world datasets of ranking tasks. The experimental results show that the proposed method can outperform existing vector embedding-based approaches both in effectiveness and efficiency.

ACKNOWLEDGMENTS

This research was supported by the Natural Science Foundation of China (61902209, U2001212), and Beijing Outstanding Young Scientist Program (NO. BJJWZYJH012019100020098) and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China.". We thank Zhewei Wei and Feng Zhang from the Renmin University of China for the discussion on the efficient indexing method.

REFERENCES

[1] Lars Arge, Mark De Berg, Herman Haverkort, and Ke Yi. 2008. The priority R-tree: A practically efficient and worst-case optimal R-tree. *ACM Transactions on Algorithms (TALG)* 4, 1 (2008), 1–30.

[2] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*. 322–331.

[3] Stefan Berchtold, Daniel A Keim, and Hans-Peter Kriegel. 1996. The X-tree: An index structure for high-dimensional data. In *Very Large Data-Bases*. 28–39.

[4] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *arXiv preprint arXiv:2003.07820* (2020).

[5] Zhuyun Dai and Jamie Callan. 2019. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687* (2019).

[6] Shib Sankar Dasgupta, Michael Boratko, Dongxu Zhang, Luke Vilnis, Xiang Lorraine Li, and Andrew McCallum. 2020. Improving Local Identifiability in Probabilistic Box Embeddings. *arXiv preprint arXiv:2010.04831* (2020).

[7] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*. 253–262.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[9] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*. PMLR, 1646–1655.

[10] Luyu Gao, Zhuyun Dai, and Jamie Callan. 2021. COIL: Revisit Exact Lexical Match in Information Retrieval with Contextualized Inverted List. *arXiv preprint arXiv:2104.07186* (2021).

[11] Luyu Gao, Zhuyun Dai, Tongfei Chen, Zhen Fan, Benjamin Van Durme, and Jamie Callan. 2020. Complementing lexical retrieval with semantic residual embedding. *arXiv preprint arXiv:2004.13969* (2020).

[12] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2946–2953.

[13] Antonin Guttman. 1984. R-trees: A dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*. 47–57.

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939* (2015).

[15] Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2553–2561.

[16] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[17] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2010. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence* 33, 1 (2010), 117–128.

[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734* (2017).

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data* (2019).

[20] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[21] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906* (2020).

[22] Alice Lai and Julia Hockenmaier. 2017. Learning to predict denotational probabilities for modeling entailment. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. 721–730.

[23] Xiang Li, Luke Vilnis, Dongxu Zhang, Michael Boratko, and Andrew McCallum. 2018. Smoothing the geometry of probabilistic box embeddings. In *International Conference on Learning Representations*.

[24] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

[25] Yu A Malkov and Dmitry A Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence* 42, 4 (2018), 824–836.

[26] Lang Mei, Jun He, Hongyan Liu, and Xiaoyong Du. 2019. Latent path connected space model for recommendation. In *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*. Springer, 163–172.

[27] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

[28] Maximillian Nickel and Douwe Kiela. 2017. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems* 30 (2017), 6338–6347.

[29] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[30] Hongyu Ren, Weihua Hu, and Jure Leskovec. 2020. Query2box: Reasoning over knowledge graphs in vector space using box embeddings. *arXiv preprint arXiv:2002.05969* (2020).

[31] Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94*. Springer, 232–241.

[32] Malcolm Slaney and Michael Casey. 2008. Locality-sensitive hashing for finding nearest neighbors [lecture notes]. *IEEE Signal processing magazine* 25, 2 (2008), 128–131.

[33] Sandeep Subramanian and Soumen Chakrabarti. 2018. New embedded representations and evaluation protocols for inferring transitive relations. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1037–1040.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[35] Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2015. Order-embeddings of images and language. *arXiv preprint arXiv:1511.06361* (2015).

[36] Luke Vilnis, Xiang Li, Shikhar Murty, and Andrew McCallum. 2018. Probabilistic embedding of knowledge graphs with box lattice measures. *arXiv preprint arXiv:1805.06627* (2018).

[37] Luke Vilnis and Andrew McCallum. 2014. Word representations via gaussian embedding. *arXiv preprint arXiv:1412.6623* (2014).

[38] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808* (2020).

[39] Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible ranking baselines using Lucene. *Journal of Data and Information Quality (JDIQ)* 10, 4 (2018), 1–20.

[40] Jingtao Zhan, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. 2021. Optimizing Dense Retrieval Model Training with Hard Negatives. *arXiv preprint arXiv:2104.08051* (2021).

[41] Shuai Zhang, Huoyu Liu, Aston Zhang, Yue Hu, Ce Zhang, Yumeng Li, Tanchao Zhu, Shaojian He, and Wenwu Ou. 2021. Learning User Representations with Hypercuboids for Recommender Systems. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 716–724.

[42] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Kaiyuan Li, Yushuo Chen, Yujie Lu, Hui Wang, Changxin Tian, Xingyu Pan, Yingqian Min, Zhichao Feng, Xinyan Fan, Xu Chen, Pengfei Wang, Wendi Ji, Yaliang Li, Xiaoling Wang, and Ji-Rong Wen. 2020. RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms. *arXiv preprint arXiv:2011.01731* (2020).

APPENDIX A

Lemma. The overlapping volume function $V\left(box\left(A \wedge B\right)\right)$ is a kernel function.

Proof. First we begin with the situation that $K = 1$.

$$V\left(box\left(A \wedge B\right)\right) = \max\left(\min\left(A_1^u, B_1^u\right) - \max\left(A_1^l, B_1^l\right), 0\right) \tag{30}$$

Let $\phi\left(x, a, b\right)$ be the mapping functions defined as:

$$\phi\left(x, a, b\right) = \begin{cases} 1, if\ a \le x \le b \\ 0,\ otherwise \end{cases} \tag{31}$$

$V\left(box\left(A \wedge B\right)\right)$ function can associate with the inner product measure, which is a kernel function:

$$V\left(box\left(A \wedge B\right)\right) = \int_{-\infty}^{+\infty} \phi\left(x, A_1^l, A_1^u\right) \cdot \phi\left(x, B_1^l, B_1^u\right) dx \tag{32}$$

Considering that the product of two kernel function is still kernel function:

$$(K_1 \otimes K_2)\left((x_1, x_2), \left(x_1^{'}, x_2^{'}\right)\right) = K_1\left(x_1, x_1^{'}\right) \cdot K_2\left(x_2, x_2^{'}\right) \quad (33)$$

we can infer that for dimension $K > 1$, $V\left(box\left(A \wedge B\right)\right)$ is still kernel function.

$$V\left(box\left(A \wedge B\right)\right) = \prod_{k=1}^{K} V\left(box\left(A_k \wedge B_k\right)\right) \quad (34)$$