Set2Box: Similarity Preserving Representation Learning for Sets

Geon Lee
KAIST AI
geonlee0325@kaist.ac.kr

Chanyoung Park
KAIST ISysE & AI
cy.park@kaist.ac.kr

Kijung Shin
KAIST AI & EE
kijungs@kaist.ac.kr

Abstract—Sets have been used for modeling various types of objects, and measuring similarity between them has been a key building block of a wide range of applications. However, as sets have grown in numbers and sizes, the computational cost and storage required for set similarity computation have become substantial. In this work, we propose Set2Box, which represents sets as boxes to precisely capture overlaps of sets and thus accurately estimate various similarity measures. Additionally, based on the proposed box quantization scheme, we design Set2Box+, which yields more concise but more accurate box representations of sets. Through extensive experiments on 8 real-world datasets, we show that, compared to baseline approaches, Set2Box is (a) Accurate: achieving up to \(40.8\times\) smaller estimation error while requiring 60% fewer bits to encode sets, (b) Concise: yielding up to 96.8\(\times\) more concise representations with similar estimation error, and (c) Versatile: enabling the estimation of four set-similarity measures from a single representation of each set. For reproducibility, the source code and datasets used in the paper are available at https://github.com/geon0325/Set2Box.

I. INTRODUCTION

Sets are ubiquitous, modeling various types of objects in many domains, including texts, purchase records, social circles, and online discussions. Moreover, a number of set similarity measures (e.g., Jaccard Index), most of which are based on the overlaps between sets, have been developed.

As a result of the omnipresence of sets, measuring their similarity has been employed as a fundamental building block of a wide range of applications, including (a) plagiarism detection: a document is modeled as a “bag of words,” and documents whose set representations are highly similar are suspected of plagiarism [1], (b) gene expression mining: the functionality of a set of genes is estimated by comparing the set with other sets with known functionality [2], (c) recommendation: users who purchased similar sets of items are identified for collaborative filtering [3], and more (e.g., graph compression [4] and medical image analysis [5]).

As sets grow in numbers and sizes, computation of set similarity requires substantial computational cost and storage. For example, similarities between tens of thousands of movies, which are represented as sets of up to hundreds of thousands of users who have rated them, were measured for movie recommendation [3]. In order to reduce the space and computation required for set-similarity computation, a number of approaches based on hashing and sketching [6], [7] have been developed. While their simplicity and theoretical guarantees are tempting, significant gains are expected if patterns in a given collection of sets can be learned and exploited.

In this paper, we propose Set2Box, a learning-based approach for compressed representations of sets from which various similarity measures can be estimated accurately in constant time. The key idea of Set2Box is to represent sets as boxes, which share primary characteristics of sets, to capture the overlaps between sets and thus their similarity based on them. In addition, we propose Set2Box+, which yields even more concise but more accurate boxes based on the proposed box quantization scheme. Our contributions are as follows:

- **Accurate & Versatile Algorithm:** We propose Set2Box, a set representation learning method that accurately preserves similarity between sets in terms of four measures.
- **Concise & Effective Algorithm:** Based on an end-to-end box quantization scheme, we devise Set2Box$^+$ which yields more accurate and concise similarity estimation.
- **Extensive Experiments:** Using 8 real-world datasets, we validate the advantages of Set2Box$^+$ over its competitors and the effectiveness of each of its components.

In Section II, we review related work. In Section III, we provide preliminaries. In Section IV, we present Set2Box and Set2Box$^+$. In Section V, we provide experimental results. Lastly, we offer conclusions in Section VI.

II. RELATED WORK

**Similarity-Preserving Embedding:** Representation learning for preserving similarities between instances has been studied for graphs [8], [9], images [10], and texts [11]. These methods aim to yield high-quality embeddings by minimizing the information loss of the original data. However, most of them are designed to preserve the predetermined similarity matrix, which are not extensible to new measures [9].

**Box Embedding:** Thanks to the powerful expressiveness of boxes [12], they have been used in diverse applications including knowledge bases [13]–[15], word embedding [16], image embedding [17], and recommender systems [18], [19]. In an algorithmic aspect, methods for improving the optimization of learning boxes (e.g., Gaussian convolutions [20] and Gumbel random variables [21]) have been presented.

**Differentiable Product Quantization:** Product quantization [22], [23] is an effective strategy for vector compression, and recently, deep learning methods for learning discrete codes in an end-to-end manner have been proposed [24], [25].
III. Preliminaries

In this section, we introduce notations and define the problem. Then, we review some intuitive methods for the problem.

**Notations:** Consider a set \( S = \{s_1, \ldots, s_{|S|}\} \) of sets and a set \( E = \{e_1, \ldots, e_{|E|}\} \) of entities. Each set \( s \in S \) is a non-empty subset of \( E \) and its size (i.e., cardinality) is denoted by \(|s|\). A representation of the set \( s \) is denoted by \( z_s \) and its encoding cost (the number of bits to encode \( z_s \)) in \( \text{bits} \) is denoted by \( \text{Cost}(z_s) \). Refer to Table I for frequently-used notations.

**Problem Definition:** The problem of learning similarity-preserving set representations is formulated as:

**Problem 1** (Similarity-Preserving Set Embedding).
1) Given: (1) a set \( S \) of sets and (2) a budget \( b \)
2) Find: a latent representation \( z_s \) of each set \( s \in S \)
3) to Minimize: the difference between (1) the similarity between \( s \) and \( s' \), and (2) the similarity between \( z_s \) and \( z_{s'} \) for all \( s \neq s' \in S \)
4) Subject to: the total encoding cost \( \text{Cost}\{z_s : s \in S\} \leq b \).

In this paper, we consider four set-similarity measures and use the mean squared error (MSE)\(^1\) to measure the differences, while our proposed methods are not specialized to the choices.

**Desirable Properties:** We expect set embeddings for Problem 1 to be: (a) **accurate:** similarities approximated using learned representations are close to those between sets, (b) **concise:** less amount of memory is desiredly used to store embeddings while keeping them informative, (c) **generalizable:** embeddings of unseen sets can be obtained, (d) **versatile:** representations can be used to approximate diverse similarity measures, and (e) **fast:** set similarities are rapidly estimated.

**Intuitive Methods:** We discuss simple and intuitive set-embedding methods for similarity preservation.

- **Random Hashing [6]:** Each set \( s \) is encoded as a binary vector \( z_s \) by mapping each entity into one of the \( d \) different values using a hash function \( h(\cdot) : E \rightarrow \{1, \ldots, d\} \). Specifically, the representation \( z_s \in \{0, 1\}^d \) is derived by \( z_{s[i]} = 1 \) if \( \exists e \in s \) s.t. \( h(e) = i \) and 0 otherwise. The size of the set \( s \) is estimated from the L1 norm of \( z_s \), i.e., \( |s| \approx \|z_s\|_1 \).

- **Vector Embedding:** Another popular approach is to represent sets as vectors and compute the inner products between them to estimate a predefined set similarity. More precisely, given two sets \( s \) and \( s' \) and their vector representations \( z_s \) and \( z_{s'} \), it aims to approximate predefined \( \text{sim}(s, s') \) by the inner product of \( z_s \) and \( z_{s'} \), i.e., \( \langle z_s, z_{s'} \rangle \approx \text{sim}(s, s') \).

However, random hashing cannot accurately represent sets whose sizes are larger than \( d \), and vector embedding shows weakness in its versatility, i.e., it can only preserve a predefined similarity measure. The proposed end-to-end methods, \text{SET2BOX} and \text{SET2BOX}*, effectively address these issues.

\(^1\sum_{s \neq s' \in S} (\text{sim}(s, s') - \text{sim}(z_s, z_{s'}))^2 \text{sim}(\cdot, \cdot) \) and \( \text{sim}(\cdot, \cdot) \) are similarity between sets and that between latent representations, respectively.

### IV. Proposed Method

In this section, we first present \text{SET2BOX}, a novel algorithm for learning similarity-preserving set representations using boxes. Then we propose \text{SET2BOX}*, an advanced version of \text{SET2BOX}, which derives better conciseness and accuracy.

#### A. \text{SET2BOX}: Preliminary Version

We first present \text{SET2BOX}, a preliminary set representation method that effectively learns the set itself and the structural relations with other sets for similarity preserving.

**Concepts:** A box is a \( d \)-dimensional hyper-rectangle whose representation consists of its center and offset [12]. The center describes the location of the box in the latent space and the offset is the length of each edge of the box. Formally, given a box \( B = (c, f) \) whose center \( c \in \mathbb{R}^d \) and offset \( f \in \mathbb{R}^d \) are in the same latent space, the box is defined as a bounded region:

\[
B = \{ p \in \mathbb{R}^d : c - f \leq p \leq c + f \}.
\]

Let \( m \in \mathbb{R}^d \) and \( M \in \mathbb{R}^d \) be the vectors of the minimum and the maximum at each dimension, respectively, i.e., \( m = c - f \) and \( M = c + f \). The intersection of two boxes \( B_X = (c_X, f_X) \) and \( B_Y = (c_Y, f_Y) \) is also a box, represented as:

\[
B_X \cap B_Y = \{ p \in \mathbb{R}^d : \max(m_X, m_Y) \leq p \leq \min(M_X, M_Y) \}.
\]

The volume \( \text{V}(B) \) of the box \( B \) is computed by the product of the length of an edge in each dimension, i.e., \( \text{V}(B) = \prod_{i=1}^d \text{V}(B[i]) \). The volume of the union of the two boxes is simply computed by \( \text{V}(B_X) + \text{V}(B_Y) - \text{V}(B_X \cup B_Y) \).

**Representation:** The core idea of \text{SET2BOX} is to model each set \( s \) as a box \( B_s = (c_s, f_s) \) so that the relations with other sets are properly preserved in the latent space. To this end, \text{SET2BOX} approximates the volumes of the boxes to the relative sizes of the sets, i.e., \( \text{V}(B_s) \propto |s| \). In addition, it aims to preserve the relations between different sets by approximating the volumes of the intersection of the boxes to the intersection sizes of the sets, i.e., \( \text{V}(B_{s_i} \cap B_{s_j}) \propto |s_i \cap s_j| \).

**Objective:** Recall that our goal is to derive **accurate** and **versatile** representations of sets, and towards the first goal, we take relations beyond pairwise into consideration. To this end, we design an objective function that aims to preserve elemental relations among triple of sets. Specifically, given a triple \( \{s_i, s_j, s_k\} \) of sets, we consider seven cardinalities from different levels of subsets: (1) \(|s_i|, |s_j|, |s_k|\), (2) \(|s_i \cap s_j|, |s_j \cap s_k|, |s_k \cap s_i|\), and (3) \(|s_i \cap s_j \cap s_k|\) which contain single,

### TABLE I: Frequently-used symbols.

| Notation | Definition |
|----------|------------|
| \( S = \{s_1, \ldots, s_{|S|}\} \) | set of sets |
| \( E = \{e_1, \ldots, e_{|E|}\} \) | set of entities |
| \( B = (c, f) \) \( \forall (B) \) | a box with center \( c \) and offset \( f \) volume of box \( B \) |
| \( T^+ \) and \( T^- \) | a set of positive & negative samples |
| \( Q_e \in \mathbb{R}^{|E| \times d} \) | center embedding matrix of entities |
| \( Q_f \in \mathbb{R}^{|E| \times d} \) | offset embedding matrix of entities |
| \( D \) | number of subspaces |
| \( K \) | number of key boxes in each subspace |
pair, and triple-wise information, respectively, and we denote them from \( c_{ij}(s_i, s_j, s_k) \) to \( c_{ijk}(s_i, s_j, s_k) \). These elements fully describe the relations among the three sets, and any similarity measures are computable using them. In this regard, we aim to preserve the ratios of the seven cardinalities by the volumes of the boxes \( B_{s_i}, B_{s_j}, \) and \( B_{s_k} \) by minimizing the objective:

\[
J(s_i, s_j, s_k, B_{s_i}, B_{s_j}, B_{s_k}) = \sum_{t=1}^{3} \left( p_t(s_i, s_j, s_k) - \hat{p}_t(B_{s_i}, B_{s_j}, B_{s_k}) \right)^2,
\]

where \( p_t \) is the ratio of the \( t \)-th cardinality among the three sets (i.e., \( p_t = c_t / \sum_{t} c_t \)) and \( \hat{p}_t \) is the corresponding ratio estimated by the boxes. We sample a set \( T^+ \) of positive triples (three overlapping sets) and a set \( T^- \) of negative triples (three uniform random sets) and, using \( T = T^+ \cup T^- \), minimize

\[
\mathcal{L} = \sum_{\{s_i, s_j, s_k\} \in T} J(s_i, s_j, s_k, B_{s_i}, B_{s_j}, B_{s_k}). \tag{1}
\]

Notably, the proposed objective function aims to capture not only the pairwise interactions between sets, but also the triple-wise relations to capture high-order overlapping patterns of the sets. In addition, it does not rely on any predefined similarity measure. It is a general objective for learning structural patterns of sets and their neighbors, and thus it enables the model to yield accurate estimates to diverse metrics.

**Box Embedding:** Here, how can we derive the box \( B_s \) of a set \( s \), i.e., its center \( c_s \) and offset \( f_s \)? To make the method generalizable to unseen sets, we introduce a pair of learnable matrices \( Q^c \in \mathbb{R}^{d \times d} \) and \( Q^f \in \mathbb{R}^{d \times d} \) of entities, where \( Q^c_i \in \mathbb{R}^d \) and \( Q^f_i \in \mathbb{R}^d \) represent the center and offset of an entity \( e_i \), respectively.

Then, the embeddings of the entities in the set \( s \) are aggregated to obtain \( c_s \) and \( f_s \). Here, we use attentions to highlight the entities that are important to obtain the center or the offset of the box. To this end, we define a pooling function that takes the context of each set into account, termed set-context pooling (SCP). Specifically, given a set \( s \) and an item embedding matrix \( Q \) (which can be either \( Q^c \) or \( Q^f \)), it first obtains the set-specific context vector \( b_s \):

\[
b_s = \sum_{e_i \in s} \alpha_i Q_i \quad \text{where} \quad \alpha_i = \frac{\exp(a^T Q_i)}{\sum_{e_j \in s} \exp(a^T Q_j)},
\]

where \( a \) is a global context vector shared by all sets. Then using the context vector \( b_s \), which specifically contains the information on set \( s \), it obtains the output embedding from:

\[
SCP(s, Q) = \sum_{e_i \in s} \omega_i Q_i \quad \text{where} \quad \omega_i = \frac{\exp(b_s^T Q_i)}{\sum_{e_j \in s} \exp(b_s^T Q_j)}.
\]

To be precise, \( c_s = SCP(s, Q^c) \) and \( f_s = |s|^{-\frac{1}{2}} SCP(s, Q^f) \). For the offset \( f_s \), we multiply an additional regularizer \( |s|^{-\frac{1}{2}} SCP(s, Q^f) \) without which a natural condition for boxes (spec., \( \min_{e_i \in s} Q^f_i \leq f_s \leq \max_{e_i \in s} Q^f_i \)) does not hold (see [26] for details).

**Smoothing Boxes:** By definition, a box \( B = (c, f) \) is a bounded region with hard edges whose volume is \( V(B) = \prod_{i=1}^{d} \text{ReLU}(M[i] - m[i]) \) where \( m = c - f \) and \( M = c + f \).

This, however, disables gradient-based optimization when boxes are disjoint [20], and thus we smooth the boxes by \( V(B) = \prod_{i=1}^{d} \text{Softplus}(M[i] - m[i]) \) where \( \text{Softplus}(x) = \frac{1}{\beta} \log(1 + \exp(\beta x)) \) is an approximation to \( \text{ReLU}(x) \), and it becomes closer to \( \text{ReLU} \) as \( \beta \) increases. In this way, any pairs of boxes overlap each other, and thus non-zero gradients are computed for optimization.

**Encoding Cost:** Each box consists of two vectors, a center and an offset, and it requires \( 2 \cdot 32d = 64d \) bits to encode them. \(^2\) Thus, \( 64|S|d \) bits are required for \( |S| \) sets.

**B. SET2Box\(^+\): Advanced Version**

We describe SET2Box\(^+\), which enhances SET2Box in terms of conciseness and accuracy, based on an end-to-end box quantization scheme. Specifically, SET2Box\(^+\) compresses the the box embeddings into a compact set of key boxes and a set of discrete codes to reconstruct the original boxes.

**Box Quantization:** We propose box quantization, a novel scheme for compressing boxes by using substantially smaller number of bits. Note that conventional product quantization methods [24], which are for vector compression, are straightforwardly applicable, by independently reducing the center and the offset of the box. However, it hardly makes use of geometric properties of boxes, and thus it does not properly reflect the complex relations between them. The proposed box quantization scheme effectively addresses this issue through two steps: (1) box discretization and (2) box reconstruction.

- **Box Discretization.** Given a box \( B_s = (c_s, f_s) \) of set \( s \), we discretize the box as a \( K \)-way \( D \)-dimensional discrete code \( C_s = \{1, \ldots, K\}^D \) which is more compact and requires much less number of bits to encode than real numbers. To this end, we divide the \( d \)-dimensional latent space into \( D \) subspaces (\( \mathbb{R}^{d/D} \)) and, for each subspace, learn \( K \) key boxes. Specifically, in the \( i \)-th subspace, the \( j \)-th key box is denoted by \( K^i_j = (c^i_j, f^i_j) \) where \( c^i_j \in \mathbb{R}^{d/D} \) and \( f^i_j \in \mathbb{R}^{d/D} \) are the center and offset of the key box, respectively. The original box \( B_s \) is also partitioned into \( D \) sub-boxes \( B^{(1)}_s, \ldots, B^{(D)}_s \) and the \( i \)-th code of \( C_s \) is decided by:

\[
C_s[i] = \arg \max_j \simmax \left( B^{(i)}_s, K^i_j \right)
\]

where \( \simmax(\cdot, \cdot) \) measures the similarity between two boxes, and we can flexibly select the criterion. In this paper, we specify the \( \simmax \) function, using softmax, as:

\[
C_s[i] = \arg \max_j \frac{\exp(\text{BOR}(B^{(i)}_s, K^i_j))}{\sum_{j'} \exp(\text{BOR}(B^{(i)}_s, K^{i'}_{j'}))}
\]

where BOR (Box Overlap Ratio) is defined to measure how much a box \( B_X \) and a box \( B_Y \) overlap:

\[
\text{BOR}(B_X, B_Y) = \frac{1}{2} \left( \frac{V(B_X \cap B_Y)}{V(B_X)} + \frac{V(B_X \cap B_Y)}{V(B_Y)} \right).
\]

As shown in Figure 1, the proposed box quantization scheme incorporates the geometric relations between boxes, differently \(^2\)We assume that we are using float-32 to represent each real number.
Once we obtain set representations, the key box, i.e., the probability of \( \tau \) indicates the probability for \( i \)th element of set \( C_s \), Eq. (2) is non-differentiable, and to this end, we utilize the softmax with the temperature \( \tau \):

\[
\tilde{C}_s^{(i)}[j] = \frac{\exp \left( \text{BOR} \left( B_s^{(i)}, K_s^{(i)} \right) / \tau \right)}{\sum_j \exp \left( \text{BOR} \left( B_s^{(i)}, K_s^{(i)} \right) / \tau \right)}.
\]  

\( \tilde{C}_s^{(i)} \) is a \( K \)-dimensional probabilistic vector whose \( j \)th element indicates the probability for \( K_s^{(i)} \) being assigned as the closest key box, i.e., the probability of \( C_s[i] = j \). Then, the key box \( \tilde{K}_s^{(i)} = (\tilde{C}_s^{(i)} \mid \tilde{K}_s^{(i)}) \) in the \( i \)th subspace is the weighted sum of the \( K \) key boxes, i.e., \( \tilde{K}_s^{(i)} = \sum_{j=1}^{K} \tilde{C}_s^{(i)}[j] \cdot K_s^{(i)} \). If \( \tau = 0 \), Eq. (3) is equivalent to the arg max function, i.e., a one-hot vector where \( C_s[i]^{(i)} \) dimension is 1 and others are 0. In this case, \( \tilde{K}_s^{(i)} \) becomes equivalent to the discrete \( C_s \), which is the exact reconstruction derivable from the discrete code \( C_s \). However, since this hard selection is non-differentiable and thus prevents an end-to-end optimization, we resort to the approximation by using the softmax with \( \tau \neq 0 \) which is fully differentiable. Specifically, we use different \( \tau \)s in forward (\( \tau = 0 \)) and backward (\( \tau = 1 \)) passes, which effectively enables differentiable optimization.

Joint Training: For further improvement, we introduce a joint learning scheme in the box quantization scheme. Given a triple \( \{s_1, s_2, s_3\} \) of sets from the training data \( T \), we obtain their boxes \( B_{s_1}, B_{s_2}, \) and \( B_{s_3} \) and their reconstructed ones \( \tilde{B}_{s_1}, \tilde{B}_{s_2}, \) and \( \tilde{B}_{s_3} \) using the box quantization. While the basic version of \( \text{SET}2\text{BOX}^+ \) optimizes \( J(s_1, s_2, s_3; B_{s_1}, B_{s_2}, B_{s_3}) \), we jointly train the original boxes together with the reconstructed ones so that both types of boxes can achieve high accuracy. To this end, we consider the following eight losses:

\[
\begin{align*}
J(s_1, s_2, s_3; B_{s_1}, B_{s_2}, B_{s_3}), & \quad J(s_1, s_2, s_3; \tilde{B}_{s_1}, B_{s_2}, B_{s_3}), \\
J(s_1, s_2, s_3; B_{s_1}, \tilde{B}_{s_2}, B_{s_3}), & \quad J(s_1, s_2, s_3; B_{s_1}, B_{s_2}, \tilde{B}_{s_3}), \\
J(s_1, s_2, s_3; \tilde{B}_{s_1}, \tilde{B}_{s_2}, B_{s_3}), & \quad J(s_1, s_2, s_3; \tilde{B}_{s_1}, B_{s_2}, \tilde{B}_{s_3}), \\
J(s_1, s_2, s_3; B_{s_1}, \tilde{B}_{s_2}, \tilde{B}_{s_3}), & \quad J(s_1, s_2, s_3; \tilde{B}_{s_1}, \tilde{B}_{s_2}, \tilde{B}_{s_3}), 
\end{align*}
\]

where we denote them by \( J_1 \) to \( J_8 \), for the sake of brevity. Notably, \( J_1 \), which utilizes only the original boxes, is an objective used for \( \text{SET}2\text{BOX} \), and \( J_8 \) considers only the reconstructed boxes. Based on these joint views from different types of boxes, the final loss function we aim to minimize is:

\[
\mathcal{L} = \sum_{(s_1, s_2, s_3) \in T} \lambda (J_1 + J_2 + J_3 + J_4 + J_5 + J_6 + J_7 + J_8),
\]

where \( \lambda \) is the coefficient for balancing the losses between the joint views and the loss from the reconstructed boxes. In this way, both original and reconstructed ones are trained together in the latent space. Note that even though both types of boxes are jointly trained to achieve high accuracy, only the reconstructed boxes are used for inference.

Encoding Cost: To encode the box for each set, \( \text{SET}2\text{BOX}^+ \) requires (1) key boxes and (2) discrete codes to encode each set, which requires \( 64Kd \) bits and \( D \log_2 K \) bits, respectively. Thus, it requires \( 64Kd + |S|D \log_2 K \) bits to encode \( |S| \) sets. Notably, if \( K \ll |S| \), then \( 64Kd \) bits are negligible, and typically, \( D \log_2 K \ll 64d \) holds. Thus, the encoding cost of \( \text{SET}2\text{BOX}^+ \) is considerably smaller than that of \( \text{SET}2\text{BOX} \).

Similarity Computation: Once we obtain set representations, it is desirable to rapidly compute the estimated similarities in the latent space. Boxes, which \( \text{SET}2\text{BOX} \) and \( \text{SET}2\text{BOX}^+ \) derive, require constant time \( O(d) \) to compute a pairwise similarity between two sets (formalized in [26]).

V. EXPERIMENTAL RESULTS

We review our experiments designed for answering Q1-Q3.

Q1. Accuracy & Conciseness: Does \( \text{SET}2\text{BOX}^+ \) derive concise and accurate set representations than its competitors?

Q2. Effectiveness: How does \( \text{SET}2\text{BOX}^+ \) yield concise and accurate representations? Are all its components useful?

Q3. Effects of Parameters: How do the parameters of \( \text{SET}2\text{BOX}^+ \) affect the quality of set representations?

A. Experimental Settings

Below, we briefly describe settings (see [26] for details).

Machines & Implementations: All experiments were conducted on a Linux server with RTX 3090Ti GPUs.

Datasets: Yelp (YP), Amazon (AM), Netflix (NF), and Movielens (ML) are review datasets where each set is a group of items that a user rated. Gplus (GP) and Twitter (TW) are social networks where each set is a group of neighbors of each node.

Baselines: We compare \( \text{SET}2\text{BOX} \) and \( \text{SET}2\text{BOX}^+ \) with:

- **SET2BIN** encodes each set \( s \) as a binary vector \( z_s \in \{0, 1\}^d \) using a random hash function. See Section III for details.

- **Gplus** requires \( 64Kd \) bits and \( D \log_2 K \) bits, respectively. Thus, it requires \( 64Kd + |S|D \log_2 K \) bits to encode \( |S| \) sets. Notably, if \( K \ll |S| \), then \( 64Kd \) bits are negligible, and typically, \( D \log_2 K \ll 64d \) holds. Thus, the encoding cost of \( \text{SET}2\text{BOX}^+ \) is considerably smaller than that of \( \text{SET}2\text{BOX} \).

- **Similarity Computation:** Once we obtain set representations, it is desirable to rapidly compute the estimated similarities in the latent space. Boxes, which \( \text{SET}2\text{BOX} \) and \( \text{SET}2\text{BOX}^+ \) derive, require constant time \( O(d) \) to compute a pairwise similarity between two sets (formalized in [26]).

V. EXPERIMENTAL RESULTS

We review our experiments designed for answering Q1-Q3.

Q1. Accuracy & Conciseness: Does \( \text{SET}2\text{BOX}^+ \) derive concise and accurate set representations than its competitors?

Q2. Effectiveness: How does \( \text{SET}2\text{BOX}^+ \) yield concise and accurate representations? Are all its components useful?

Q3. Effects of Parameters: How do the parameters of \( \text{SET}2\text{BOX}^+ \) affect the quality of set representations?

A. Experimental Settings

Below, we briefly describe settings (see [26] for details).

- **Machines & Implementations:** All experiments were conducted on a Linux server with RTX 3090Ti GPUs.

- **Datasets:** Yelp (YP), Amazon (AM), Netflix (NF), and Movielens (ML) are review datasets where each set is a group of items that a user rated. Gplus (GP) and Twitter (TW) are social networks where each set is a group of neighbors of each node.

- **Baselines:** We compare \( \text{SET}2\text{BOX} \) and \( \text{SET}2\text{BOX}^+ \) with:
  - **SET2BIN** encodes each set \( s \) as a binary vector \( z_s \in \{0, 1\}^d \) using a random hash function. See Section III for details.
Cosine Similarity
To verify the conciseness of $S$, we examine the effectiveness of $S$.

Recall that vector-based methods, $S_{2V}$, are not versatile, and thus they need to be trained specifically for each metric, while the proposed methods $S_{2B}$ and $S_{2Box}$ do not separate training. The encoding cost of each method is analyzed in [26].

Evaluation: We used 20% of the sets for training (5% in Netflix), and the remaining sets are split into two halves for validation and test. We sampled 100,000 pairs uniformly at random for and measured the Mean Squared Error (MSE) to evaluate the accuracy of the set similarity approximation. We consider four representative set-similarity measures for evaluation: Overlap Coefficient (OC), Cosine Similarity (CS), Jaccard Index (JI), and Dice Index (DI).

B. Q1. Accuracy & Conciseness
We compare the MSE of the set similarity estimation derived by $S_{2Box^+}$ and its competitors. We set dimensions to 256 for $S_{2Bin}$, 8 for vector based methods ($S_{2Vec}$ and $S_{2Vec^+}$), and 4 for $S_{2Box}$, so that they use the same number of bits to encode sets. For $S_{2Box^+}$, we set $(d, D, K) = (32, 16, 30)$, which results in only $31\% - 40\%$ of the encoding cost of the other methods, unless otherwise stated.

Accuracy: As seen in Figure 2, $S_{2Box^+}$ yields the most accurate set representations while using a smaller number of bits to encode them. For example, in the Twitter, $S_{2Box^+}$ gives $40.8\times$ smaller MSE for the Jaccard Index compared to $S_{2Bin}$. In both cases, $S_{2Box^+}$ uses about $31\%$ of the encoding costs used by the competitors.

Conciseness: To verify the conciseness of $S_{2Box^+}$, we measure the accuracy of competitors across various encoding costs. As seen in Figure 3, $S_{2Box^+}$ yields compact representations of sets while keeping them informative. Vector-based methods are prone to the curse of dimensionality and hardly benefit from high dimensionality. While the MSE of $S_{2Bin}$ decreases with respect to its dimension, it still requires larger space to achieve the MSE of $S_{2Box^+}$. For example, $S_{2Bin}$ requires $34.9\times$ more bits to achieve the same accuracy of $S_{2Box^+}$ in Yelp. This is more noticeable in larger datasets, where $S_{2Bin}$ requires up to $96,8\times$ of the encoding cost of $S_{2Box^+}$, as shown in Figure 3c.

C. Q2. Effectiveness
To verify the effectiveness of each component of $S_{2Box^+}$, we conduct ablation studies by comparing it with its variants. We first consider the following variants:

- **$S_{2Box-PQ}$**: For a box $B = (c, f)$, we apply an end-to-end differentiable product quantization (PQ) [24] to the center $c$ and the offset $f$ independently. It yields two independent discrete codes for the center and the offset, and thus its encoding cost is about twice that of $S_{2Box^+}$.

- **$S_{2Box-BQ}$**: $S_{2Box^+}$ with $\lambda = 0$, where the proposed box quantization is applied but joint training is not. We set $(d, D, K) = (32, 8, 30)$ for $S_{2Box-PQ}$ and $(32, 16, 30)$ for $S_{2Box-BQ}$ and $S_{2Box^+}$ so they all require the same amount of storage.

Effects of Box Quantization: We examine the effectiveness of the proposed box quantization scheme by comparing $S_{2Box-BQ}$ with $S_{2Box-PQ}$. As shown in Table II, on average, $S_{2Box-BQ}$ yields up to $26\%$ smaller MSE than $S_{2Box-PQ}$ while using about the same number of bits. While $S_{2Box-PQ}$ discretizes the center and the offset of the boxes independently, without considering their geometric
TABLE II: The proposed schemes: box quantization and joint training in SET2BOX+ incrementally improves the accuracy (in terms of MSE) averaged over all considered datasets.

| Method        | OC  | CS  | JI  | DI  |
|---------------|-----|-----|-----|-----|
| SET2BOX-PQ    | 0.0129 | 0.0028 | 0.0012 | 0.0023 |
| SET2BOX-BQ    | 0.0106 (-17%) | 0.0023 (-17%) | 0.0009 (-26%) | 0.0019 (-17%) |
| SET2BOX+      | 0.0077 (-40%) | 0.0016 (-44%) | 0.0007 (-41%) | 0.0013 (-42%) |

Properties, the proposed box quantization scheme effectively takes the geometric relations between boxes into account and thus yields high-quality compression.

**Effects of Joint Training:** We analyze the effects of the joint training scheme of SET2BOX+ by comparing SET2BOX-BQ ($\lambda = 0$) and SET2BOX+ ($\lambda \geq 0$). As summarized in Table II, joint training reduces the average MSEs on the considered datasets, by up to 44%, together with the box quantization scheme. These results imply that learning quantized boxes simultaneously with the original boxes improves the quality of the quantization and thus its effectiveness. We also observe that joint training helps not only stabilize but also facilitate the training optimization [26].

**Effects of Boxes:** To confirm the effectiveness of using boxes for representing sets, we consider SET2BOX-ORDER, which is also a region-based geometric embedding method:

- **SET2BOX-ORDER** [27]: A set $s$ is represented as a $d$-dimensional vector $z_s \in \mathbb{R}_+^d$ whose volume is computed as $V(z_s) = \exp\left(-\sum_i z_{s,i}\right)$. It is equivalent to restrict boxes to be located in positive latent space.

We set the dimensions for SET2BOX and SET2BOX to 8 and 4, respectively, so that their encoding costs are the same. In Table III, we compare SET2BOX with SET2BOX-ORDER in terms of the average MSE on the considered datasets for each measure. SET2BOX yields more accurate representations than SET2BOX-ORDER, implying the effectiveness of boxes to represent sets for similarity preservation.

**D. Q3. Effects of Parameters**

We analyze how parameters in SET2BOX+ affect the embedding quality of the set representations. In summary, the estimation error decreases as the number of subspaces ($D$) and the number of key boxes in each subspace ($K$) increase, and especially it is affected more heavily by $D$ than by $K$. Detailed experimental results are available in [26].

**VI. CONCLUSIONS**

In this work, we propose SET2BOX and SET2BOX+, space-efficient similarity-preserving embedding methods for sets. Compared to the competitors, by representing sets as boxes with novel quantization and training schemes, SET2BOX+ is (a) **Accurate**: with 40.8× smaller estimation error (when encoding costs are similar or smaller) (b) **Concise**: with 96.8× smaller encoding costs (when accuracies are similar), and (c) **Versatile**: accurate in terms of various similarity measures.

**Acknowledgements:** This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00157, Robust, Fair, Extensible Data-Centric Continual Learning) (No. 2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)).

**References**

[1] L. Moussiades and A. Vakali, “Pdetec: A clustering approach for detecting plagiarism in source code datasets,” The Computer Journal, vol. 48, no. 6, pp. 651–661, 2005.
[2] N. A. Yousri and D. M. Elkaftashi, “Associating gene functional groups with multiple clinical conditions using jaccard similarity,” in BIBM, 2011.
[3] Y. Koren, “Factorization meets the neighborhood: a multifaceted collabora-
tive filtering model,” in KDD, 2008.
[4] K. Shin, A. Ghoting, M. Kim, and H. Raghavan, “Sweg: Lossless and lossy summarization of web-scale graphs,” in WWW, 2019.
[5] L. Chauvin, K. Kumar, C. Desrosiers, W. Wells III, and M. Toews, “Efficient pairwise neuroimage analysis using the soft jaccard index and 3d keypoint sets,” arXiv:2103.06966, 2021.
[6] R. Guerraoui, A.-M. Kermarrec, O. Ruas, and F. Tâiani, “Smallier, faster & lighter knn graph constructions,” in WWW, 2020.
[7] B. Cui, H. T. Shen, J. Shen, and K. L. Tan, “Exploring bit-difference for approximate knn search in high-dimensional databases,” in AcmDM, 2005.
[8] Y. Xie, M. Gong, S. Wang, W. Liu, and B. Yu, “Sim2vec: Node similarity preserving network embedding,” Information Sciences, vol. 495, pp. 37–51, 2019.
[9] A. Tsitsulin, D. Mottin, P. Karras, and E. Müller, “Verse: Versatile graph embeddings from similarity measures,” in WWW, 2018.
[10] Q. Li, Z. Sun, R. He, and T. Tan, “Deep supervised discrete hashing,” in NeurIPS, 2017.
[11] Y. Li, Y. Sun, and N. Zhu, “Bertocnn: Similarity-preserving enhanced knowledge distillation for stance detection,” Plos one, vol. 16, no. 9, p. e0257130, 2021.
[12] L. Vilnis, X. Li, S. Murty, and A. McCallum, “Probabilistic embedding of knowledge graphs with box lattice measures,” in ACL, 2018.
[13] R. Abboud, I. Ceylan, T. Lukasiewicz, and T. Salvadori, “Boxe: A box embedding model for knowledge base completion,” in NeurIPS, 2020.
[14] X. Chen, M. Boratko, M. Chen, S. S. Dasgupta, X. L. Li, and A. McCallum, “Probabilistic box embeddings for uncertain knowledge graph reasoning,” in ACL, 2021.
[15] H. Ren, W. Hu, and J. Leskovec, “Query2box: Reasoning over knowledge graphs in vector space using box embeddings;” in ICLR, 2019.
[16] S. Dasgupta, M. Boratko, S. Atmakuri, X. Li, L. Di, D. Patel, and A. McCallum, “Wordbox2: Learning word representation using box embeddings;” arXiv:2106.14361, 2021.
[17] A. Rau, G. Garcia-Hernando, D. Stoyanov, G. J. Brotow, and D. Turmukhambetov, “Predicting visual overlap of images through interpretable non-metric box embeddings;” in ECCV, 2020.
[18] S. Zhang, H. Liu, A. Zhang, Y. Hu, C. Zhang, Y. Li, T. Zhu, S. He, and W. Ou, “Learning user representations with hypercuboids for recommender systems,” in WSDM, 2021.
[19] K. Deng, J. Huang, and J. Qin, “Box4rec: Box embedding for sequential recommendation;” in PAKDD, 2021.
[20] X. Li, L. Vilnis, D. Zhang, M. Boratko, and A. McCallum, “Smoothing the geometry of probabilistic box embeddings;” in ICLR, 2018.
[21] S. Dasgupta, M. Boratko, D. Zhang, L. Vilnis, X. Li, and A. McCallum, “Improving local identifiability in probabilistic box embeddings;” in NeurIPS, 2020.
[22] H. Jegou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search;” IEEE TPAMI, vol. 33, no. 1, pp. 117–128, 2010.
[23] T. Ge, K. He, Q. Ke, and J. Sun, “Optimized product quantization for approximate nearest neighbor search;” in CVPR, 2013.
[24] T. Chen, L. Li, and Y. Sun, “Differentiable product quantization for end-to-end embedding compression;” in ICML, 2020.
[25] T. Chen, M. R. Min, and Y. Sun, “Learning k-way d-dimensional discrete codes for compact embedding representations;” in ICML, 2018.
[26] G. Lee, C. Park, and K. Shin, “Set2box: Similarity preserving representation learning for sets (extended version);” in CVPR, 2022.
[27] A. Lai and J. Hockenmaier, “Learning to predict denotational probabilities for modeling entailment;” in ACL, 2017.

TABLE III: SET2BOX yields smaller MSE on average in the considered datasets than SET2BOX-ORDER.

| Method       | OC  | CS  | JI  | DI  |
|--------------|-----|-----|-----|-----|
| SET2BOX-ORDER| 0.0020 | 0.0033 | 0.0008 | 0.0027 |
| SET2BOX      | 0.0121 (-62%) | 0.0028 (-44%) | 0.0006 (-22%) | 0.0022 (-17%) |