CrossEM: A Prompt Tuning Framework for Cross-modal Entity Matching

Qin Yuan[§] Ye Yuan[§] Zhenyu Wen[†] Chi Chen[§] Guoren Wang[§]
[§]Beijing Institute of Technology [†]Zhejiang University of Technology
yuanq1020@gmail.com; yuan-ye@bit.edu.cn; wenluke427@gmail.com;
chenchibit@outlook.com; wanggrbit@126.com

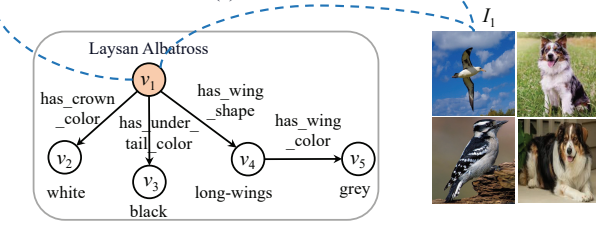
***Abstract*—Entity matching (EM) aims to identify equivalent entities across different data sources. Current EM assumes that these data are either homogeneous with aligned schema or heterogeneous but can be transformed into a unified modality. There is an urgent need to consider the entities with different modalities to support practical application scenarios over data lakes such as multi-modal data integration and recommendation system. It is impractical to unify their data modalities. To support EM on heterogeneous entity with different data formats and modalities, we propose cross-modal entity matching in this paper. Inspired by the promising performance achieved by recent pre-trained models, we perform cross-modal entity matching by prompt-tuning pre-trained multi-modal large models (MMLMs) in an unsupervised manner. However, the prompt-tuning faces three challenging issues: (i) objective gap between pre-training and tuning of MMLMs; (ii) data modality gap between the inputs of MMLMs and our matching task; (iii) prompt efficiency on large data. Therefore, we firstly propose a novel EM framework (namely, CrossEM) that addresses cross-modal EM as a matching probability problem with specific prompt-tuning. Secondly, two alternative prompt generation methods are designed to extract structural knowledge from heterogeneous data to overcome the data modality gap with pre-trained models. Thirdly, we present an improved matching framework (namely, CrossEM$^+$) to boost the prompt efficiency on large heterogeneous data. Experimental evaluations verify that our methods significantly outperform the state-of-the-art approaches on three benchmarks. Furthermore, our case study highlights the considerable potential of cross-modal EM in improving the performance of downstream tasks, thereby benefitting a wider range of research areas.**

I. Introduction

Entity Matching (EM) aims to identify whether two entities from different data sources refer to the same real-world entity, which is one of the fundamental and significant tasks in data management [1], [2]. Most existing solutions assume that two sources are either homogeneous with aligned schema or heterogeneous but can be transformed into a unified format. For example, [3], [4] study EM over relational tables with the aligned schema, and [5], [6] address EM across two knowledge graphs with similar structures. [7], [8], [9] perform EM over relational, semi-structured and textual data, while these can be unified into textual modality. However, due to the increasing data diversity in data repositories such as data lakes [10], there is an urgent need to generalize the entity matching problem to more practical scenarios [7]. Taking animal entities as an example, the attribute information of animals is usually stored in structured relational tables or semi-structured graphs, while their visual characteristics are depicted through unstructured images or videos. Figure 1(a) shows two types of birds in colors, wings, origins and foods. Figures 1(b) and (c) illustrate a portion of knowledge graph and images related to animals, respectively. It is impractical to unify the modalities of these data because they have different data formats and visual data in the form of pixel matrix is schemaless. On the other hand, it is difficult to represent these various data into a common feature space, which requires the introduction of expensive representation learning models in the preprocessing step.

	Bird Name	Primary Color	Wing Shape	Origin	Food
t_1	Laysan Albatross	white	long	Antarctica	fishes
t_2	Woodpecker	black	spotted	North America	insects

(a) Table: Birds



(b) A part of animals knowledge graph (c) A set of images

Fig. 1. Example of cross-modal entity matching.

In this paper, we study *cross-modal entity matching* that attempts to match cross-modal entities to support more practical application scenarios over data lake such as multi-modal knowledge graph creation [6], [11], [12] and recommendation system [13]. Some works [7], [8], [9], [10] study generalized entity matching (GEM) to match entities in heterogeneous data sources. However, they cannot support the multi-modal scenarios mentioned above due to the modality differences (e.g., visual and textual). Recent works [4], [6], [7] have achieved promising results for EM by leveraging the power of pre-trained large models (LMs) and prompt-tuning paradigm. Nevertheless, they do not directly support cross-modal EM due to the need for high-quality labeled examples [7] or limitations in data modality [4], [6]. To the best of our knowledge, cross-modal EM considering multiple heterogeneous data with different formats and modalities such as graph and images remains unexplored.

To overcome the modality differences and label requirements, pre-trained multi-modal large model (MMLM) is a

good choice. MMLMs recently have shown impressive performance in various multi-modal understanding tasks, such as visual question answering [14] and text-to-vision generation [15]. These MMLMs are usually trained with large-scale natural language and visual data, and enable to understand and recognize user's intention well. Inspired by this, we attempt to transfer the multi-modal understanding capabilities of MMLM to address the problem of cross-modal entity matching. The challenges mainly include the following three aspects.

Challenge 1: *How to tune a pre-trained MMLM for cross-modal EM in an unsupervised manner (objective gap)?* Although fine-tuning and prompt-tuning techniques [16] have achieved success on matching problem [7], there is a significant objective gap between the pre-training and tuning on downstream tasks. Pre-trained MMLMs are usually trained by computing similarity or learning a common feature space. However, existing tune-based EM methods [4], [7] treat EM as a classification problem and address it by adding additional classification layers, which exploit a different objective from the pre-trained model. This objective gap affects the transfer and adaptation of MMLMs to cross-modal EM tasks.

Challenge 2: *How to effectively prompt rich knowledge in heterogeneous data to a pre-trained MMLM (data modality gap)?* Most pre-trained MMLMs are usually learned for vision and language matching tasks [17], [18], [19], there is a modality gap between the inputs of pre-trained models (unstructured textual data) and the heterogeneous data (semi-structured graph and structured tables illustrated in Figure 1). Some prompt-based methods first serialize entities with different formats into texts and then tune pre-trained language models to perform EM as a sequence classification task [4], [7]. This restricts the expression of structural knowledge entailed in graph and relational tables, and hinders the effectiveness of prompt-tuning MMLMs to cross-modal EM tasks.

Challenge 3: *How to efficiently perform cross-modal EM on large heterogeneous data (prompt efficiency)?* Prompt-tuning MMLMs to perform cross-modal EM usually requires consuming prohibitive GPU resources. To adapt EM methods to large-scale data, a prevalent approach is to split the EM dataset into multiple mini-batches and train the samples in each mini-batch independently [4], [5]. However, this destroys the structures and associations of entities, thereby adversely affects the matching results. Partitioning heterogeneous data to preserve entity associations and improve the prompt efficiency on large data is still a challenging endeavor.

To tackle the above three challenges, we propose a new prompt-tuning framework for cross-modal entity matching, namely CrossEM. In order to bridge the objective gap between pre-trained MMLM and cross-modal EM task (for C1), we address cross-modal EM as a matching probability problem with specific prompt-tuning, which has the same objective as MMLM. To break the modality gap between structured data and the inputs of MMLM (for C2), we propose structure-aware prompt generation to extract structural knowledge from heterogeneous data. To avoid expensive prompt-tuning on large data (for C3), we develop an improved matching framework CrossEM$^+$ by considering three efficiency issues, i.e., training scalability with huge candidate pairs, negative sampling in contrastive learning and effectiveness of generated prompts. Our contributions are summarized as follows:

(1) Prompt-based tuning framework. This is the first cross-modal EM framework that prompt-tunes pre-trained MMLMs to align entities from different modalities. As data diversity increases, this problem will become more important in more practical scenarios. Details will be shown in Section II.

(2) Structure-aware prompt generation. Two alternative graph prompting methods including discrete *hard-encoding prompt* and continuous *soft prompt* are designed for adapting pre-trained MMLMs to our cross-modal EM task. The former requires carefully design based on prior data and task knowledge, while the later is learned from the feedback of model on the task objective. One can flexibility select them depending on the realistic applications, introduced in Section III.

(3) Improved matching framework. We propose an improved framework CrossEM$^+$ to scale up cross-modal EM task on large heterogeneous data with the help of three optimizations: mini-batch generation, negative sampling and prompt constraint. CrossEM$^+$ improves the training efficiency and the matching accuracy over large data in a mini-batch training manner. Details will be presented in Section IV.

(4) Extensive experiments. We conduct comprehensive experimental evaluation on cross-modal EM task compared with some state-of-the-art approaches. Extensive experimental results verify the superiority of our proposed CrossEM and CrossEM$^+$ in terms of effectiveness and efficiency. Our case study indicates that cross-modal EM can significantly improve the accuracy of multi-modal knowledge graph integration. Details will be illustrated in Section V.

II. Problem and Framework Overview

A. Problem Formulation

Data lake. A data lake collects a set of heterogeneous data sources including structured (e.g., relational tables, Excels, etc.), semi-structured (e.g., Xml, Json, graph, etc.) and unstructured data (e.g., images, videos, texts, etc.).

Graph. We consider a directed graph defined as $G = (V, E, L)$, where (a) V is a finite set of vertices; (b) $E \subseteq V \times V$ is a set of edges; (c) L is the set of all unique words contained in the labels of edges and vertices; (d) $L(v)$ and $L(e)$ represent the labels of vertex $v \in V$ and edge $e \in E$, respectively.

Relational table. A relational table is denoted by a set of tuples T associated with a set of attributes. A relational database schema includes a collection of relational tables.

Json. A Json document is a collection of Json objects in the forms of key-value pairs.

During data preprocessing of the data lake, these structured and semi-structured data can be converted into graph in *data mapping* by treating the tuples of tables and the keys of Jsons as entities, the foreign keys of tables and the references of Jsons as relationships [10], [20], [21]. A graph is formulated by encoding entities into vertices and relationships as edges. For unstructured texts, some sentence parsing models based

on language structures [22], [23] can be used to construct a graph for named entities and their syntactic relationships.

Images. An image I captures the scene that includes various elements contributing to the overall understanding or interpretation of the image, such as objects, people, landscapes, backgrounds, etc. Multimedia data such as videos, can be divided into a set of images based on frames [24], [25].

Entity Matching. Entity matching [11] aims to determine whether two entities refer to the same real-world entity, which is regarded as a *pairwise matching problem* where two matched entities are called a *matching pair*. Embedding-based methods [4], [7] usually generate a vector representation for each entity so that related entities are close in the vector space. We formally define them as follows.

Definition 1: (Matching Pair) Given a data entity x, an embedding model M takes x as input and outputs its embedding vector $M(x)$. For each entity pair (x_1, x_2), we use a given similarity function sim to measure their embedding similarity, $sim(x_1, x_2)$ is large if and only if x_1 and x_2 are a *matching pair*, where sim is usually a cosine function.

In this paper, we consider entity matching task over data lake D. Using the above mentioned data mapping, structured and semi-structured data can be encoded as graph. Therefore, we can define our *cross-modal entity matching* problem based on graph and images, which is formulated as follows.

Definition 2: (Cross-modal Entity Matching) Given a graph data $G = (V, E, L)$ and an image repository $\mathbb{I} = \{I_i\}_{1 \le i \le N}$ with N images, *cross-modal entity matching* is to find all matching pairs $\mathcal{S}$ between the vertices of V and the images in $\mathbb{I}$ such that:

$$\mathcal{S} = \{(x_i, x_j) | x_i \Leftrightarrow x_j, x_i \in V, x_j \in \mathbb{I}\}, \tag{1}$$

where $\Leftrightarrow$ denotes the equivalent entities in the real world. (x_i, x_j) is a matching pair calculated by the similarity between their embeddings $M(x_i)$ and $M(x_j)$.

We do not assume a one-to-one mapping across two data sources [5], but mainly focus on whether two entities match or not between V and $\mathbb{I}$ [4]. For each source entity x_i, the target entity x_j is ranked according to their semantic similarity scores. The higher score indicates the more likelihood that x_i and x_j form a matching pair.

According to the data mapping, the left source entity may be a vertex of graphs, a tuple of tables or a key of Json objects, and the right target entity is an image entity. Due to the success of generalized entity matching such as Sudowoodo [4] and PromptEM [7], we no longer focus on the entity matching between graphs, tables and Json objects, but only on those entities with different modalities such as graph and image.

Example 1: As shown in Figures 1(b) and 1(c), v_1 and I_1 are a matching pair in our cross-modal EM setting, which both depict the entity "laysan albatross" but in different modalities. Besides, the tuple t_1 shown in Figure 1(a) also forms a matching pair with I_1 by treating t_1 as a graph [21], which illustrates "laysan albatross" with several characteristics such as "white color" and "long wings".

B. Prompt Tuning Large Models

In this paper, we aim to effectively build pairwise matching model to support our cross-modal EM in an unsupervised manner. Multi-modal large models (MMLMs) such as CLIP [17] and ALIGN [18], have recently shown their superiority in cross-modal retrieval for texts and images [19], [26], [27].

Contrastive Language-Image Pre-training (CLIP). CLIP is trained on large scale of multi-modal data and captures implicit correspondings between texts and images using rich multi-modal knowledge in a self-supervised manner [17]. CLIP is built in two encoders where one for images and another for texts. The image encoder is used to transform an image into a feature vector via either a CNN model such as ResNet [28], or a vision transformer such as ViT [29]. The text encoder takes as input a sequence of word tokens and produces a vectorized representation in Transformer architecture [30]. CLIP adopts a *contrastive loss* to learn a joint embedding space for the two modalities. Specifically, it minimizes the distance between similar pairs and maximizes those of dissimilar pairs.

Contrastive Loss. Assume that X_p and X_n are positive and negative example pairs, respectively. The contrastive loss maximizes the similarity of $(x_i, x_j) \in X_p$ in the numerator and minimizes the similarity values between x_i and other elements in the denominator. Let $N = N_1 \times N_2$ be the batch size, N_1 and N_2 are numbers of vertices and images within a batch. The sizes of X_p and X_n are N_1 and $N_1 \times (N_2 - 1)$, respectively. Therefore, the contrastive loss can be computed by averaging all positive pairs as follows.

$$L_{con} = -\frac{1}{2N_1} \sum_{(x_i, x_j) \in X_p} [l(x_i, x_j) + l(x_j, x_i)], \tag{2}$$

$$l(x_i, x_j) = -\log \frac{\exp(sim(x_i, x_j))/\tau}{\sum_{(x_i, x_k) \in X_n} \exp(sim(x_i, x_k))/\tau}. \tag{3}$$

where τ is a temperature hyper-parameter in the range $(0, 1]$. Generally, X_p is collected from the pairs with top similarity, and X_n is the remaining pairs within the same batch.

Prompt-based Tuning. Leveraging the transfer ability of CLIP, we can first define a *baseline* matching model that predicts the matching probability between two entities via *prompt tuning* the CLIP on a downstream task-specific data set [16]. For each vertex v in V, a naive prompt tuning function Pro is used to generate a prompt Pro(v) with a specific text template, such as "A photo of [MASK]" where "[MASK]" is a placeholder of vertex labels in G. For instance, "A photo of laysan albatross".

To determine whether a vertex $v \in V$ and an image $I \in \mathbb{I}$ form a matching pair, the baseline first generates a textual prompt Pro(v) for v and then feeds it into the text encoder M_T of CLIP. Meanwhile, I is fed into the image encoder M_I of CLIP. Subsequently, the matching probability of v and I is computed as follows:

$$p(v, I) = \frac{\exp(\tau \cdot < M_T(\text{Pro}(v)), M_I(I) >}{\sum_{I_i \in \mathbb{I}} \exp(\tau \cdot < M_T(\text{Pro}(v)), M_I(I_i) >}, \tag{4}$$

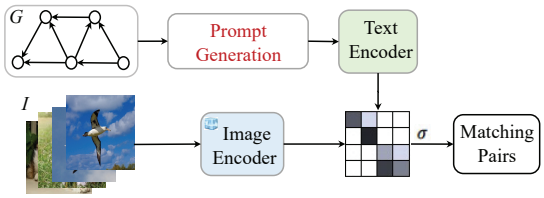


Fig. 2. Overall architecture of prompt-tuning framework CrossEM.

where $< \cdot, \cdot >$ denote the cosine similarity function and τ is a temperature hyper-parameter in range of $(0, 1]$.

Since the CLIP is pre-trained to gain semantic understanding capability for visual and textual data, it is expected that the CLIP can be transferred to perform cross-modal EM over graph data and images in an unsupervised prompting manner. The advantage of prompt tuning over fine-tuning is that it can quickly adapt to new tasks, has more efficiently computing ability and lower overfitting risk [16].

C. Framework Overview

The baseline prompt-tuning approach mentioned above works reasonably well, but it is sub-optimal for our cross-modal EM task. The main reason is that the CLIP [17] has pre-trained to capture the similarity between textual items and images. Due to the schema heterogeneity of vertices in the data lake, vertex labels (e.g., animal ids) in the naive prompt Pro are too simple to adequately represent the semantics of vertices. Without tuning for CLIP, it cannot perceive the vertex semantics, further affecting the quality of EM results.

Our framework CrossEM aims at prompt-tuning the pre-trained CLIP model to address the cross-modal EM problem. We present CrossEM by introducing the structural knowledge of vertices in the graph, where the image encoder M_I and the contrastive loss in the CLIP are frozen.

The main components of CrossEM are summarized in Figure 2 to provide an overview, and details are illustrated in Algorithm 1. Taking a heterogeneous graph G and a target image set $\mathbb{I}$ as inputs, CrossEM performs EM cross G and $\mathbb{I}$ to return matching pairs $\mathcal{S}$. For each pair of vertex and image (v, I), a prompt for v is created by summarizing its associated structural knowledge via *prompt generation* (line 5), and then fed into the text encoder to obtain vector representation (line 6). At the same time, I is represented by leveraging the image encoder M_I (line 7). Subsequently, contrastive learning and Equation 4 are used to decide whether they are a matching pair (lines 8-10). Finally, all matching pairs are computed and returned by the trained matching model M. To adapt different application requirements, we design two alternative prompt generation mechanisms, detailed in Section III.

Furthermore, consider the large and heterogeneous multi-modal data, CrossEM faces huge challenges in terms of training speed and GPU memory occupation. To improve the prompt efficiency of CrossEM and the quality of matching set, we design three optimizations to propose an improved matching framework (namely, CrossEM$^+$) in Section IV.

Algorithm 1: CrossEM

Input: A graph $G = (V, E, L)$, a set of images $\mathbb{I}$, a pre-trained CLIP model with text encoder M_T and image encoder M_I, number of training epoches n
Output: A prompt-tuning model M

1 $\mathcal{S} \leftarrow \emptyset$;
2 **for** *epoch from 1 to* n **do**
3 Randomly split entity pairs into mini-batches B;
4 **for** $B_i = (V_i, \mathbb{I}_i) \in B$ **do**
5 Pro$(V_i) \leftarrow \{\text{Pro}(v_j) | v_j \in V_i\}$;
6 $X_v \leftarrow M_T(\text{Pro}(V_i))$;
7 $X_I \leftarrow M_I(\mathbb{I}_i)$;
8 $P \leftarrow \{p(v, I) | v \in V_i, I \in \mathbb{I}_i\}$;
9 $L \leftarrow L_{con}(X_v, X_I, P)$;
10 $M \leftarrow$ back-forward(Pro, L);
11 **return** M;

Taking inputs G and $\mathbb{I}$, CrossEM$^+$ first introduces *mini-batch generation* (in Section IV-A) to block entity pairs and increase entity locality, so that most entities can find their associations within the same mini-batch and unrelated entities can be pruned during training (for line 3). By doing this, CrossEM$^+$ expects that the training efficiency on large-scale data can be improved in a mini-batch training manner. And then, *negative sampling* (in Section IV-B) samples a set of negative entity pairs to enhance the contrastive learning ability of CLIP by considering the neighbors of vertices and the local information of images (for line 8). During prompt learning, an *orthogonal prompt constraint* (in Section IV-C) is introduced for vertices to boost the quality of prompt creation by considering the orthogonality of prompts in a mini-batch (for lines 9-10). Finally, we can obtain the set of matching pairs based on the pre-trained contrastive model.

III. Prompt Generation

In this section, we detail how to generate a prompt for each vertex of G to tune the text encoder of CLIP. We first introduce structure-aware prompt in Section III-A and then present two alternative prompt functions in Sections III-B and III-C.

A. Structure-aware Prompt

Instead of exclusively using vertex labels as the baseline prompt mentioned in Section II-B, we propose *structure-aware prompt generation* to improve the prompt quality for each vertex. For example, the labels "laysan albatross" and "white" of vertices v_1 and v_2 represent the animal name and attribute, respectively. They cannot completely capture the entity information of "laysan albatross", such as colors and wings. Motivated by that, we generate prompts for vertices by summarizing their expressive neighborhood information.

We give the general mathematical definition of structure-aware prompting function as below.

Definition 3: (Structure-aware prompting function) A prompting function is used to encode a prompt for each vertex, defined by f_{pro}. For each $v \in V$ and its associated neighboring subgraph, the prompt is denoted by $f_{\text{pro}}(v)$.

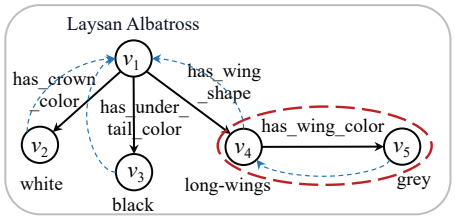


Fig. 3. Example of the hard prompt in CrossEM.

Here d-hop subgraph is considered for the vertices in G. Specifically, for each vertex $v \in V$ and an integer d, the d-hop subgraph of v is denoted by $d(v) = (V_d, E_d)$, which is induced by the vertices V_d within d hops of v, and the edge set E_d consists of all edges with two endpoints in V_d.

As discussed in early works [6], [31], choosing an appropriate prompting function has great impacts on the performance of downstream task. We design two alternative graph prompting methods including discrete *hard-encoding prompt* and continuous *soft prompt*, which separately correspond to two different work mechanisms of *sequence-based text encoder* and *feature-based text encoder* shown in Figure 4 and will be detailed later.

B. Hard-encoding Prompt

The key idea of hard-encoding prompt is to first extract the subgraph for each vertex based on graph traversal algorithms, and then format it as a textual template based on a pre-defined concatenate operation.

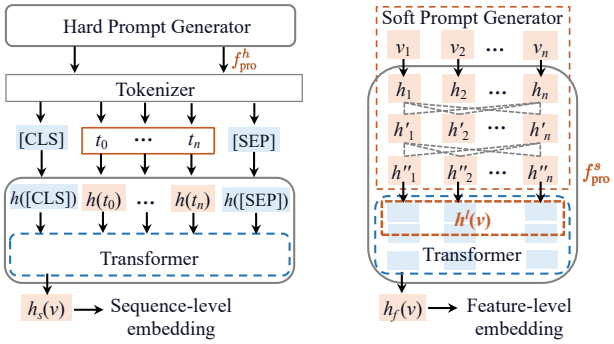
Specifically, for a vertex v and its d-hop subgraph $d(v) = (V_d, E_d)$, the hard-encoding prompt is designed by concatenating its d-hop neighbors and relationships. The *hard-encoding prompt* of v is defined by function $f^d_{\text{pro}}(v)$ as follows:

$$f^h_{\text{pro}}(v) = \text{Concat}(S, T), \tag{5}$$

where S is a set of *neighboring sub-prompts* for all neighbors of v in V_d and T is a set of pre-defined tokens. Concat is a concatenate operation for all sub-prompts in S though the specific tokens of T. Each sub-prompt $s_i \in S, i = \{1, ..., l\}$ is induced by the direct neighbors of v_i based on the breadth first search algorithm, where l is the number of neighbors of v within V_d. The following illustrates a detailed example.

Example 2: Considering the vertex v_1 and its d-hop subgraph represented by $d(v_1) = (V_d, E_d)$. Figure 3 shows the induction process of $f^d_{\text{pro}}(v_1)$ for v_1 based on $d(v_1)$. The blue dashed lines indicate the directions where the neighboring sub-prompts are inducted toward v_1. The red dashed cycle examples a sub-prompt associated with v_4, i.e., s_4 = "long-wings has wing color in grey". $f^h_{\text{pro}}(v_1)$ is denoted as "Laysan Albatross has crown color in white, has under tail color in black, has wing shape in long-wings, and long-wings has wing color in grey". Here $S = \{s_1, s_4\}$ and $T = \{$"in", "and"$\}$, it is formulated by concatenating S with the tokens in T.

Sequence-based text encoder. Figure 4(a) illustrates the workflow of the text encoder taking hard prompts as inputs. After textual prompt is generated for each v, the process are divided into the following three steps: 1) the prompt is first serialized as $\{$[CLS], $f^h_{\text{pro}}(v)$, [SEP]$\}$ where [CLS] and [SEP] are special tokens originally designed in [32], which are used to mark the beginning and end of the sequence; 2) The sequence is converted as a set of tokens $\{$[CLS], $t_0, t_1, ..., t_n$, [SEP]$\}$ in a tokenizer; 3) The original text encoder M_T extracts token embeddings $\{h(\text{[CLS]}), h(t_0), h(t_1), ..., h(\text{[SEP]})\}$ and finally outputs the head projection of h([CLS]) as the encoded sequence-level vertex embedding $h_s(v)$ of v. This embedding represents the sequence as a whole and can be compared with image embeddings extracted from M_I to perform our task, as shown in Section II-B.



(a) Sequence-based text encoder (b) Feature-based text encoder

Fig. 4. Workflow of text encoder with two different prompts. The orange zones represent (a) the discrete hard prompts and (b) the continuous soft prompts. The blue zones are the original text encoder.

Discussion. Though the hard-encoding prompt function f^h_{pro} is often simple and effective for prompting M_T, there are still three drawbacks that need to be addressed. (1) The hard-encoding prompt template needs to be carefully designed for different graph structures and even heterogeneous data entities. (2) M_T is initially trained on input tokens with a maximum length of 77, which means that some token-level features in f^h_{pro} will be truncated, thereby potentially losing important structural information of entities to harm the matching performance. (3) The hard-encoding prompt is essentially a serialization-like approach, and the concatenation order of sub-prompts is also an important factor affecting the representations of vertex semantics.

C. Soft Prompt

To release these limitations mentioned above, we present *soft prompt* to generate continuous prompt for each vertex by summarizing its structural information into a feature vector. And then, the prompt is used to incorporate as part of the input of text encoder.

As for the vertex v, the soft prompt is extracted by aggregating its d-hop neighbors. The *soft prompt* of v is defined by function $f^s_{\text{pro}}(v)$ as follows:

$$f^s_{\text{pro}}(v) = \alpha \cdot h(v) + (1 - \alpha) \sum_{v_j \in N(v_i)} h(v_j), \tag{6}$$

where α is the aggregation weight and $h(v)$ is the representation of v. Benefit from graph representation methods such as GraphSage [33] and GNN [34], the structural feature of vertex $h(v)$ is captured by aggregating information of its neighbors. We initialize each embedding by utilizing pre-trained language models such as BERT [32] and Ro-BERTa [35].

Feature-based text encoder. Figure 4(b) depicts the workflow of text encoder taking soft prompts as a part of inputs. After structural prompt feature f^s_{pro} is generated for each v, we modify the first several layers of the original text encoder M_T to take the label tokens l of vertices and f^s_{pro} as inputs. For each $v \in V$, this process consists of the following three steps: 1) to introduce textual information of vertices, M_T first encodes its label l_v as token positional embedding $h(l_v)$, which is obtained by adding the special token [CLS] in the beginning of l_v; 2) $h(l_v)$ is concatenated with the structural feature embedding $f^s_{\text{pro}}(v)$ to form the input $h^l(v)$ of the Transformers in M_T; 3) the output $h_f(v)$ of the last hidden layer in the Transformer is treated as the feature-level vertex embedding of v.

Specifically, the concatenation of token positional embedding $h(l_v)$ and the structural feature embedding $f^s_{\text{pro}}(v)$ is formulated as follows.

$$h^l(v) = ReLU \cdot (W \cdot (h(l_v) \oplus f^s_{\text{pro}}(v))), \tag{7}$$

where $\oplus$ and ReLU are a concatenate operation and an activation function, respectively. $W \in \mathbb{R}^{|V| \times (d_l + d_v)}$ is a weight matrix where d_l and d_v are the dimensions of textual and structural embeddings. Therefore, the input shape of the transformer in M_T is extended from the original d_l to $(d_l + d_v)$.

Discussion. Our proposed CrossEM essentially computes the candidate pairs of vertices and images one by one. As discussed in [36], current entity matching methods [4], [5] face efficiency issues when dealing with tens of thousands or even millions of entities. Especially, the multi-modal data processed in CrossEM is larger in magnitude and more heterogeneous. The time complexity of training CrossEM (when using soft prompt) with m epochs is $m|\mathbb{I}||V|(|V_d| + |E_d|)$, where $|V_d|$ and $|E_d|$ are the average number of vertices and edges in d-hop subgraphs for vertices in G. CrossEM faces huge challenges to the training efficiency and GPU memory occupation when processing huge graph (e.g., Freebase [37]) and images (e.g., ImageNet [38]) in real-world data lake.

IV. Improved Matching Framework

To improve the training efficiency of CrossEM and the quality of matching set, we design three optimizations by (1) introducing data preprocessing to improve the training efficiency for huge candidate pairs in data lake; (2) negative sampling to collect usefully distinctive pairs for contrastive learning; and (3) designing prompt constrains to create more effective prompts for vertices during the soft prompt learning.

For (1), we propose a task-specific **mini-batch generation** technique to increase entity locality so that most entities can find their associations within the same mini-batch. The motivation is to improve the accuracy of entity matching and the training efficiency on large data in a mini-batch training manner. For (2), we present a **negative sampling** method to sample a set of negative pairs by considering the neighbors of vertices and local information of images. We expect that the negative samples becomes more difficult to distinguish, thus forcing the contrastive representation model to learn more meaningful features and have stronger discriminant ability. For (3), we introduce an **orthogonal prompt constraint** for soft prompt generation to render the prompts generated for different vertices as possible as irrelevant with each other.

Figure 5 summarizes the main components of the improved matching framework CrossEM$^+$ to provide an overview. It takes as inputs G and $\mathbb{I}$, and performs cross-modal EM in the following four steps. (1) In the mini-batch generation, the candidate pairs of vertices in V and images in $\mathbb{I}$ are divided into some data partitions by data preprpcessing. Details in Section IV-A. (2) In the negative sampling, for each partition $(V_i, \mathbb{I}_i)$, a set of images $\mathbb{I}'_i$ that have similar local information with the neighbors of vertices in V_i while are distinct from V_i are sampled, as presented in Section IV-B. (3) During prompt learning in a mini-batch manner, the orthogonal constraint is introduced for vertices in V_i and backward propagates to advance the learning of soft prompt generation, as introduced in Section IV-C. (4) We finally obtain the set of matching pairs based on the frozen image encoder and contrastive model.

A. Mini-batch Generation

Our mini-batch generation approach aims at: (i) splitting a large-scale candidate pairs of V and $\mathbb{I}$ into K partitions; and (ii) leading every vertex entity and its potentially associated images to appear in the same mini-batch.

Motivation. The local features of entities contain a lot of specific characteristics that benefits from identifying and representing entities, such as patches of images and the neighbors of vertices, which we call *properties* of entities. Intuitively, candidate pairs with more semantically close properties are more likely to be similar. For instance, the vertex v_1 and the image I_1 shown in Figures 1(b) and 1(c) separately depict the "laysan albatross" entity in two modalities. By considering the 1-hop neighbors of v_1, we can get the property set of v_1 as $P(v_1) = \{v_2, v_3, v_4\}$. From the top of Figure 6(a), we obtain the property set of I_1 as $P(I_1) = \{c^1_1, ..., c^1_p\}$. The image I_2 shown in the bottom of Figure 6(a) illustrates a "woodpecker" entity and has properties $P(I_2) = \{c^2_1, ..., c^2_p\}$. Compared with $P(I_2)$, $P(I_1)$ has more patches close to the items of $P(v_1)$, such as "white crown" (denoted by v_2 in $P(v_1)$ and c^1_1 in $P(I_1)$) and "black tail" (denoted by v_3 in $P(v_1)$ and c^1_p in $P(I_1)$). In this way, I_1 is more likely than I_2 to be partitioned into the same mini-batch as v_1.

Main idea. To generate mini-batches for improving the efficiency of prompt training over V and $\mathbb{I}$, we present a *property-based closeness partition* approach, namely PCP. Taking a graph G and a set of images $\mathbb{I}$ as inputs, PCP outputs a mini-batch data set $D = \{(V_i, \mathbb{I}_i) | V_i \subseteq V, \mathbb{I}_i \subseteq \mathbb{I}, 1 \leq i \leq k\}$. The overall workflow of PCP is illustrated in Figure 7, which works at the following three phrases. (1) Phase 1: *property closeness calculation*. It extracts property representations for vertices in

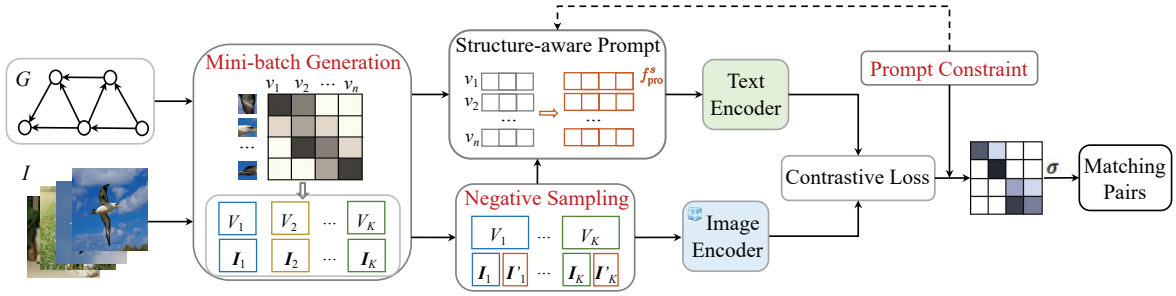
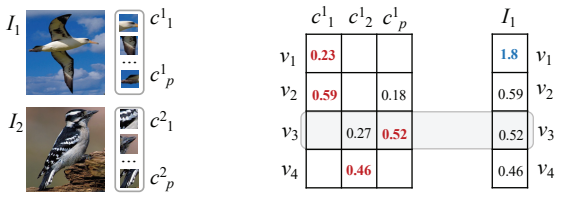
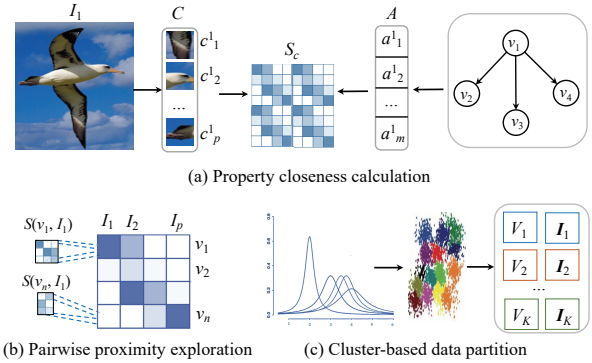


Fig. 5. Architecture of improved matching framework CrossEM$^+$. The red fonts indicate modules optimized for CrossEM.



(a) Image properties (patches) (b) Part of matrices S_c and $S(V, \boldsymbol{I})$

Fig. 6. Example of property and pairwise proximity matrix in PCP.



(a) Property closeness calculation

(b) Pairwise proximity exploration (c) Cluster-based data partition

Fig. 7. Overview of PCP for mini-batch generation.

V and images in $\mathbb{I}$, and computes a matrix S_c indicating the closeness of properties between vertices and images, as shown in Figure 7(a). (2) Phase 2: *pairwise proximity exploration*. As for a candidate vertex-image pair (v, I), the matching degree is regarded as the sum of their property closeness in S_c. Therefore, a matrix S that represents the pairwise proximity for the candidate pairs of vertices in V and images in $\mathbb{I}$ can be computed based on S_c, as depicted in Figure 7(b). (3) Phase 3: *cluster-based data partition*. It blocks image and vertex pairs into a set of mini-batches to increase entity locality, so that most vertices can find their associated images within the same mini-batch and unrelated candidate pairs can be pruned. To do this, the probability distribution of these images with respect to the vertices is computed by S, and then clusters them to generate a set of partitions, as illustrated in Figure 7(c). Details of the mini-batch generation are presented in Algorithm 2.

Phase 1: property closeness calculation (lines 1-3). Leveraging pre-trained vision model such as ResNet [28], we can crop every image I_i in $\mathbb{I}$ as a set of patches C^i and extract visual features for each patch $c_j \in C^i$, $j \in [1, p_i]$ where p_i is the number of patches in I_i. Thereby, a set of patch features can be obtained in $C = \{C_i | 1 \leq i \leq |\mathbb{I}|\}$ (line 1). Meanwhile, a set of property features A can be obtained based on vertex labels by using a pre-trained language model such as BERT [32], i.e., $A = \{A_i | 1 \leq i \leq |V|\}$ (line 2). Subsequently, the property closeness matrix S_c is computed by $A \times C$ with respect to all patches and vertex labels. $S_c \in \mathbb{R}$ is a matrix with dimension of $|V| \times p|\mathbb{I}|$ (line 3), where p is the average number of patches for all images in $\mathbb{I}$. The time and space complexities of this phase are both $O(p \cdot |V||\mathbb{I}|)$.

For example, considering v_1 and I_1 shown in Figures 1(b) and 6(a), respectively. Figure 6(b) (left) shows a property closeness matrix S_c for v_1 and I_1, which is calculated based on $P(v_1)$ and $P(I_1)$ mentioned above. Here S_c with dimensions of (4×3), where $S_c[1, 0] = 0.59$ represents the closeness between vertex v_2 and patch c^1_1 of I_1.

Phase 2: pairwise proximity exploration (lines 4-10). For a candidate vertex-image pair (v, I), we compute its proximity value $S(v, I)$ by aggregating the property closeness values in S_c for neighbors of v and patches of I as follows.

$$S(v, I) = \sum_{v_j \in N(v)} \max_{c_k \in P(I)} S_c[v_j, c_k]. \tag{8}$$

Specifically, $S(v, I)$ for v and I is calculated in the following three steps: (1) we first obtain a neighbor set $N(v)$ of v as the union of itself (treated as self-neighbor) and its d-hop neighbors V_d, i.e., $N(v) = \{v\} \cup V_d$ (line 6); (2) For each $v_j \in N(v)$, the closeness between v_j and I is calculated as the maximum value of v_j to all patches $P(I)$ of I (lines 7-9); (3) $S(v, I)$ for v and I is obtained by summarizing the closeness values of all neighbors to I. The time complexity and space complexity of this phase are $O(|V| \cdot (|V_d| + |E_d| + p|\mathbb{I}|))$ and $O(|V||\mathbb{I}|)$, respectively, where p is the average patch number of images in $\mathbb{I}$.

For example, the red numbers shown in Figure 6(b) (left) are the closeness values between vertices and images. For instance, $S(v_3, I_1) = max(0.27, 0.52) = 0.52$. Considering v_1 and its neighbor set $N(v_1) = \{v_1, v_2, v_3, v_4\}$, we can inductively obtain the proximity value of the candidate pair (v_1, I_1)

Algorithm 2: Mini-batch generation

Input: A graph $G = (V, E, L)$, a set of images $\mathbb{I}$, a pre-trained language model M_a and a vision model M_v, threshold of proximy θ, number of vertex subset k_1, cluster number k_2 of images
Output: Mini-batch data set $D = \{D_1, ..., D_b\}$

1 Extract patch features C for $\mathbb{I}$ using M_v;
2 Obtain property features A for V in G using M_a;
3 Compute property closeness matrix S_c by $A \times C$;
4 Initialize matrix $S(V, \mathbb{I})$ as 0 for all $v \in V$ and all $I \in \mathbb{I}$;
5 **for** $v \in V$ **do**
6 $N(v) \leftarrow \{v\} \cup \{v_j | v_j \in V_d\}$;
7 **for** $I \in \mathbb{I}$ **do**
8 **for** $v_j \in N(v)$ **do**
9 $S(v_j, I) \leftarrow \max_{c_k \in P(I)} S_c[v_j, c_k]$;
10 $S(v, I) \leftarrow \sum_{v_j \in N(v)} S(v_j, I)$;
11 $D \leftarrow \emptyset$;
12 Randomly divide V into $\{V_1, ..., V_{k_1}\}$;
13 **for** $V_i \in \{V_1, ..., V_{k_1}\}$ **do**
14 $\mathbb{I}'_i \leftarrow \mathbb{I} - \{\arg_{I \in \mathbb{I}} S(V_i, I) \leq \theta\}$;
15 Compute $P_i(\mathbb{I}'_i)$ for $\mathbb{I}'_i$ and V_i based on $S(V_i, \mathbb{I}'_i)$;
16 $C_I \leftarrow$ k-means$(P_i, \mathbb{I}'_i, k_2)$;
17 $C_I \leftarrow$ shuffle(C_I);
18 $D \leftarrow D \cup \{(V_i, \mathbb{I}_j) | \mathbb{I}_j \leftarrow C_{I_k}, C_{I_k} \in C_I, 1 \leq k \leq k_2\}$;
19 **return** D;

as $S(v_3, I_1) = 0.23 + max(0.59, 0.18) + max(0.27, 0.52) + 0.46 = 1.8$, shown in the blue number in Figure 6(b) (right).

Phase 3: cluster-based data partition (lines 11-18). As for vertices V and images $\mathbb{I}$, we partition the candidate vertex-image pairs into a set of mini-batch D in the following four steps. (1) We first randomly divide V into a set of vertex subset (line 12). (2) For each $V_i \subseteq V$, we obtain the candidate images $\mathbb{I}'_i$ by pruning irrelevant images with low proximity values to the vertices in V_i. This is because the pairs formed by these images and the vertices in V_i may have limited improvement for the discriminative ability of the model (line 13). (3) Our cluster-based partition method divides images into a set of image subsets (line 15-16). (4) For each image subset $\mathbb{I}_j \subseteq \mathbb{I}$, data partition is formed by collecting and shuffling the pairs of vertices in V_i and images in $\mathbb{I}_j$.

Next, we detailed present our cluster-based partition method as follows. Intuitively, images with close proximity values contain more similar patches and have more similar matching probabilities with respect to the vertices in V_i. As for $V_i \subseteq V$ and $\mathbb{I}_j \subseteq \mathbb{I}$, we define the pairwise proximity for V_i and $\mathbb{I}_j$ as a set of probability distributions in the size of $|\mathbb{I}_j|$, written as $P_i(\mathbb{I}_j)$. For each $I \in \mathbb{I}_j$, $P_i(I)$ depicts a normalized distribution of pairwise matching probabilities between I and V_i, which is computed by $S(v, I)$ for I to all $v \in V_i$. Therefore, we compute the proximity distribution $P_i(\mathbb{I}'_i)$ for each V_i, and then divide $\mathbb{I}'_i$ by using a cluster algorithm (such as k-means) such that images with similar distribution places in the same partition.

The time complexity of this phase is $O(k_1 \cdot \log|V| \cdot |\mathbb{I}|)$, and space complexity is $O(\log|V| \cdot \log|\mathbb{I}|)$, where k_1 is the size of vertex subsets in V.

Totally, our proposed mini-batch generation takes time complexity and space complexity are $O(p \cdot |V||\mathbb{I}| + k_1 \cdot \log|V| \cdot |\mathbb{I}|)$ and $O(p \cdot |V||\mathbb{I}| + \log|V| \cdot \log|\mathbb{I}|)$, respectively. By introducing mini-batch generation, candidate pairs used for the training of CrossEM$^+$ are reduced from $|V||\mathbb{I}|$ in CrossEM to $\log|V| \log|\mathbb{I}|$. Therefore, regarding the prompt learning and contrastive representation as black boxes, the time complexity of m training epochs in CrossEM$^+$ becomes $O(m \log|V| \log|\mathbb{I}|)$.

Algorithm 3: Property-based negative sampling

Input: Data partition D, a matrix $S(V, \mathbb{I})$, batch size N
Output: A mini-batch training set B

$B \leftarrow \emptyset$;
for $D_i \in D$ **do**
  $D_i \leftarrow$ shuffle(D_i);
  $B_i \leftarrow B_i \cup D_i$;
  $count \leftarrow$ ceil$(|D_i|/N) \cdot N - |D_i|$;
  if $count = 0$ **then**
    Continue;
  for $v \in D_i.V$ **do**
    obtain random integer number k;
    $\mathbb{I}^s_i \leftarrow \arg_I \text{top}_k S[v, :]$;
    $Pair \leftarrow \{(v, I_j) | I_j \in \mathbb{I}^s_i, I_j \notin D_i.\mathbb{I}\}$;
    for $(v, I) \in Pair$ **do**
      if $count > 0$ **then**
        $B_i \leftarrow B_i \cup \{(v, I)\}$;
        $count \leftarrow count - 1$;
  $B \leftarrow B \cup$ shuffle(B_i);
$B \leftarrow$ shuffle(B);
return B;

B. Negative Sampling

As mentioned in Section II-C, the default sampling method in contrastive learning is to uniformly sample data items to obtain negative examples [4], [17]. This is not sufficient enough when the pairs associate with the same properties but are not similar in practice, as it mainly checks whether the pairs contain overlapping properties and does not capture the more important features that distinguish one from others.

For example, two images I_1 and I_2 illustrated in Figure 6(a) depict the entities "laysan albatross" (i.e., v_1) and "woodpecker", respectively. Both of them contain the property "black tail", as shown in the patches c^1_p and c^2_p. However, we can note that the property "spots" shown in c^2_1 is a more important visual feature that can help distinguish I_1 from I_2. Therefore, (v_1, I_2) trends to be sampled as a harder negative example and put into the same mini-batch as (v_1, I_2). The model is guided to learn this essential discriminant feature "spots" such that brings I_1 and v_1 close and separates I_2 from v_1.

We sample a set of negative pairs that are more difficult to distinguish, so that essential features of entities can be learned to enhance the discriminitive ability of the model. A *property-based negative sampling* method is proposed and works as follows. Taking inputs as the data partitions $D = \{D_1, ..., D_b\}$

and the proximity matrix $S(V, \mathbb{I})$ created in Section IV-A, it outputs a set of sampled images $\mathbb{I}'_i$ for each D_i. The sampling procedure is illustrated in Algorithm 3. Specifically, for each partition $D_i = (V_i, \mathbb{I}_i)$ in D, we select images $\mathbb{I}^s_i$ that have higher proximity values to the vertices in V_i but are not contained in $\mathbb{I}_i$. And then, $\mathbb{I}^s_i$ is merged into $\mathbb{I}_i$ to form candidate pairs until the number of pairs reaches the nearest integer multiple of the batch size N (lines 9-15). During this process, we shuffle candidate pairs within batches (line 16) and batches in data partitions (lines 3 and 17) to reduce the dependence of the model on data and further ensure the model's generalization ability.

In this module, the time complexities of shuffle operation and selection are $O(\log|V| \cdot \log|\mathbb{I}|)$ and $O(\log|\mathbb{I}| \cdot \log k)$, respectively, where k is the random integer number and $k \ll \log|\mathbb{I}|$. There, the total time complexity is $O(\log|V| \log|\mathbb{I}|)$.

C. Orthogonal Prompt Constraint

As presented in Section III-C, structure-aware prompts are learned by vertices and their neighbor structures. Considering two entities v_1 and I_2 that contain similar structural features but do not form a matching pair. For example, entities "laysan albatross" and "woodpecker" both have wings, crowns and bellies, but in different colors. We intuitively expect the two entities to have distinct prompts to reduce misleading of the model.

We introduce an *orthogonal prompt constraint* into CrossEM$^+$, where prompts for different vertices are as irrelevant to each other as possible. During prompt learning, the orthogonal constraint backward propagates to advance the learning of soft prompt creation, so that the contrastive model can further guide vertices to correctly align with images.

Specifically, the soft prompt $f^s_{pro}(v_i)$ for each vertex v_i is first initialized as the token embedding of its label $L(v_i)$ using pre-trained language model such as BERT [32] and Ro-BERTa [35]. And then, the constraint keeps orthogonality for each vertex during prompt updating. The loss function associated with the constraint is formulated as follows.

$$L_o = \sum_{i=0}^{|B|} ||f^s_i \cdot (f^s_i)^T - I_E||_{F_1}, \tag{9}$$

where I_E is an identify matrix and the F_1 norm is used for element-level calculations. f^s_i is a prompt matrix created by stacking all prompts of vertices in a mini-batch, i.e., $f^s_i = \{f^s_{pro}(v)|v \in B_i.V\}$.

To integrate prompt constraint into our CrossEM$^+$, we linearly combine the contrastive loss with the orthogonal constraint. Formally, the combined loss function is

$$L = \beta \cdot L_c + (1 - \beta) \cdot L_o, \tag{10}$$

where β is the hyper-parameter controlling the weights of two loss functions.

TABLE I
DATASET STATISTICS.

Datasets	# Vertices	# Edges	# Tuples	# Images
CUB [39]	512	3,245	312	11,788
SUN [40]	819	2,130	717	16,594
FB2K-IMG [42]	2,667	8,382	-	20,455
FB6K-IMG [42]	6,342	30,884	-	44,813
FB10K-IMG [42]	10,856	78,747	-	69,629

V. EXPERIMENTS

In this section, we experimentally evaluate our proposed CrossEM and CrossEM$^+$ on three real-world datasets in the following aspects: (i) the overall performance of CrossEM and CrossEM$^+$ in accuracy, efficiency and scalability; (ii) ablation study for different modules of our proposed methods; and (iii) a case study in multi-modal knowledge graph integration.

A. Experimental Settings

Datasets and Evaluation Metrics. We utilize three publicly available datasets CUB (Caltech UCSD Birds 200, [39]), SUN (SUN Attribute, [40]) and FB15K-237-IMG [41]. CUB and SUN are two standard cross-modal semantic grounding benchmarks equipped with attributes, where CUB includes 11,788 images of 200 birds with 312 attributes, SUN has 16,549 images from 717 scene classes with 102 attributes. We use the same train-test splits of vertices as in [42] for CUB and SUN datasets. FB15K-237-IMG is a popular datasets in multi-modal knowledge completion where the graph is a subset of the large-scale knowledge graph Freebase [37] and each vertex associated with 10 images. We derive three subset datasets from it to verify the efficiency and scalability of our methods, namely FB2K-IMG, FB6K-IMG and FB10K-IMG, which include 54M (million), 284M, 755M entity pairs, respectively. Detailed statistics are shown in Table I.

To evaluate the performance of CrossEM and CrossEM$^+$, we use the following metrics: (i) Hits@k (k=1, 3, 5, H@k for short) and Mean Reciprocal Rank (MRR) are employed for the accuracy evaluation. Higher H@k and MRR indicate better performance. (ii) Running time (T for short) in seconds and the maximum GPU Memory usage (Mem for short) in GB are used to evaluate the training efficiency on different datasets. Here, the running time means the average training time of each epoch in every approach. We use the NVIDIA Nsight Systems to monitor the occupation of GPU memory.

Competitors. To demonstrate the performance of our proposed method, we typically compare to three types of state-of-the-art multi-modal approaches for a comprehensive evaluation. (1) Dual encoder methods, which directly measure the distance of cross-modal representations, such as CLIP [17] and ALIGN [18]. (2) Fusion encoder methods, which map multi-modal data into a common feature space, including VisualBERT [26], ViLBERT [27], IMRAM [19] and TransAE [43]. (3) Prompt-tuning methods, which tune pre-trained model by specific prompts, containing GPPT [31], CrossEM with hard prompt (w/ f^h_{pro}), soft prompt (w/ f^s_{pro}) and CrossEM$^+$.

Specifically, CLIP [17] is to contrastively learn a transferable language-image pre-training model from natural lan-

guage supervision by pairing images with relevant language descriptions. ALIGN [18] trains large-scale models using large amounts of noisy text data to scale up vision-language representation learning for various tasks such as image classification. VisualBERT [26] consists of a stack of Transformer layers that implicitly align elements of an input text and regions in an associated input image with self-attention. ViLBERT [27] is a pre-trained visual-language model that processes both visual and textual inputs in separate streams, and interacts through co-attention transformer layers for learning joint representations. IMRAM [19] utilizes recurrent attention memory to integrate information from both text and image modalities for cross-modal image-text retrieval. TransAE [43] combines multi-modal auto-encoder with TransE to encode the visual and textual knowledge into the unified representation, where the hidden layer of the auto-encoder is used to be entity representations in the TransE model. We modify these model by serializing the graph into texts as presented in our hard prompt in Section III-B. GPPT [31] is a supervised graph prompt model that generalizes graph representation model to downstream graph tasks. We modify its task objective to binary classification objective like previous EM works [7] and provide feedback in a supervised manner. For the first four baseline methods, we use pre-trained models to predict the results on test dataset. For the remaining methods, we leverage the source code from the original repositories to produce the results.

To demonstrate the contributions of different modules, we conduct ablation study and design five variants introduced as follows. For prompt generation, we compare with different prompt mechanisms, i.e., CrossEM w/ f_{pro}^{h} and CrossEM w/ f_{pro}^{s}. For mini-batch generation in CrossEM^{+}, we compare with CrossEM^{+} w/o MBG by replacing our mini-batch generation (MBG) module with randomly partition for entity pairs. For negative sampling in CrossEM^{+}, we compare with CrossEM^{+} w/o NS by replacing our property-based negative sampling (NS) method with uniformly sampling pairs as introduced in Section II-A. For orthogonal prompt constraint (OPC) in CrossEM^{+}, we compare with CrossEM^{+} w/o OPC by removing OPC from CrossEM^{+}.

To describe the application of our cross-modal EM task, we perform a case study to illustrate the advantage of our methods in multi-modal knowledge graph integration by comparing with the following approaches, i.e., ViLBERT [27], TransAE [43], DistMult [44], RotatE [45] and RSME [46] mentioned in [47], as well as MKGformer [47], our CrossEM w/ f_{pro}^{h}, CrossEM w/ f_{pro}^{s} and CrossEM^{+}. The state-of-the-art MKGformer [47] integrates visions and texts via coarse-grained prefix-guided interaction and fine-grained correlation-aware fusion modules for knowledge graph completion tasks.

Implementation Details. We implement our methods in PyTorch [48] and Huggingface [49]. Unless particularly specified, we use RoBERTa-base model [35] as the pre-trained LM and AdamW as the optimizer for all the experiments. We use the CLIP model equipped with the ViT/32 image encoder [29] and 12 Transformer layers in the text encoder [50]. In the soft prompt generation, we experimentally set the structural feature extraction in Equation 6 to GNN [34] in CUB and SUN datasets, and to GraphSage [33] in FB15k dataset. During prompt learning, we extend the maximum length of input tokens from the originally 77 to 512. We fix the projector dimension of text and image to 768 and 512, respectively. In mini-batch generation, we take the ResNet18 [28] pretrained on ImageNet as the backbone to extract the patch features for each image without fine-tuning. The learning rate is set to 0.0005, the batch size is set to 256, the number of epochs is set to 30. We tune the hyper-parameters by doing a grid search and selecting the one with the best performance. Specifically, the aggregation weight of soft prompt α and the loss weight β are both continuously selected from [0, 1] with a step size of 0.1 each time. All experiments are conducted on a machine with an Intel Core i9-10900K CPU, a NVIDIA GeForce RTX3090 GPU with 24GB memory. We report all results of competitors in the optimal settings.

B. Overall Performance.

Exp-1: Accuracy of entity matching. To demonstrate the accuracy of our proposed methods CrossEM and CrossEM^{+}, we compare them with the seven competitors above mentioned on three real datasets.

From Table II, we have the following three findings. (1) CrossEM w/ f_{pro}^{h}, CrossEM w/ f_{pro}^{s} and CrossEM^{+} have higher accuracy than others on CUB and SUN datasets. Compared with the dual encoders and single encoders, CrossMat^{+} at least has 15.91% and 45.29% advancements in Hits@1 average on three datasets, respectively. CrossEM^{+} also has higher accuracy than others on FB2K-IMG dataset. It verifies the superiority of our framework, especially on the attribute-equipped graph data like CUB and SUN. (2) The results of CrossEM^{+} has higher accuracy than CLIP (baseline formulated in Section II-B) on all different datasets. There are 15.91%, 9.86% and 0.14 improvements in Hits@1, Hits@5 and MRR values average on three datasets, respectively. It verifies the superiority of prompts in our framework. (3) CrossEM w/ f_{pro}^{s} is more accurate than CrossEM w/ f_{pro}^{h} on CUB and SUN datasets. There are average 4.69% improvements in Hits@1. CrossEM w/ f_{pro}^{h} has more higher accuracy than CrossEM w/ f_{pro}^{s} on FB2K-IMG dataset, and has 6.95%, 9.63% and 0.08 improvements on Hits@1, Hits@5 and MRR values, respectively. Meanwhile, CrossEM w/ f_{pro}^{s} is only 11.76%, 11.77% and 0.12 away from CrossEM^{+} with the best performance. This verifies that our hard and soft prompts are alternative on different realistic applications and datasets.

Exp-2: Efficiency of prompt learning. To verify the efficiency, we evaluate the average training time T and the maximum GPU usage Mem for each epoch on three datasets.

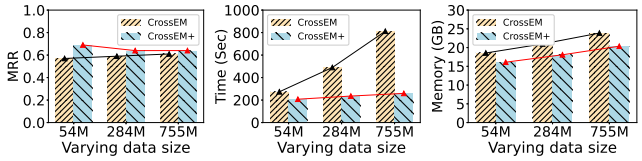
As illustrated in Table III, we can find that (1) CrossEM^{+} takes less training time for each epoch than others on all datasets, and its average efficiency is improved by about 22% (reduced by 35.67 seconds) compared to the sub-optimal method. This is because CrossEM^{+} trains matching model in a mini-batch manner, so that entities can find their equivalences within the same mini-batch to improve training efficiency. (2)

TABLE II
OVERALL ACCURACY ON DIFFERENT DATASETS.

Methods	CUB				SUN				FB2K-IMG			
	H@1	H@3	H@5	MRR	H@1	H@3	H@5	MRR	H@1	H@3	H@5	MRR
Dual encoder approaches												
ALIGN [18]	33.5	55.5	66.0	0.48	27.04	45.62	50.67	0.38	24.505	35.237	40.286	0.32
CLIP [17]	68.00	82.00	84.00	0.74	26.39	40.28	43.06	0.31	62.06	75.40	78.52	0.66
Fusion encoder approaches												
VisualBERT [26]	14.00	19.00	24.00	0.17	3.12	17.29	24.36	0.13	21.70	32.40	43.90	0.27
ViLBERT [27]	24.10	37.50	53.30	0.56	2.40	13.71	18.24	0.11	23.30	33.50	45.70	0.26
TransAE [43]	4.20	11.45	16.30	0.39	19.42	21.50	29.30	0.22	19.80	37.60	44.10	0.35
IMRAM [19]	5.90	16.80	20.10	0.12	16.50	27.30	34.20	0.31	24.80	39.61	42.80	0.36
Prompt-tuning approaches												
GPPT [31]	16.94	19.77	25.32	0.19	3.64	8.92	15.91	0.07	1.21	5.34	11.63	0.08
CrossEM w/ f^h_{pro}	72	90	94	0.79	51.39	58.33	58.33	0.54	60.43	70.59	77.54	0.65
CrossEM w/ f^s_{pro}	78.0	94.0	94.0	0.84	54.78	**59.51**	**59.72**	**0.58**	53.48	63.64	67.91	0.57
CrossEM$^+$	**82.00**	**94.00**	**96.00**	**0.86**	**56.94**	58.33	59.48	0.57	**65.24**	**76.47**	**79.68**	**0.69**

TABLE III
OVERALL EFFICIENCY ON DIFFERENT DATASETS.

Methods	CUB		SUN		FB2K-IMG	
	T	Mem	T	Mem	T	Mem
TransAE [43]	44	11.3	194	12.7	237	16.4
IMRAM [19]	86	16.7	647	19.9	461	23.8
CrossEM w/ f^s_{pro}	53	10.5	404	11.7	274	18.6
CrossEM$^+$	**42**	**9.3**	**118**	**10.2**	**208**	**16.1**



(a) MRR (b) Time (c) Memory

Fig. 8. Scalability in different data size of FB15K-237-IMG.

CrossEM$^+$ takes less GPU memory than others on all datasets, and its average save GPU about 7.29% (reduced by 0.93 GB) compared to the sub-optimal method. The reason is that during mini-batch training most irrelevant entity pairs are pruned and do not need to calculate gradients through them. (3) Compared with CrossEM w/ f^s_{pro}, Cross$^+$ improves the average training time and GPU usage by about 51% (reduced by 124 seconds) and 12.72% (reduced by 1.73 GB) on the three datasets, respectively. The results show that although the effectiveness of entity matching is limited by the backbone model CLIP, CrossEM$^+$ significantly improves the model efficiency through the optimization techniques proposed in Section IV.

Exp-3: Scalability in different data size. To further investigate the scalability of our proposed methods, we evaluate on the FB15K-IMG dataset with different scales of 54M, 284M and 755M vertex-image pairs.

Figure 8 reports the accuracy (using MRR value), training time and GPU memory usage of CrossEM$^+$ and CrossEM w/ f^s_{pro} (marked as CrossEM in the legend). We can see that (1) CrossEM$^+$ has higher accuracy, takes less training time and GPU memory than CrossEM w/ f^s_{pro} at all scales of data sizes. This is because (i) the mini-batch training increases entity locality so that entities can find their equivalences in the same mini-batch as much as possible; (ii) our improved method CrossEM$^+$ reduces the training complexity from quadratic to logarithmic in one epoch, as discussed in Section IV-A; (2) With the increasing of data sizes, the training time and GPU memory usage in both CrossEM w/ f^s_{pro} and CrossEM$^+$ increase, while the growth in CrossEM$^+$ is more slow. CrossEM w/ f^s_{pro} always has higher accuracy than CrossEM$^+$ in different data scales. These verify that CrossEM$^+$ outperforms CrossEM w/ f^s_{pro} in terms of scalability.

Overall, our proposed method CrossEM$^+$ achieves an average improvement of 15.91% in Hits@1 over the baseline, and brings 51% and 12.72% improvements on training time and GPU usage over CrossEM. Therefore, CrossEM$^+$ outperforms other competitors in terms of accuracy and efficiency, and scales well on different data sizes.

C. Ablation Study.

We further conduct ablation study to verify the effectiveness of different modules in CrossEM and CrossEM$^+$. We compare CrossEM with two prompt designs (i.e.,w/ f^h_{pro} and w/ f^s_{pro}), and compare CrossEM$^+$ with its variants without the key module, i.e., mini-batch generation (MBG), negative sampling (NS) and orthogonal prompt constraint (OPC). The results are listed in Table IV.

Effect of different prompts on CrossEM. As shown in the 1-st and 2-nd rows, CrossEM w/ f^h_{pro} and CrossEM w/ f^s_{pro} have close Hits and MRR values on all different datasets. This verifies that our proposed two prompts are alternative.

Effect of MBG on CrossEM$^+$. From the 3-rd and 6-th rows in Table IV, we can obviously see that MBG can significantly improves the training efficiency without reducing matching accuracy. For example, compared with CrossEM$^+$ w/o MBG, CrossEM$^+$ reduces the average training time by 152 seconds and GPU memory usage by 0.95 GB on the three datasets. This demonstrates the advantage of MBG.

Effect of NS on CrossEM$^+$. From the 4-th and 6-th rows in Table IV, we know that (1) CrossEM$^+$ w/o NS takes more GPU memory usage than CrossEM$^+$. (2) In most cases, CrossEM$^+$ w/o NS requires more training time but has slightly lower accuracy than CrossEM$^+$.

Effect of OPC on CrossEM$^+$. From the 5-th and 6-th rows in Table IV, we can obviously see that CrossEM$^+$ w/o OPC

TABLE IV
ABLATION STUDIES IN ACCURACY FOR DIFFERENT COMPONENTS OF OUR METHODS.

	CUB					SUN					FB2K-IMG				
Methods	H@1	H@5	MRR	T	Mem	H@1	H@5	MRR	T	Mem	H@1	H@5	MRR	T	Mem
CrossEM w/ f^h_{pro}	72	94	0.79	-	-	51.4	58.3	0.54	-	-	60.43	70.59	0.65	-	-
CrossEM w/ f^s_{pro}	78	94	0.84	53	10.53	56.9	59.7	0.58	404	11.65	53.48	63.64	0.57	273	18.56
CrossEM$^+$ w/o MBG	82	96	0.86	61	9.36	23.6	29.1	0.25	443	11.45	64.71	**77.54**	**0.70**	321	16.53
CrossEM$^+$ w/o NS	82	96	0.86	**33**	10.29	56.9	59.7	0.58	173	10.36	64.17	75.40	0.68	264	18.15
CrossEM$^+$ w/o OPC	81	96	0.86	59	10.53	56.9	59.7	0.58	227	11.26	58.29	68.98	0.62	224	17.26
CrossEM$^+$ (full)	**82**	**96**	**0.86**	42	**9.33**	**56.9**	**59.7**	**0.58**	**118**	**10.20**	**65.24**	76.47	0.69	**208**	**16.11**

TABLE V
PERFORMANCE OF MULTI-MODAL KNOWLEDGE GRAPH INTEGRATION.

Methods	H@1	H@3	H@5	MRR
ViLBERT [27]	23.3	33.5	45.7	0.21
TransAE [43]	19.9	31.7	46.3	0.23
DistMult [44]	19.10	30.10	44.60	0.21
RotatE [45]	24.10	37.50	53.30	0.56
RSME [46]	24.20	34.30	46.70	0.24
MKGformer [47]	25.6	36.7	50.4	0.45
CrossEM w/ f^h_{pro}	60.43	70.59	77.54	0.65
CrossEM w/ f^s_{pro}	53.48	63.64	67.91	0.57
CrossEM+	**65.24**	**76.47**	**79.68**	**0.69**

has a slight decrease in accuracy compared to CrossEM$^+$, while taking more training time and GPU memory usage. This demonstrates the advantage of OPC.

D. Case Study: Multi-modal Knowledge Graph Integration.

Table V presents the advantage of our cross-modal EM over multi-modal knowledge graph integration on the FB15K-237-IMG dataset. We can see that (1) our proposed methods outperform other state-of-the-art approaches. For example, CrossEM$^+$, CrossEM w/ f^s_{pro} and CrossEM w/ f^h_{pro} are improved by 39.64%, 27.88% and 34.83% in Hits@1 value over MKGformer, respectively. (2) CrossEM$^+$ outperforms CrossEM w/ f^h_{pro}, followed by CrossEM w/ f^s_{pro}, with improvements of 4.81% and 11.76%, respectively. This demonstrates that cross-modal EM can benefit various downstream tasks such as multi-modal knowledge graph integration.

VI. RELATED WORK

Entity Matching. Entity Matching (EM) is one of the fundamental and significant tasks in data management. Existing EM works mainly divided into two categories in data perspective: traditional EM approaches that perform EM over homogeneous data with aligned schema [1], [2], [3], and generalized EM approaches that process EM over multiple heterogeneous data sources [7], [8], [9], [10], [21]. Generalized EM [7], [8] usually converts different sources into a unify data format. TDmatch [8] performs relational table and text document matching in an unsupervised learning way via graph creation and random walk. HER [21] links entities across a relational database and a graph based on a parametric simulation method. Machamp [9] benchmarks the GEM for different types such as structured tables, semi-structured, or textual data. Recently, representation learning technology has been used widely in EM task and achieved promising performance [4], [7]. PromptEM [7] unifies heterogeneous data as textual sequences and designs specific prompt-tuning to transform GEM as a masked language task to predict target words in a low-resource setting. Sudowoodo [4] fine-tuns language model to perform EM for relational tables by using a contrastive learning model. ZeroEA [6] achieves zero-training EM for knowledge graph based on pre-trained language models. However, modality heterogeneity and label requirements need to be addressed in cross-modal EM task.

Cross-modal Matching. Cross-modal matching approaches mostly are designed for visual and textual data to retrieve a set of related texts or images from another modalities. It can be typically divided into two categories: dual encoder methods that directly measure the distances of cross-modal representations such as CLIP [17], ALIGN [18], and fusion encoder methods that map these data into a common space via attention mechanism or generative adversarial network such as VisualBERT [26], ViLBERT [27], IMRAM [19]. Recently, multi-modal large models have received widespread attention and large-scale pre-trained encoders, e.g., CLIP [17], ALIGN [18] and Flamingo [51] have shown their superiority in cross-modal retrieval tasks. These approaches work well for texts and images, but directly extending them to our task has limited by the heterogeneity of data modality, especially graph and image data. Some works attempt to integrate graph and image, such as multi-modal knowledge graph completion [47], which mainly focuses on link prediction, relation extraction and entity recognition. These are orthogonal to our work. We aims to perform cross-modal entity matching by incorporating knowledge of graph structures with visual features.

VII. CONCLUSION

In this paper, we study the problem of cross-modal entity matching based on prompt-tuning, which is the first time to prompt-tune MMLM to align entities from different modalities. It is non-trivial as three challenging issues: (i) objective gap; (ii) data modality gap; and (iii) prompt efficiency. Therefore, our CrossEM framework designs two alternative prompt generation methods to break the objective and data modality gaps. To further improve the prompt efficiency on large data, CrossEM$^+$ is proposed by considering three optimizations: (i) the training scalability with huge candidate pairs, (ii) negative sampling in contrastive learning and (iii) effectiveness of generated prompts. Experimental evaluations demonstrate the superiority of CrossEM and CrossEM$^+$. In the future, we plan to explore a general prompt-tuning method to support more data management tasks such as data cleaning.

REFERENCES

[1] Y. Li, J. Li, Y. Suhara, A. Doan, and W. Tan, "Deep entity matching with pre-trained language models," *Proc. VLDB Endow.*, vol. 14, no. 1, pp. 50–60, 2020.

[2] Z. Miao, Y. Li, and X. Wang, "Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1303–1316.

[3] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park, G. Krishnan, R. Deep, E. Arcaute, and V. Raghavendra, "Deep learning for entity matching: A design space exploration," in *Proceedings of the 2018 international conference on management of data*, 2018, pp. 19–34.

[4] R. Wang, Y. Li, and J. Wang, "Sudowoodo: Contrastive self-supervised learning for multi-purpose data integration and preparation," in *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 2023, pp. 1502–1515.

[5] C. Ge, X. Liu, L. Chen, B. Zheng, and Y. Gao, "Largeea: Aligning entities for large-scale knowledge graphs," *Proc. VLDB Endow.*, vol. 15, no. 2, pp. 237–245, 2021.

[6] N. Huo, R. Cheng, B. Kao, W. Ning, N. A. H. Haldar, X. Li, J. Li, M. M. Najafi, T. Li, and G. Qu, "Zeroea: A zero-training entity alignment framework via pre-trained language model," *Proceedings of the VLDB Endowment*, vol. 17, no. 7, pp. 1765–1774, 2024.

[7] P. Wang, X. Zeng, L. Chen, F. Ye, Y. Mao, J. Zhu, and Y. Gao, "Promptem: Prompt-tuning for low-resource generalized entity matching," *Proc. VLDB Endow.*, vol. 16, no. 2, pp. 369–378, 2022.

[8] N. Ahmadi, H. Sand, and P. Papotti, "Unsupervised matching of data and text," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 1058–1070.

[9] J. Wang, Y. Li, and W. Hirota, "Machamp: A generalized entity matching benchmark," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 4633–4642.

[10] Q. Yuan, Y. Yuan, Z. Wen, H. Wang, C. Chen, and G. Wang, "Exploring heterogeneous data lake based on unified canonical graphs," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 1834–1838.

[11] Z. Sun, Q. Zhang, W. Hu, C. Wang, M. Chen, F. Akrami, and C. Li, "A benchmarking study of embedding-based entity alignment for knowledge graphs," *Proc. VLDB Endow.*, vol. 13, no. 11, pp. 2326–2340, 2020.

[12] R. Zhang, B. D. Trisedya, M. Li, Y. Jiang, and J. Qi, "A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning," *The VLDB Journal*, vol. 31, no. 5, pp. 1143–1168, 2022.

[13] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He, "A survey on knowledge graph-based recommender systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3549–3568, 2020.

[14] K. Marino, M. Rastegari, A. Farhadi, and R. Mottaghi, "Ok-vqa: A visual question answering benchmark requiring external knowledge," in *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, 2019, pp. 3195–3204.

[15] Y. Liu, K. Zhang, Y. Li, Z. Yan, C. Gao, R. Chen, Z. Yuan, Y. Huang, H. Sun, J. Gao *et al.*, "Sora: A review on background, technology, limitations, and opportunities of large vision models," *arXiv preprint arXiv:2402.17177*, 2024.

[16] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[17] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[18] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," in *International conference on machine learning*. PMLR, 2021, pp. 4904–4916.

[19] H. Chen, G. Ding, X. Liu, Z. Lin, J. Liu, and J. Han, "Imram: Iterative matching with recurrent attention memory for cross-modal image-text retrieval," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 12 655–12 663.

[20] Q. Yuan, Y. Yuan, Z. Wen, H. Wang, and S. Tang, "An effective framework for enhancing query answering in a heterogeneous data lake," in *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2023, pp. 770–780.

[21] W. Fan, L. Geng, R. Jin, P. Lu, R. Tugay, and W. Yu, "Linking entities across relations and graphs," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 634–647.

[22] P. R. Asveld, "Generating all permutations by context-free grammars in greibach normal form," *Theoretical computer science*, vol. 409, no. 3, pp. 565–577, 2008.

[23] Y. Gao, T. K. Huang, and R. J. Passonneau, "ABCD: A graph framework to convert complex sentences to a covering set of simple sentences," pp. 3919–3931, 2021.

[24] X. Shang, T. Ren, J. Guo, H. Zhang, and T.-S. Chua, "Video visual relation detection," in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 1300–1308.

[25] X. Sun, T. Ren, Y. Zi, and G. Wu, "Video visual relation detection via multi-modal feature fusion," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 2657–2661.

[26] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang, "Visualbert: A simple and performant baseline for vision and language," *arXiv preprint arXiv:1908.03557*, 2019.

[27] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," *Advances in neural information processing systems*, vol. 32, 2019.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[29] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[31] M. Sun, K. Zhou, X. He, Y. Wang, and X. Wang, "GPPT: graph pre-training and prompt tuning to generalize graph neural networks," in *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, A. Zhang and H. Rangwala, Eds. ACM, 2022, pp. 1717–1727.

[32] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186.

[33] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[34] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[35] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019.

[36] X. Zhao, W. Zeng, J. Tang, W. Wang, and F. M. Suchanek, "An experimental study of state-of-the-art entity alignment approaches," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2610–2625, 2020.

[37] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008, pp. 1247–1250.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[39] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-ucsd birds 200," 2010.

[40] G. Patterson and J. Hays, "Sun attribute database: Discovering, annotating, and recognizing scene attributes," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 2751–2758.

[41] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[42] Y. Xian, B. Schiele, and Z. Akata, "Zero-shot learning-the good, the bad and the ugly," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4582–4591.

[43] Z. Wang, L. Li, Q. Li, and D. Zeng, "Multimodal data enhanced representation learning for knowledge graphs," in *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*. IEEE, 2019, pp. 1–8.

[44] B. Yang, W. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[45] X. Zhou, Y. Yi, and G. Jia, "Path-rotate: Knowledge graph embedding by relational rotation of path in complex space," in *10th IEEE/CIC International Conference on Communications in China, ICCC 2021, Xiamen, China, July 28-30, 2021*. IEEE, 2021, pp. 905–910.

[46] M. Wang, S. Wang, H. Yang, Z. Zhang, X. Chen, and G. Qi, "Is visual context really helpful for knowledge graph? A representation learning perspective," in *MM '21: ACM Multimedia Conference, Virtual Event, China, October 20 - 24, 2021*, H. T. Shen, Y. Zhuang, J. R. Smith, Y. Yang, P. César, F. Metze, and B. Prabhakaran, Eds. ACM, 2021, pp. 2735–2743.

[47] X. Chen, N. Zhang, L. Li, S. Deng, C. Tan, C. Xu, F. Huang, L. Si, and H. Chen, "Hybrid transformer with multi-level fusion for multimodal knowledge graph completion," in *SIGIR '22: The 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, Madrid, Spain, July 11 - 15, 2022*, E. Amigó, P. Castells, J. Gonzalo, B. Carterette, J. S. Culpepper, and G. Kazai, Eds. ACM, 2022, pp. 904–915.

[48] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035.

[49] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020*, Q. Liu and D. Schlangen, Eds. Association for Computational Linguistics, 2020, pp. 38–45.

[50] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[51] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 23 716–23 736, 2022.