How to Talk to Language Models: Serialization Strategies for Structured Entity Matching

Haoteng Yin*
Purdue University
yinht@purdue.edu

Jinha Kim and Prashant Mathur and Krishanu Sarker and Vidit Bansal
Amazon
{jinhak,pramathu,kssarker,bansalv}@amazon.com

Abstract

Entity matching (EM), which identifies whether two data records refer to the same real-world entity, is crucial for knowledge base construction and enhancing data-driven AI systems. Recent advances in language models (LMs) have shown great potential in resolving entities with rich textual attributes. However, their performance heavily depends on how structured entities are "talked" through serialized text. The impact of this serialization process remains underexplored, particularly for entities with complex relations in knowledge graphs (KGs). In this work, we systematically study entity serialization by benchmarking the effect of common schemes with LMs of different sizes on diverse tabular matching datasets. We apply our findings to propose a novel serialization scheme for KG entities based on random walks and utilize LLMs to encode sampled semantic walks for matching. Using this lightweight approach with open-source LLMs, we achieve a leading performance on EM in canonical and highly heterogeneous KGs, demonstrating significant throughput increases and superior robustness compared to GPT-4-based methods. Our study on serialization provides valuable insights for the deployment of LMs in real-world EM tasks.

1 Introduction

Entity matching (EM) aims to identify whether two data records refer to the same real-world entity, even if their descriptions differ (Herzog et al., 2007). As a core challenge in data cleaning and integration, EM has a wide range of applications, from knowledge base construction to empowering data-driven AI systems (Diefenbach et al., 2018; Guu et al., 2020; Frey et al., 2023). The flourishing of knowledge-based AI applications (e.g., recommendation, question answering, and information retrieval) has particularly driven the need to integrate entities from different knowledge graphs (KGs). Classic EM tasks (Getoor and Machanavajjhala, 2012) focus on structured data in relational tables with homogeneous schemas, while more recent studies (Köpcke et al., 2010; Sun et al., 2020; Wang et al., 2021) have expanded to semi-structured (e.g., JSON, XML), unstructured (e.g., text) and complex relational data (e.g., KGs), as well as entities with missing and/or noisy attributes (see Fig. 1). The complexity of EM lies in identifying and linking inconsistent, incomplete, denormalized records across multiple data sources. Typically, EM involves two steps: *blocking*, which eliminates clear non-matches to reduce the number of pairwise comparisons, and *matching*, which identifies true matches from the filtered candidate set. While rule-based (Fan et al., 2009; Singh et al., 2017) and traditional learning-based (Konda et al., 2016) matchers work well for tabular data, they struggle with tasks that involve noisy or unstructured data (Mudgal et al., 2018). Neural network-based methods, particularly those that utilize word embeddings and sequence models (Manning, 2017), have demonstrated their effectiveness in matching entities with rich textual attributes (Parikh et al., 2016; Ebraheem et al., 2018).

Recent research (Brunner and Stockinger, 2020; Li et al., 2020; Peeters and Bizer, 2021; Paganelli et al., 2022; Akbarian Rastaghi et al., 2022; Wang et al., 2022; Zeakis et al., 2023; Peeters and Bizer, 2025; Wang et al., 2025; Wadhwa et al., 2024; Huang and Zhao, 2024) has explored using pretrained encoder models and large language models (LLMs) to resolve and match entities based on their textual attributes. Modern language models can generate highly contextualized embeddings that capture semantic meaning across the entire input, greatly alleviating the word ambiguity problem when comparing entity attributes. Particularly, with more grounded knowledge and generalization capability, LLMs show better performance than classic pretrained encoder models on challenging

*Work done during an internship at Amazon.

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 7851–7865

EM tasks (Mudgal et al., 2018; Wang et al., 2021; Jiang et al., 2024b). However, existing LLM-based matchers use chat-based interactions, which are severely limited by high latency, computational complexity, and API call cost. As a result, previous studies are unable to *fully evaluate* the target LLM without downsampling the test set (Peeters and Bizer, 2025; Li et al., 2024).

To use language models (LMs) for matching, entities must first be serialized into sequences, a process known as *entity serialization*. For instance, a restaurant record comes with attributes of "title", "address", and "phone". Its serialization can be done by sequentially concatenating all attributes, which imposes an artificial order, placing "title" ahead of "address" and "phone". This practice breaks the permutation invariance of the attributes, which has unknown effects on model utility. Nested attributes (e.g., "address" consists of "street", "city", and "zipcode") in semi-structured data are often flattened, which loses hierarchical information and dilutes key signals. These issues further deteriorate for graph-structured data, where serialization often results in either significant loss of structural information or increased overhead to preserve relations between entities in sequences.

So far, no comprehensive study has been performed to analyze the serialization effect on LMs for structured entity matching. Existing schemes to serialize tabular and semi-structured data are ad hoc (Brunner and Stockinger, 2020; Li et al., 2020; Wang et al., 2021; Peeters and Bizer, 2025), forcing attributes into predefined orders that may distort their inherent non-sequential and/or hierarchical relationships as shown in the above examples. Such practices potentially hamper the model's understanding of the entity, thereby harming its utility. While structured entities with complex relations are commonly serialized through verbal descriptions (Agarwal et al., 2021; Fatemi et al., 2023; Wang et al., 2023; Wadhwa et al., 2024), which greatly limits the model's ability to generate rich and concise representations for matching, particularly for knowledge graph entities with hyper-relations (Galkin et al., 2020).

In this study, we systemically benchmark seven common serialization schemes (see Table 1) for LMs on tabular and semi-structured data and explore new strategies for LLMs to efficiently perform matching on entities with complex relations. There are three key questions to be answered: **RQ1** whether attribute order matters under the settings

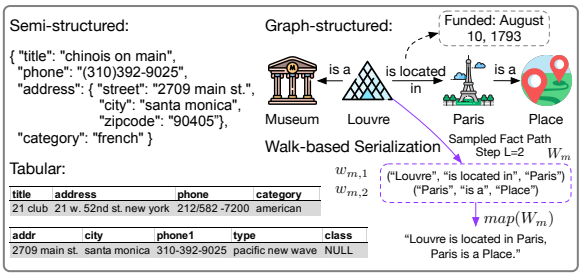

Figure 1: Real-world matching tasks involve various types of structured entities, posing great challenges in their serialization for applying language models.

of zero-shot and with fine-tuning; **RQ2** how different ways of handling missing or dirty values affect the model; and **RQ3** are special tokens in serialization helpful for matching structured entities. Based on our findings, we propose a novel serialization scheme for LLMs based on random walks: it efficiently captures both semantic and structural contexts from KG entities, leading to robust entity embeddings for matching. Our results show that open-source LLMs with the proposed walk-based strategy achieve state-of-the-art (SoTA) performance with superior throughput for matching entities in four canonical and highly heterogeneous KG datasets (Sun et al., 2020; Jiang et al., 2024b).

Our contributions are summarized as follows: (1) We are the first to systematically study the effect of entity serialization on LMs for matching across tabular, semi-structured, and graph-structured data. (2) We empirically evaluate and compare how serialization schemes affect LMs with different sizes and backbones. (3) We propose a scalable paradigm for applying open-source LLMs with walk-based serialization to generate robust embeddings for matching KG entities, outperforming previous SoTA GPT-4-based systems by 2995× speedup.

2 Preliminaries and Prior Work

2.1 Notations & Problem Formulation

Entity matching (EM) takes two collections D and D' of data records as input and outputs a set $M \subseteq D \times D'$ of entity pairs, where each pair $(e, e') \in M$ is identified as the same real-world entity. An entity e is a set of key-value pairs $e = \{(\text{attr}_i, \text{val}_i)\}_{i=1}^{k}$, where attr_i denotes the attribute name and val_i is the corresponding value in base types of number, string or list. *Classic EM* (Getoor and Machanavajjhala, 2012; Christen, 2012) assumes that entities in relational tables have a homogeneous schema, where e and e' share the

same set of attributes $\{\text{attr}_i\}_{i=1}^{k}$; while *generalized EM* (Wang et al., 2021) removes this assumption, allowing entities e to have a heterogeneous schema, with nested or unstructured attributes.

In addition to tables and semi-structured data, another classic type of structured entity is stored in graphs (see Fig. 1). In particular, *knowledge graph* (KGs) $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ is an organized representation of real-world entities $\mathcal{E}$ and their relationship $\mathcal{R}$ through triplets of facts $(e_h, r, e_t) \in \mathcal{F}$, where $e_h, e_t \in \mathcal{E}$ and $r \in \mathcal{R}$. The integration of real-world entities from KGs is known as *entity alignment*: Given $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$ and $\mathcal{G}' = (\mathcal{E}', \mathcal{R}', \mathcal{F}')$, the task is to find the identical entity set $\{(e, e') | e \in \mathcal{E}, e' \in \mathcal{E}'\}$, where each pair (e, e') represents the same real-world entity but exists in different KGs.

2.2 Language Models for Entity Matching

Pretrained encoder models, including BERT (Devlin et al., 2019) and RoBERTa (Liu, 2019), show great abilities in natural language understanding for various downstream tasks. For EM tasks, the model takes a pair of entities serialized by the function $\mathcal{S}(\cdot)$ as input and is fine-tuned with objectives in two common settings: (1) cross-encoder: the joint of serialized entities e, e' is fed into model f_{LM} *simultaneously*, followed by a classifier g predicting "match" or "no match" (Li et al., 2020; Brunner and Stockinger, 2020; Wang et al., 2021; Peeters and Bizer, 2021) as $g(f_{\text{LM}}(\mathcal{S}(e, e'))) \rightarrow \{0, 1\}$; (2) bi-encoder: serialized entities e, e' are fed to model f_{LM} *independently* and their similarity is computed by a scoring function d (or neural networks) over their embeddings (Paganelli et al., 2022; Zeakis et al., 2023) as $d(\mathbf{h}_e, \mathbf{h}_{e'}) \in [0, 1]$, where $\mathbf{h}_e = f_{\text{LM}}(\mathcal{S}(e))$.

LLMs, including GPT (Radford et al., 2019) and Llama (Touvron et al., 2023), are designed primarily for language generation. They often show greater generalizability and zero-shot performance than pretrained encoder models on new tasks, which is particularly useful for EM tasks with limited samples for training or out-of-distribution samples for inference. Several studies have utilized LLMs to solve EM problems through chat-based interactions, but suffer from high latency and inference cost (Peeters and Bizer, 2025; Li et al., 2024). To save prompting token budgets, serialization schemes adopted in these works are all plain text. Fan et al. (2024); Wang et al. (2025); Huang and Zhao (2024); Jiang et al. (2024a) studied how to cost-effectively deploy LLMs for matching but mainly focused on improving throughput by prompt engineering or blocking techniques. However, the bottleneck of text generation ultimately limits their applicability.

Index	Scheme	Attribute Order	Special Tokens	NULL
S1	Fixed Order (Li et al., 2020)	Fixed	[COL], [VAL]	✓
S2	Random Order	Random	[COL], [VAL]	✓
S3	Pairwise Order (Naeim abadi et al., 2023)	Pairwise	[COL], [VAL]	✓
S4	Valid Value (Wang et al., 2021)	Fixed	[COL], [VAL]	x
S5	Plain Format (Brunner and Stockinger, 2020)	Fixed	x	x
S6	Span Typing (Li et al., 2020)	Fixed	[COL], [VAL], [LAST]	✓
S7	JSON Format (Sisaengsuwanchai et al., 2023)	Fixed / Nested	x	x

Table 1: Summary & Comparison of Common Serialization Schemes for Structured Entities.

2.3 Prior work on Serialization

The serialization process converts an entity e from a set of key-value pairs $\{(\text{attr}_i, \text{val}_i)\}_{i=1}^{k}$ into a meaningful sequence that can be ingested by LMs. Existing serialization schemes can be divided into three categories based on entity structure types.

Serialization for Tabular Data. For pretrained BERT models, the standard scheme for serializing entities in relational tables is introduced by Li et al. (2020): $\mathcal{S}(e) \rightarrow$ "`[COL]` attr_1 `[VAL]` $\text{val}_1 \ldots$ `[COL]` attr_k `[VAL]` val_k", where `[COL]` and `[VAL]` are special tokens indicating the start of attribute names and values, respectively. This scheme is flexible in serializing tabular data with homogeneous and heterogeneous schema. To serialize an entity pair (e, e'), let $\mathcal{S}(e, e') \rightarrow$ "`[CLS]` $\mathcal{S}(e)$ `[SEP]` $\mathcal{S}(e')$ `[SEP]`", where `[SEP]` is the token that separates two sequences and `[CLS]` is the token used in BERT models to encode serialized entity pairs into a joint representation for classification.

Multiple schemes are developed with variations in **(1)** Attribute order: S1 *Fixed order* concatenates attributes sequentially following the given order of the table schema (Wang et al., 2021; Miao et al., 2021; Hegselmann et al., 2023). S2 *Random order* permutes entity attributes during serialization, leading to misaligned attributes between pairs of serialized entities. S3 *Pairwise order* puts the values of common attributes together in serialized entity pairs (Naeim abadi et al., 2023) and fills the mismatched attributes with the value `NULL` as $\mathcal{S}_p(e, e') \rightarrow$ "`[CLS]` `[COL]` attr_1 `[VAL]` val_1^e, $\text{val}_1^{e'} \ldots$ `[COL]` attr_k `[VAL]` val_k^e, `NULL` `[SEP]`." **(2)** Special tokens: S4 *Valid value* removes attributes with missing values from the output (Wang et al., 2021). S5 *Plain format* omits `[COL]` and `[VAL]` tokens (Brunner and Stockinger, 2020) and sim-

ply concatenates the names and values of all attributes as $\mathcal{S}(e) \rightarrow$ "attr$_1$ val$_1$ ... attr$_k$ val$_k$." Due to its simplicity, this scheme is widely adopted to pair with matching query prompts for decoder-only models (Peeters and Bizer, 2025; Sisaengsuwanchai et al., 2023; Wang et al., 2024). S6 *Span typing* is introduced by Li et al. (2020) to inject domain knowledge into serialized entities by annotating the type of a span of their attribute values. Special tokens are inserted to reflect the type of the recognized span $\{(s_i, t_i, \text{type}_i)\}_{i \geq 1}$ from attribute values, where s_i, t_i are the start/end positions of the span with the annotated type. For instance, the phone number "(123)456-7890" can be replaced with "(123) 456 - [LAST] 7890 [/LAST]", where [LAST], [/LAST] are added to indicate the start/end of the last 4 digits of phone number that might help the model compare between entities.

Serialization for Semi-Structured Data. An attribute attr$_i$ of semi-structured entities can be either values in base types or an entity itself, such as JSON format with nested attributes. Its serialization can follow a similar fashion as tabular data, but differs in: the [COL] and [VAL] tokens are recursively added along with attribute names and values nested at each level i (underlined) as $\mathcal{S}(e) \rightarrow$ "[CLS] [COL] attr$_1$ [VAL] val$_1$ ... [COL] attr$_k$ [VAL]... [COL] attr$_{k_i}$ [VAL] val$_{k_i}$ ...[SEP]." For list-type attribute values, all elements in the list are merged into a single string, separated by commas. Alternatively, one can skip all the special tokens and directly use S7 *JSON format* or XML-style parentheses structures (Sisaengsuwanchai et al., 2023), which preserves the hierarchical information in some sense. All seven serialization variants are summarized and compared in Table 1.

Serialization for Graph Data. Unlike tabular or semi-structured data, the inherent relationships between entities in a graph make serialization an open challenge. Plain graphs are often serialized as a flat list of nodes and edges. To better align with the text corpora that LMs were pretrained on, graphs often get contextualized through a mapping function (Wang et al., 2023; Fatemi et al., 2023), such as using TV character names and friendships "G describes a friendship graph among Ned, Cat, Daenerys, ... In this friendship graph: Ned and Cat are friends..." For entities with complex relations, Agarwal et al. (2021) formulates it as a data-to-text generation task and uses Seq2seq models to verbalize KG entities. For example, the KG entity "Louvre" in Fig. 1 is described as "Louvre is a museum located in Paris." Madaan et al. (2022); Jiang et al. (2024a) exploit code format to aid LLMs processing graphs by providing a structure abstraction defined in Python-style classes. For instance, the KG entity class is defined as "`class Entity(): def __init__(self,name,id,tuples=[]):...def get_neighbors(self):...`" These methods primarily focus on flattening graphs, while the new serialization strategy we proposed exploits both semantic relationship and structural context between structured entities in KGs through random walks, detailed in Sec. 3.2.

3 Serialization for Structured Entities

In this section, we investigate how different serializations of structured entities affect the utility and efficiency of language models for matching tasks. Building on our findings, we explore new strategies for LLMs to efficiently encode structured entities with complex relations, especially with application to entity alignment in KGs.

3.1 Effects of Entity Serialization on Tabular & Semi-structured Data

To study the impact of entity serialization on LMs, we decompose the problem into *serialization scheme* and *model size*. The serialization scheme determines how structured entities are converted into sequences, which affects the model's ability to interpret the input. Specifically, we benchmark the effect of serialization schemes $\mathcal{S}(\cdot)$ listed in Table 1 for pretrained encoder models and open-source LLMs on EM tasks under zero-shot and fine-tuning settings. We quantify and analyze the impact of attribute order (e.g., fixed, random, or pairwise), special tokens (e.g. COL, VAL, *span typing*), and sequence formats (e.g., plain, JSON) in entity serialization on the model performance.

The model size determines the capacity of a model to represent entities, which affects the utility of learned representations used for matching. Pretrained encoder models are cost-effective for EM due to their compact sizes and low inference cost. Previous work (Pham et al., 2021; Sugawara et al., 2020; Albilali et al., 2021) has shown that input shuffle can lead to significant performance degradation for BERT models in natural language understanding, which may be amplified through serializing structured entities for matching. LLMs

are known to accept various input formats, and they have strong generalizations and capabilities of context understanding in new domains. Preliminary studies (Peeters and Bizer, 2025; Sisaengsuwanchai et al., 2023; Wang et al., 2024) directly pair entities serialized as plain text with prompts to query LLMs and collect their responses in natural language. This QA-based matching is greatly limited by the high latency and inference cost. Instead, we propose to utilize a bi-encoder framework to obtain entity embeddings from the target LLM for matching and study the impact of serialization. For open-source LLMs, such conversion can be done by LLM2Vec (BehnamGhader et al., 2024).

Our empirical study seeks to understand the optimal form of serialization to maximize the matching performance of different LMs on tabular and semi-structured data. We briefly summarize the most exciting results here and defer the detailed analysis to Sec. 4.2:

- With fine-tuning, pretrained encoder models are more sensitive to how entities are serialized than LLMs on structured EM tasks;
- Both encoder models and LLMs have a preference for serialization schemes that are close to the corpus they were pretrained on, especially plain text without special tokens.
- Injecting randomness into attribute orders during serialization can be beneficial for models under supervised fine-tuning, especially on noisy data.

Our benchmark results show that: (1) plain text is sufficient for LMs to capture subtle differences between entities with rich attributes. (2) the predefined attribute order is not optimal for matching, while the injected randomness can improve the model's robustness to input perturbations. These two key observations will play an important role in designing new strategies for serializing structured entities with complex relations.

3.2 Serialization Strategies for Knowledge Graph Entities

KG entities not only contain attributes but are also connected to each other through complex relations. The knowledge stored in the form of fact triples poses unique challenges for applying LM to their matching. Classical methods rely on measuring the similarity of entity embeddings derived from knowledge representation learning (KRL) techniques (Zhang et al., 2022). Recent studies (Yang et al., 2024; Jiang et al., 2024a) have begun to leverage the reasoning power of LLMs to align entities across KGs, but suffer from high latency and inference cost. The fundamental challenge lies in how to effectively encode KG entities and efficiently use LLMs to extract their semantic and structural information for matching.

To tackle this challenge, we apply our findings in Sec. 3.1 to propose a lightweight serialization $\mathcal{S}_w$ for graph-structured entities by sampling semantically meaningful paths on KGs through random walks (see Fig. 1). Specifically, given a target entity $e_i \in \mathcal{E}$, the walk-based serialization:

- Samples M-many L-step walks starting from the root entity e_i on graph $\mathcal{G}$, and obtains a collection of sampled facts $\{W_m\}_{m=1}^{M}$, where $W_m = (w_{m,1}, \ldots, w_{m,L})$ and $w_{m,1} = (e_i, r, e_j)$.
- Applies a function $map(\cdot)$ on each path W_m that maps the entities and their relations from all L sampled facts into a sentence, separated by commas. For example, the path W_m that contains two facts $w_{m,1}$ =("Louvre", "located in", "Paris") and $w_{m,2}$ =("Paris", "is a", "Place") can be processed by $map(W_m)$ = "Louvre located in Paris, Paris is a Place."
- Concatenates sentences of all M paths into a paragraph as $\mathcal{S}_w(e_i; \mathcal{G}) = \cup_{m=1}^{M} map(W_m)$.

If entities contain other attributes, previous serialization schemes can be used to obtain an entity-level description and then call the $map(\cdot)$ function.

This walk-based serialization $\mathcal{S}_w$ naturally encodes the local structures of KG entities in a sequential form, in which relative positions and semantic relationships between entities are embedded. By adjusting the number M of path sampling and the step size L, the output sequence can be controlled according to the model's context window and desired neighborhood coverage, without being affected by irregular sizes of entity-induced subgraphs. We further extend the bi-encoder framework of LLMs to KG entities by pairing it with the proposed strategy $\mathcal{S}_w$, i.e., $\mathbf{h}_e = f_{\text{LLM}}(\mathcal{S}_w(e; \mathcal{G}))$. The proposed Walk-LLM provides a scalable paradigm that can efficiently utilize the extensive parametric knowledge from off-the-shelf LLMs through sampled semantic-rich paths to match KG entities at scale.

Index	Dataset	Format	Domains	$\|D\|$	$\|D'\|$	#Attr k	#Pos	Index	Dataset	Format	#Entities $\|\mathcal{E}\|$	#Relations $\|\mathcal{R}\|$	#Facts $\|\mathcal{F}\|$	#Anchors
D1	Rel-HETER	Hetero. Table	POI	534	332	6; 7	946	D7	DBP15K(EN-FR)	KG	15,000	193; 166	96,318; 80,112	15,000
D2	Semi-Rel	Table; JSON	Movies	29,180	32,823	8; 13.81	2,183	D8	DBP-WIKI	KG	100,000	413; 261	293,990; 251,708	100,000
D3	iTunes-Amazon	Homo. Table (Dirty)	Music	6,907	55,923	8	539	D9	ICEWS-WIKI	KG	11,047; 15,896	272; 226	3,527,881; 198,257	5,058
D4	Walmart-Amazon	Homo. Table (Dirty)	Electronics	2,554	22,074	5	10,242	D10	ICEWS-YAGO	KG	26,863; 22,734	272; 41	4,192,555; 107,118	18,824
D5	IMDb-TVDB	Hetero. Table	TV Shows	5,118	7,810	30; 9	1,072	/						
D6	IMDb-DBpedia	Hetero. Table	Movies	27,615	23,182	4; 7	22,863	/						

Table 2: Dataset Summary & Statistics (D1-6 Tabular and Semi-structured Data; D7-10 Knowledge Graphs).

Scheme	D1 (MRR)			D2 (MRR)			D3 (MRR)			D4 (MRR)			D5 (MRR)			D6 (MRR)		
	RoBERTa	Mistral	Llama	RoBERTa	Mistral	Llama	RoBERTa	Mistral	Llama	RoBERTa	Mistral	Llama	RoBERTa	Mistral	Llama	RoBERTa	Mistral	Llama
S1 Fixed	**1.000**	0.995	**1.000**	0.888	0.826	0.914	0.634	0.787	0.771	0.958	**0.972**	0.962	0.929	0.933	0.929	**0.944**	0.923	0.942
S2 Random	0.991	**1.000**	**1.000**	0.868	0.828	0.824	0.672	0.770	0.753	0.960	0.970	**0.976**	**0.940**	**0.946**	**0.944**	0.929	0.925	0.939
S3 Pairwise	0.995	/	/	**0.906**	/	/	0.672	/	/	0.929	/	/	0.928	/	/	0.931	/	/
S4 Valid	**1.000**	**1.000**	**1.000**	0.854	**0.942**	**0.964**	0.546	0.790	0.789	0.965	0.964	0.968	0.929	0.929	0.931	0.930	0.924	**0.943**
S5 Plain	**1.000**	**1.000**	**1.000**	0.837	0.891	0.926	**0.736**	**0.795**	0.764	**0.973**	0.953	0.970	0.929	0.931	0.930	0.924	0.920	0.942
S6 Span	**1.000**	**1.000**	**1.000**	0.860	0.920	0.913	0.663	0.778	**0.803**	0.965	0.963	0.971	0.610	0.932	0.932	0.934	0.923	0.936
S7 JSON	0.994	0.987	**1.000**	0.884	0.833	0.877	0.667	0.782	0.759	0.970	0.965	0.970	0.920	0.932	0.930	0.936	**0.926**	0.942

Table 3: Language Models (Fine-tuned) for Structured EM with Different Serialization (1st **bold**, 2nd underline).

4 Experiments

4.1 Experiment Settings

Dataset. Six diverse and challenging datasets (Das et al.; Wang et al., 2021; Obraczka et al., 2021; Papadakis et al., 2011) are selected to quantify the impact of serialization schemes on LMs for structured EM. D1 Rel-HETER, D5 IMDb-TVDB, and D6 IMDb-DBPedia are heterogeneous tabular data of entities from restaurants, TV shows, and movies, respectively. D2 Semi-Rel contains book entities with nested attributes in JSON format. We also adopted a dirty version of the D3 iTunes-Amazon and D4 Walmart-Amazon datasets by randomly shifting attribute values to different fields. These two datasets are used to measure the model's robustness against noisy attribute value pairs.

To address the alignment of structured entities in KGs, we pick four evaluation datasets. D7 DBP15K(EN-FR) is a classical dataset for aligning bilingual entity pairs of DBpedia. D8 DBP-WIKI is used for entity alignment across Wikipedia and DBpedia. Both datasets (Sun et al., 2020) have an equal number of entities in KGs with similar structural features, such as the number of facts and density. D9 ICEWS-WIKI and D10 ICEWS-YAGO (Jiang et al., 2024b) are two new datasets of highly heterogeneous KGs (HHKG) with different numbers of entities and distinct structures.

Table 2 summarizes the statistics of selected datasets for evaluation. The attribute names of datasets D1-D6 are listed in Table 7, Appx. B.1. The license information and source for all datasets can be found in Table 8, Appx. B.1.

Backbone Models. BERT (Devlin et al., 2019) and RoBERTa (Liu, 2019) are the most commonly used encoder models on EM tasks, which have shown high utility in practice. Two popular LLMs of Llama3-8B (AI@Meta, 2024) and Mistral-7B (Jiang et al., 2023) are also evaluated. We utilize LLM2Vec (BehnamGhader et al., 2024) to transform these open-source LLMs into text encoders, where no additional prompts are required other than serialized entities. To our knowledge, this is the first exploration of using Llama3-8B and Mistral-7B as text encoders for EM.

Baselines. Three types of methods are selected for matching KG entities, which cover different input features of KGs and KRL techniques: translation-based method MTransE (Chen et al., 2017), GNN-based methods of RDGCN (Wu et al., 2019) and Dual-AMN (Mao et al., 2021), LM-based methods of BERT (Devlin et al., 2019), BERT-INT (Tang et al., 2020), and the previous SoTA methods of Simple-HHEA (Jiang et al., 2024b) and two-stage GPT-4-based ChatEA (Jiang et al., 2024a).

Metrics. In line with widely adopted evaluation methods for EM tasks, we use two ranking metrics: Hits@K, measuring the percentage of correct predicted entity pairs among the top-K matches, and mean reciprocal rank (MRR), calculating the average inverse ranking of correct predicted match pairs. For datasets D1-6, we use the BM25 algorithm (Robertson et al., 2009) to pair each matched pair with $K = 10$ hard non-match entities to obtain the ranking. For datasets D7-10, we follow the same evaluation setting as in Jiang et al. (2024a).

Implementation Details. For encoder models, we incorporate their base models with the Ditto framework (Li et al., 2020) as cross-encoder for matching, which is one of the first dedicated EM systems using pretrained LMs (see Fig. 4 in Appx.

Models	DBP15K(EN-FR)			DBP-WIKI			ICEWS-WIKI			ICEWS-YAGO		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
MTransE †	0.247	0.577	0.360	0.281	0.520	0.363	0.021	0.158	0.068	0.012	0.084	0.040
RDGCN†	0.873	0.950	0.901	0.974	0.994	0.980	0.064	0.202	0.096	0.029	0.097	0.042
Dual-AMN†	0.954	0.994	0.970	0.983	0.996	0.991	0.083	0.281	0.145	0.031	0.144	0.068
BERT	0.811	0.859	0.829	0.800	0.852	0.818	0.609	0.739	0.655	0.794	0.871	0.822
BERT-INT †	**0.990**	0.997	0.993	**0.996**	0.997	0.996	0.561	0.700	0.607	0.756	0.859	0.793
Simple-HHEA†	0.959	0.995	0.972	0.975	0.991	0.988	0.720	0.872	0.754	0.847	0.915	0.870
ChatEA (GPT-4)†	**0.990**	**1.000**	**0.995**	0.995	**1.000**	**0.998**	0.880	0.945	0.912	0.935	0.955	0.944
Walk-Mistral-7B$_{M=10}$	0.980	0.999	0.988	0.980	0.998	0.988	0.986	0.999	0.992	**0.980**	0.994	**0.986**
Walk-Llama3-8B$_{M=10}$	0.983	0.999	0.990	0.974	0.997	0.984	**0.988**	**1.000**	**0.993**	0.974	**0.995**	0.983

Table 4: Result Comparison on Matching Entities from Canonical KGs and HHKGs (1st **bold**, 2nd underline).

A.1). The probability of a pair match predicted by the model is used for ranking. For open-source LLMs, we use them as bi-encoders, which are fine-tuned by contrastive-based objectives (see Fig. 6 in Appx. A.2). The scheme S3 *pairwise order* is incompatible with bi-encoders and thus skipped. The source code is available at https://github.com/amazon-science/serializeEM.

4.2 Experiments on Tabular & Semi-structured Entities

Table 3 shows how serialization schemes shape language models of different sizes and backbones on datasets D1-6 with fine-tuning. The results for zero-shot are reported in Table 9, Appx. B.3.

RQ1-2 Randomly shuffling attribute orders does not necessarily hurt the matching performance; in fact, the randomness injected by S2 *Random order* can improve the robustness of RoBERTa, especially on dirty datasets D3-4. Instead, S1 *Fixed order*, the common by default choice, does not show an advantage over other schemes, especially falling far behind on D3-4. This result suggests that breaking the permutation invariance of attributes can negatively affect the model, especially when data quality is not ideal. S3 *Pairwise order* shows its strength in matching semi-structured entities on D2, suggesting that aligning common attributes between entity pairs can potentially enhance the signal and mitigate the side effects of flattening nested attributes. In addition, removing attributes with missing values through S4 *Valid value* is generally not beneficial to pretrained encoder models.

As for LLMs, they perform reasonably well on the noisy datasets D3-4 for zero-shot. Randomly permuting the order of attributes without supervising signals usually hurts the model, but the performance drop is not substantial. These two observations show the overall robustness of LLMs to input perturbations. After fine-tuning, the results of LLMs with all types of serialization are greatly improved, with S2 and S4 achieving the largest gains and the best performance in general. This observation suggests that structural properties of entity attributes and data quality play a crucial role in fine-tuning LLMs for matching.

RQ3 S5 *Plain format* leads four of six datasets, and comparable results can be observed from S7 *JSON format*. This indicates that using special tokens to retain the structure of the input by S1,2,4 or inject domain knowledge by S6 *Span typing* is not helpful in improving encoder models and potentially causes more overhead. For LLMs, the model without fine-tuning shows a strong preference for S5 and S7, which do not contain special tokens and are mostly close to the text corpus on which the model was pretrained. Surprisingly, with fine-tuning, LLMs were able to capture the domain knowledge injected by S6 on D3. However, it is not as effective on other datasets, which are similarly observed on pretrained encoder models.

4.3 Experiments on KG Entities

Table 4 compares different baseline models on matching entities in canonical and highly heterogeneous KGs. Results marked with † are from Jiang et al. (2024a). Pairing walk-based serialization with open-source LLMs achieves performance comparable to BERT-INT and GPT-4-based ChatEA on canonical KG datasets. On two HHKG datasets, our proposed method Walk-LLM obtains MRR scores of 0.993 (+8.88%) and 0.986 (+4.45%), respectively, which is a significant improvement over ChatEA. These results highlight the uniform effectiveness and consistent superiority of the proposed Walk-LLM on various types of KG datasets. It also shows that the power of LLMs can be effectively exploited through entity embeddings in combination with rich contextual input produced by the walk-based strategy but does not suffer from high interactive inference costs like ChatEA.

Models	ICEWS-WIKI (non-trivial)			ICEWS-YAGO (non-trivial)			ICEWS-WIKI (dirty)			ICEWS-YAGO (dirty)		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
BERT	0.359	0.587	0.438 (↓33.1%)	0.566	0.726	0.624 (↓24.1%)	0.211	0.235	0.220	0.225	0.250	0.234
Simple-HHEA	0.490	0.777	0.579 (↓23.2%)	0.614	0.788	0.675 (↓22.4%)	0.298	0.522	0.374	0.309	0.497	0.373
Llama3-8B (MNTP)	0.653	0.895	0.737	0.571	0.790	0.647	0.226	0.358	0.296	0.234	0.297	0.258
Mistral-7B (MNTP)	0.821	0.948	0.868	0.822	0.924	0.860	0.385	0.553	0.444	0.339	0.467	0.384
Walk-Mistral-7B$_{M=10}$	**0.981**	0.998	**0.989** (↓0.3%)	**0.942**	**0.985**	**0.959** (↓2.7%)	**0.834**	**0.975**	**0.888**	**0.786**	**0.944**	**0.844**
Walk-Llama3-8B$_{M=10}$	0.980	**1.000**	**0.989** (↓0.4%)	0.937	0.977	0.954 (↓3.0%)	0.745	0.907	0.804	0.673	0.840	0.732

Table 5: Result Comparison on Matching Non-trivial and Dirty Entities from HHKGs (1st **bold**, 2nd underline).

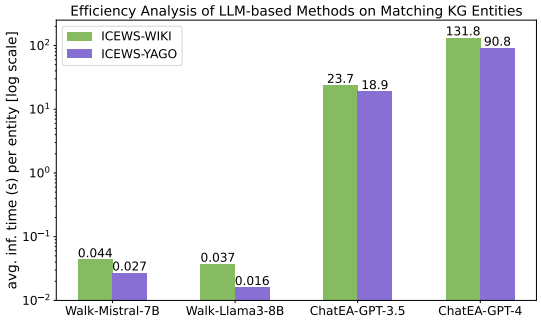

Figure 2: Per-query Runtime Comparison of LLM-based Methods on Matching Entities across HHKGs.

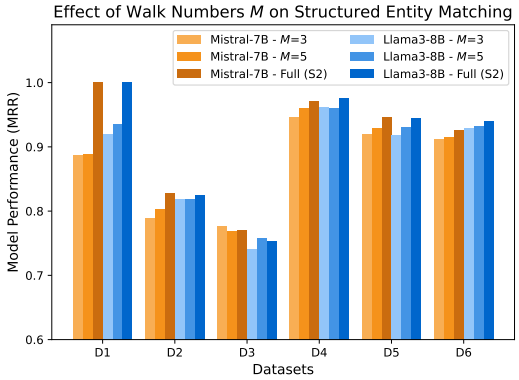

Figure 3: Effect of Walk Numbers M on LLMs with Fine-tuning for Matching Tabular and Semi-structured Entities. M represents the number of entity attributes sampled in the serialized input, and increasing M eventually leads to the input sequence equivalent to S2 *Random order* without downsampling.

4.3.1 Ablation Study

Hard Examples. Table 5 (Left, non-trivial) reports results on entity pairs with non-identical names in the source and target KGs (23% overall) from the HHKG datasets. Here, we focus on LM-based methods because they rely primarily on name attributes to differentiate entities. BERT and Simple-HHEA suffer more than 20% performance degradation on these non-trivial entity pairs, while the proposed Walk-LLM barely drops. We provide the results by comparing embeddings of entity names directly generated from LLMs for reference. These models are trained through LLM2Vec with the objective of masked next token prediction (MNTP), where the structure in KG is excluded from the input. The model performance gap between MNTP and Walk-LLM further emphasizes that the proposed walk-based strategy introduces a strong ability to distinguish hard examples.

Model Robustness. Table 5 (Right, dirty) compares different LM-based methods on the noisy version of HHKG datasets, where either half of the "name" attribute is masked or gets mixed with its URL uniformly at random. This simulates one kind of dirty data commonly seen in practice while keeping the modification simple. Both BERT and Simple-HHEA suffer severe performance degradation due to their over-reliance on entity names, while Walk-LLM is robust to such perturbations. Semantic paths sampled by random walks can mitigate the impact of input noise on model utility. A direct comparison with ChatEA cannot be done due to its limited accessibility. Based on results over noisy embeddings (see Fig. 3 of Jiang et al. (2024a)), both Walk-LLM and ChatEA achieve better robustness than Simple-HHEA. Note that ChatEA has 2 stages, where the first stage relies on embeddings generated by Simple-HHEA (or other encoders) and thus is still sensitive to input noise.

4.3.2 Efficiency Analysis

Fig. 2 compares the averaged inference time per entity between LLM-based methods to match KG entities. Note that, the runtime results of GPT-based methods are obtained from Table 5 of ChatEA (Jiang et al., 2024a). Walk-LLM ($M = 10$) achieves a speedup of more than 538 $\times$ and 2995 $\times$ than ChatEA with the GPT-3.5 and GPT-4 backbones, respectively. The reduction in inference cost comes from two main aspects: (1) Replacing chat-based queries with a bi-encoder framework that matches entities based on comparing their embeddings generated from context-rich features sampled by random walks. (2) Running open-source LLMs with fewer parameters locally is cheaper than the API calls of online GPT models.

Models	ICEWS-WIKI			ICEWS-YAGO		
	Hits@1	Hits@10	MRR	Hits@1	Hits@10	MRR
Walk-Mistral-7B$_{M=5}$	0.977	0.999	0.987	0.969	0.994	0.980
Walk-Mistral-7B$_{M=10}$	0.986	0.999	0.992	0.980	0.994	0.986
Walk-Mistral-7B$_{\text{Subg}}$	0.982	1.000	0.990	0.974	0.995	0.983
Walk-Llama3-8B$_{M=5}$	0.983	1.000	0.991	0.972	0.994	0.981
Walk-Llama3-8B$_{M=10}$	0.988	1.000	0.993	0.974	0.995	0.983
Walk-Llama3-8B$_{\text{Subg}}$	0.987	1.000	0.993	0.973	0.994	0.982

Table 6: Effect of Walk Numbers M on LLMs with Fine-tuning for Matching Entities from HHKGs. subg indicates the use of all neighboring entities in KGs as the input without downsampling.

4.4 Hyperparameter Study

We start with the effect of the number of walks on model performance using walk-based serialization over tabular and semi-structured datasets. Fig. 3 shows that the walk-based serialization of sampling M entity attributes ($M = 3, 5$) provides reasonable results compared to S2 *Random order* without downsampling. The number of entity attributes in D1-6 is mostly between 4 and 9. The performance of the walk-based strategy improves with the increase of M and eventually converges to the results of S2. For the KG dataset, Table 6 summarizes the comparison of model performance when sampling different numbers of paths M for each entity. Similarly, increasing M can improve the matching performance of the model, and it is sufficient to set the step L to 1. However, when all neighboring entities in the induced subgraph are used as input without downsampling, the performance is not substantially boosted. Meanwhile, using the full subgraph increases the training and inference time by 4 times compared to $M = 10$. Considering the trade-off between complexity and utility, the choice of M should generally follow the density of the input graph.

5 Conclusion

In this work, we systematically study the effect of entity serialization on language models with different sizes and backbones for matching structured entities. We empirically find that BERT-based encoder models are more sensitive to serialization than LLMs, especially in terms of attribute order and data quality. Both types of language models prefer serialization schemes that are close to the text corpus on which they are pretrained. When entity attributes are noisy, injecting randomness into the input can benefit the model's performance. By applying these findings, we propose a random walk-based serialization for open-source LLMs to scalably generate robust embeddings for matching KG entities. The proposed LLM-Walk achieves SoTA performance and is three orders of magnitude faster than GPT-4-based methods.

6 Limitations

While our study provides a comprehensive analysis of serialization strategies for entity matching, several limitations remain. First, the scope of the benchmark datasets is mainly tabular and knowledge graph entities, which may not generalize to other types of structured data, such as complex nested or temporal entities. Second, although our proposed random walk-based serialization method demonstrates strong performance in knowledge graphs, it may struggle with entities that lack well-defined or sufficient graph structure, potentially impacting the method's efficacy in domains with sparse or incomplete relations. Third, our exploration in this work is limited to open-source LLMs like Llama and Mistral. We have not evaluated our methods over proprietary or closed-source models like GPT-4o (and inter alia). Lastly, the implications of entity serialization on downstream tasks that leverage structured data and LLMs, e.g., in question answering with knowledge graphs, are not explored in this work.

Overall, these limitations suggest that while the findings offer valuable insights, further work is needed to fully understand the generalizability and practical applicability of the studied serialization approaches in real-world scenarios.

7 Ethics Statement

This work focuses on studying how the serialization of structured entities affects language models on matching tasks and is conducted under the guidelines of the ACL Ethics Policy. The datasets used for experiments are publicly available, widely recognized by the research community, and, to the best of our knowledge, contain no personally identifiable information or content that is harmful, offensive, or biased.

References

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Lan-*

guage Technologies, pages 3554–3565, Online. Association for Computational Linguistics.

AI@Meta. 2024. Llama 3 model card. https://www.llama.com/.

Mehdi Akbarian Rastaghi, Ehsan Kamalloo, and Davood Rafiei. 2022. Probing the robustness of pre-trained language models for entity matching. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, pages 3786–3790.

Eman Albilali, Nora Altwairesh, and Manar Hosny. 2021. What does BERT learn from Arabic machine reading comprehension datasets? In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 32–41, Kyiv, Ukraine (Virtual). Association for Computational Linguistics.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. LLM2Vec: Large language models are secretly powerful text encoders. In *First Conference on Language Modeling*.

Misha Bilenko. 2019. Duplicate detection, record linkage, and identity uncertainty: Datasets. https://www.cs.utexas.edu/~ml/riddle/data.html.

Ursin Brunner and Kurt Stockinger. 2020. Entity matching with transformer architectures-a step forward in data integration. In *Proceedings of the 23rd International Conference on Extending Database Technology*, pages 463–473. OpenProceedings.

Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. 2017. Multilingual knowledge graph embeddings for cross-lingual knowledge alignment. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1511–1517.

Peter Christen. 2012. *The Data Matching Process*, pages 23–35. Springer Berlin Heidelberg, Berlin, Heidelberg.

Sanjib Das, AnHai Doan, Paul Suganthan G. C., Chaitanya Gokhale, Pradap Konda, Yash Govind, and Derek Paulsen. The magellan data repository. https://sites.google.com/site/anhaidgroup/projects/data.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Dennis Diefenbach, Vanessa Lopez, Kamal Singh, and Pierre Maret. 2018. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information Systems*, 55:529–569.

Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. Distributed representations of tuples for entity resolution. *Proc. VLDB Endow.*, 11(11):1454–1467.

Meihao Fan, Xiaoyue Han, Ju Fan, Chengliang Chai, Nan Tang, Guoliang Li, and Xiaoyong Du. 2024. Cost-effective in-context learning for entity resolution: A design space exploration. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 3696–3709. IEEE.

Wenfei Fan, Xibei Jia, Jianzhong Li, and Shuai Ma. 2009. Reasoning about record matching rules. *Proc. VLDB Endow.*, 2(1):407–418.

Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. 2023. Talk like a graph: Encoding graphs for large language models. *arXiv preprint arXiv:2310.04560*.

Johannes Frey, Lars-Peter Meyer, Natanael Arndt, Felix Brei, and Kirill Bulert. 2023. Benchmarking the abilities of large language models for rdf knowledge graph creation and comprehension: How well do llms speak turtle? *arXiv preprint arXiv:2309.17122*.

Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. 2020. Message passing for hyper-relational knowledge graphs. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7346–7359, Online. Association for Computational Linguistics.

Lise Getoor and Ashwin Machanavajjhala. 2012. Entity resolution: theory, practice & open challenges. *Proc. VLDB Endow.*, 5(12):2018–2019.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR.

Stefan Hegselmann, Alejandro Buendia, Hunter Lang, Monica Agrawal, Xiaoyi Jiang, and David Sontag. 2023. Tabllm: Few-shot classification of tabular data with large language models. In *Proceedings of the 26th International Conference on Artificial Intelligence and Statistics*, pages 5549–5581. PMLR.

Thomas N. Herzog, Fritz J. Scheuren, and William E. Winkler. 2007. *Data Quality and Record Linkage Techniques*, 1st edition. Springer Publishing Company, Incorporated.

Qianyu Huang and Tongfang Zhao. 2024. Leveraging large language models for entity matching. *arXiv preprint arXiv:2405.20624*.

AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, D de las Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. 2023. Mistral 7b (2023). *arXiv preprint arXiv:2310.06825*.

Xuhui Jiang, Yinghan Shen, Zhichao Shi, Chengjin Xu, Wei Li, Zixuan Li, Jian Guo, Huawei Shen, and Yuanzhuo Wang. 2024a. Unlocking the power of large language models for entity alignment. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7566–7583, Bangkok, Thailand. Association for Computational Linguistics.

Xuhui Jiang, Chengjin Xu, Yinghan Shen, Yuanzhuo Wang, Fenglong Su, Zhichao Shi, Fei Sun, Zixuan Li, Jian Guo, and Huawei Shen. 2024b. Toward practical entity alignment method design: Insights from new highly heterogeneous knowledge graph datasets. In *Proceedings of the ACM on Web Conference 2024*, pages 2325–2336.

Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeff Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: toward building entity matching management systems. *Proc. VLDB Endow.*, 9(12):1197–1208.

Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proc. VLDB Endow.*, 3(1–2):484–493.

Huahang Li, Longyu Feng, Shuangyin Li, Fei Hao, Chen Jason Zhang, Yuanfeng Song, and Lei Chen. 2024. On leveraging large language models for enhancing entity resolution. *arXiv preprint arXiv:2401.03426*.

Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep entity matching with pre-trained language models. *Proc. VLDB Endow.*, 14(1):50–60.

Yinhan Liu. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. 2022. Language models of code are few-shot commonsense learners. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1384–1403, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Christopher Manning. 2017. Representations for language: From word embeddings to sentence meanings. *Simons Institute for the Theory of Computing, UC Berkeley*.

Xin Mao, Wenting Wang, Yuanbin Wu, and Man Lan. 2021. Boosting the speed of entity alignment 10×: Dual attention matching network with normalized hard sample mining. In *Proceedings of the Web Conference 2021*, pages 821–832.

Zhengjie Miao, Yuliang Li, and Xiaolan Wang. 2021. 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*, SIGMOD '21, page 1303–1316, New York, NY, USA. Association for Computing Machinery.

Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep learning for entity matching: A design space exploration. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, page 19–34, New York, NY, USA. Association for Computing Machinery.

Ali Naeim abadi, Mir Tafseer Nayeem, and Davood Rafiei. 2023. Product entity matching via tabular data. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 4215–4219.

Daniel Obraczka, Jonathan Schuchart, and Erhard Rahm. 2021. Eager: embedding-assisted entity resolution for knowledge graphs. *arXiv preprint arXiv:2101.06126*.

Matteo Paganelli, Francesco Del Buono, Andrea Baraldi, and Francesco Guerra. 2022. Analyzing how bert performs entity matching. *Proc. VLDB Endow.*, 15(8):1726–1738.

George Papadakis, Ekaterini Ioannou, Claudia Niederée, and Peter Fankhauser. 2011. Efficient entity resolution for large heterogeneous information spaces. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*, pages 535–544.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, Austin, Texas. Association for Computational Linguistics.

Ralph Peeters and Christian Bizer. 2021. Dual-objective fine-tuning of bert for entity matching. *Proc. VLDB Endow.*, 14(10):1913–1921.

Ralph Peeters and Christian Bizer. 2025. Entity matching using large language models. In *Proceedings of the 28th International Conference on Extending Database Technology*, pages 529–541. OpenProceedings.

Thang Pham, Trung Bui, Long Mai, and Anh Nguyen. 2021. Out of order: How important is the sequential order of words in a sentence in natural language understanding tasks? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1145–1160, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Rohit Singh, Vamsi Meduri, Ahmed Elmagarmid, Samuel Madden, Paolo Papotti, Jorge-Arnulfo Quiané-Ruiz, Armando Solar-Lezama, and Nan Tang. 2017. Generating concise entity matching rules. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, page 1635–1638, New York, NY, USA. Association for Computing Machinery.

Khanin Sisaengsuwanchai, Navapat Nananukul, and Mayank Kejriwal. 2023. How does prompt engineering affect chatgpt performance on unsupervised entity resolution? *arXiv preprint arXiv:2310.06174*.

Saku Sugawara, Pontus Stenetorp, Kentaro Inui, and Akiko Aizawa. 2020. Assessing the benchmarking capacity of machine reading comprehension datasets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8918–8927.

Zequn Sun, Qingheng Zhang, Wei Hu, Chengming Wang, Muhao Chen, Farahnaz Akrami, and Chengkai Li. 2020. A benchmarking study of embedding-based entity alignment for knowledge graphs. *Proc. VLDB Endow.*, 13(12):2326–2340.

Xiaobin Tang, Jing Zhang, Bo Chen, Yang Yang, Hong Chen, and Cuiping Li. 2020. Bert-int:a bert-based interaction model for knowledge graph alignment. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3174–3180. International Joint Conferences on Artificial Intelligence Organization.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Somin Wadhwa, Adit Krishnan, Runhui Wang, Byron C Wallace, and Luyang Kong. 2024. Learning from natural language explanations for generalizable entity matching. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6114–6129, Miami, Florida, USA. Association for Computational Linguistics.

Heng Wang, Shangbin Feng, Tianxing He, Zhaoxuan Tan, Xiaochuang Han, and Yulia Tsvetkov. 2023. Can language models solve graph problems in natural language? In *Advances in Neural Information Processing Systems*, volume 36.

Jiapu Wang, Kai Sun, Linhao Luo, Wei Wei, Yongli Hu, Alan Wee-Chung Liew, Shirui Pan, and Baocai Yin. 2024. Large language models-guided dynamic adaptation for temporal knowledge graph reasoning. In *Advances in Neural Information Processing Systems*, volume 37.

Jin Wang, Yuliang Li, and Wataru Hirota. 2021. Machamp: A generalized entity matching benchmark. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, pages 4633–4642.

Pengfei Wang, Xiaocan Zeng, Lu Chen, Fan Ye, Yuren Mao, Junhao Zhu, and Yunjun Gao. 2022. Promptem: prompt-tuning for low-resource generalized entity matching. *Proc. VLDB Endow.*, 16(2):369–378.

Tianshu Wang, Xiaoyang Chen, Hongyu Lin, Xuanang Chen, Xianpei Han, Le Sun, Hao Wang, and Zhenyu Zeng. 2025. Match, compare, or select? an investigation of large language models for entity matching. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 96–109, Abu Dhabi, UAE. Association for Computational Linguistics.

Cameron R Wolfe. 2024. Decoder-only transformers: The workhorse of generative llms. `https://cameronrwolfe.substack.com/p/decoder-only-transformers-the-workhorse`.

Yuting Wu, Xiao Liu, Yansong Feng, Zheng Wang, Rui Yan, and Dongyan Zhao. 2019. Relation-aware entity alignment for heterogeneous knowledge graphs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5278–5284. International Joint Conferences on Artificial Intelligence Organization.

Linyao Yang, Hongyang Chen, Xiao Wang, Jing Yang, Fei-Yue Wang, and Han Liu. 2024. Two heads are better than one: Integrating knowledge from knowledge graphs and large language models for entity alignment. *arXiv preprint arXiv:2401.16960*.

Alexandros Zeakis, George Papadakis, Dimitrios Skoutas, and Manolis Koubarakis. 2023. Pre-trained embeddings for entity resolution: An experimental analysis. *Proc. VLDB Endow.*, 16(9):2225–2238.

Rui Zhang, Bayu Distiawan Trisedya, Miao Li, Yong Jiang, and Jianzhong Qi. 2022. A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning. *The VLDB Journal*, 31(5):1143–1168.

A Architecture of Entity Matching Systems Using Language Models

A.1 Pretrained Encoder Models

Entity matching requires rich contextualized representations, where pretrained encoder models such as BERT can produce entity embeddings with high throughput and low latency but have limited ability of reasoning and generalization. The most common entity matching (EM) system using pretrained language models (LMs) is based on the cross-encoder framework [1] proposed by Li et al. (2020), where the model takes serialized entities in pairs as input and formulates their matching as a classification task, as shown in Fig. 4.

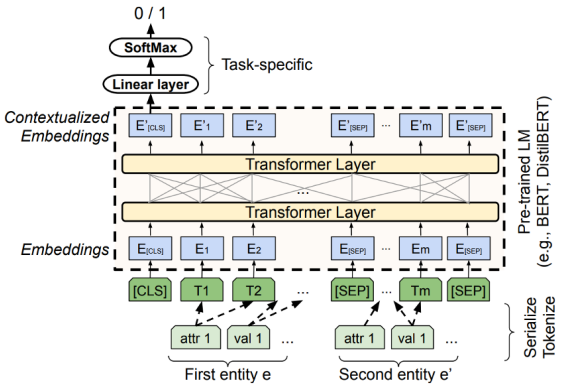

Figure 4: Architecture of Ditto (Li et al., 2020).

A.2 Large Language Models (Decoder-only)

Compared to encoder models, LLMs with billions of parameters contain more knowledge from pertaining, but their decoding part is very expensive and time-consuming. LLM2vec [2] (BehnamGhader et al., 2024) can transform any pretrained decoder-only LLM into a (universal) text encoder, where the model can be used in the bi-encoder framework for matching that breaks the bottleneck of text generation in chat-based matching (See Figs. 5, 6).

B More Experimental Details

B.1 Benchmark Datasets

The following provides a detailed description of the 10 datasets selected for entity matching and alignment tasks. Table 7 lists the attribute names for tabular and semi-structured datasets.

- D1 Rel-HETER: This task is for matching between structured tables with heterogeneous schema. The source is the Fodors-Zagats task from the Deep Matcher datasets (Bilenko, 2019).

- D2 Semi-Rel: This task is for matching between semi-structured and structured tables. The source is the set of 5 Movie tasks collected by the Magellan project (Das et al.).

- D3 iTunes-Amazon (Dirty) contains music data from iTunes and Amazon. This was created by students in the CS 784 data science class at UW-Madison. The dirty version was obtained by modifying the structured iTunes-Amazon dataset to simulate dirty data. Specifically, for each attribute other than "title", they randomly moved each value to the attribute "title" in the same tuple with 50% probability.

- D4 Walmart-Amazon (Dirty) contains product data from Walmart and Amazon. The procedure for generating the dirty version of this dataset is the same as that for D3.

- D5 IMDb-TVDB consists of two individual data sources, which comprise movie descriptions from imdb.com (IMDb) and TV shows from TheTVDB.com (TVDB) (Obraczka et al., 2021).

- D6 IMDb-DBpedia matches movies from IMDb and DBpedia (Papadakis et al., 2011), but has no overlap with the IMDb data source of D5.

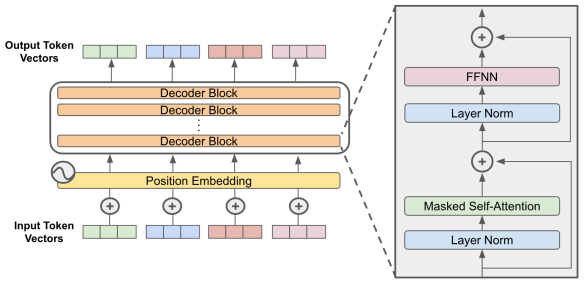

Figure 5: Architecture of Decoder-only Transformer Model (Vaswani et al., 2017). Image source (Wolfe, 2024).

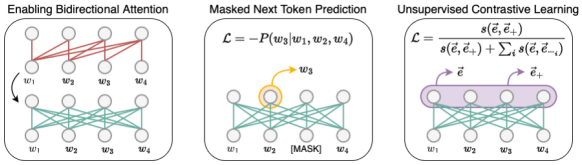

Figure 6: 3 Steps of LLM2vec to convert decoder-only models as universal text embedding models: enabling bidirectional attention, fine-tuning with masked next token prediction and unsupervised contrastive learning (BehnamGhader et al., 2024).

[1] https://github.com/megagonlabs/ditto
[2] https://github.com/McGill-NLP/llm2vec

Dataset	D attribute names	D' attribute names
D1 Rel-HETER	name, address, phone, category	addr, city, phone, type, class
D2 Semi-Rel	id, title, authors, venue, year	Movie1: id, name, year_range, relase_date, director, creator, cast, duration, rating:[rating_value, content_rating], genre, url, description; Movie2: id, name, year, director, writers(list), actor(list); Movie3: id, title, year, director, creators, cast, genre, duration, rating: [rating, content_rating], summary; Movie4: id, title, time, director, year, stars(list), rating:[rotten_tomatoes, audience_rating, review(list)]; Movie5: id, movide_name, year, directors, actors, movide_rating, genre, duration.
D3 iTunes-Amazon	id, Song_Name, Artist_Name, Album_Name, Genre, Price, CopyRight Time, Released	same as D
D4 Walmart-Amazon	id, title, category, brand, modelno, price	same as D
D5 IMDb-TVDB	title, name, episodeNumber, seasonNumber, deathYear, birthYear, endYear, startYear, genre_list, primaryProfessions, runtimeMinutes	title, name, abstract, episodeNumber, seasonNumber, releaseDate, job
D6 IMDb-DBpedia	id, title, starring, writer, editor, aggregate value	id, title, actor name, director name, year, genre, aggregate value

Table 7: The schema (attribute names) of tabular and semi-structured data.

Backbone Model	#param	License	Model Card
BERT.base	110M	Apache License 2.0	https://huggingface.co/google-bert/bert-base-uncased
RoBERTa.base	125M	MIT License	https://huggingface.co/FacebookAI/roberta-base
LLM2Vec-Mistral-7B-Instruct-v2-mntp	7B	MIT License	https://huggingface.co/McGill-NLP/LLM2Vec-Mistral-7B-Instruct-v2-mntp
LLM2Vec-Meta-Llama-3-8B-Instruct-mntp	8B	MIT License	https://huggingface.co/McGill-NLP/LLM2Vec-Meta-Llama-3-8B-Instruct-mntp
Dataset	Type	License	Data Source
D1 Rel-HETER, D2 Semi-Rel	Clean, Hetero.	BSD-3-Clause	https://github.com/megagonlabs/machamp
D3 iTunes-Amazon, D4 Walmart-Amazon	Dirty, Homo.	GPL-3.0 license	https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md
D5 IMDb-TVDB, D6 IMDb-DBpedia	Clean, Homo.	CC BY 4.0	https://zenodo.org/records/6950980
D7 DBP15K(EN-FR), D8 DBP-WIKI	KG, Canonical	GPL-3.0 license	https://github.com/nju-websoft/OpenEA
D9 ICEWS-WIKI, D10 ICEWS-YAGO	KG, Highly Hetero.	CC BY 4.0	https://github.com/IDEA-FinAI/Simple-HHEA/tree/main/data

Table 8: Data license and model card of pretrained language models.

- D7 DBP15K(EN-FR) and D8 DBP-WIKI are two canonical entity alignment datasets. DBP15K(EN-FR) is for bilingual entity alignment on DBpedia, and DBP-WIKI is for aligning entities obtained from Wikipedia and DBpedia.

- D9 ICEWS-WIKI and D10 ICEWS-YAGO come from Highly Heterogeneous Knowledge Graph Datasets (Jiang et al., 2024b). Both datasets are for KG entity alignment, integrating the event knowledge graph derived from the Integrated Crisis Early Warning System (ICEWS) and general KGs (i.e., WIKIDATA, YAGO).

B.2 Experimental Settings

To benchmark the effect of entity serialization schemes on pretrained encoder models, we use the default setting of the Ditto framework (Li et al., 2020), which only modifies how entity serialization is performed, to fine-tune the encoder model for each scheme $\mathcal{S}$ listed in Table 1 on datasets D1-6, and report the results in Table 3. For LLMs, we first transform the base model (listed in Table 8) using LLM2vec and then obtain the entity embeddings directly. For matching tasks, the default instruction used by LLM2Vec is replaced by "Given a description of a real-world entity, retrieve the most relevant entities that match or align with the given entity." The matching score for each entity tuple (one positive versus k negative pairs) in datasets D1-6 is computed under each serialization scheme, and the zero-shot performance is summarized in Table 9. We further fine-tune each model on the train set of all six datasets at once for each serialization and report their performance in Table 3.

For all baselines compared for matching KG entities, we followed the original hyperparameter settings reported in their papers with the same 3:7 splitting ratio in the training/testing set. All baselines use the same preprocessing procedure to obtain the initial features from four KG datasets. We fine-tuned the base model of open-source LLMs under supervised fine-tuning through LLM2vec with recommended parameters and entity pairs serialized by the walk-based strategy ($M = 10, L = 1$) as input. Datasets D7-8 and D9-10 have common domains with DBpedia and ICEWS, respectively. Thus, we fine-tune one model for datasets with a sharing domain. The reported results are the average of 5 runs.

The code base is developed based on `Pytorch 2.3.1` with `Transformer 4.40.2`, and `llm2vec 0.2.2`. The experiments are performed on a server of Ubuntu 24.04LTS OS with 3.0 GHz 2nd Gen Intel Xeon processors (96 cores) and 8 NVIDIA A100 Tensor Core GPUs (40G).

B.3 More Experimental Results

Zero-shot EM with LLMs To study how attribute order, special tokens, or missing/dirty attributes affect LLMs for entity matching under the zero-shot setting, we compare the entity embeddings generated by off-the-shelf LLMs in one run

Scheme	D1 (MRR)		D2 (MRR)		D3 (MRR)		D4 (MRR)		D5 (MRR)		D6 (MRR)	
	Mistral	Llama	Mistral	Llama	Mistral	Llama	Mistral	Llama	Mistral	Llama	Mistral	Llama
S1 Fixed	0.903	0.901	0.843	0.813	**0.739**	0.612	0.840	0.780	0.713	0.544	0.695	0.685
S2 Random	0.883	0.892	0.839	0.812	0.696	0.626	0.834	0.769	0.707	0.499	0.670	0.663
S4 Valid	0.903	0.901	0.852	0.830	0.703	0.548	0.815	0.743	0.718	0.579	**0.712**	0.719
S5 Plain	0.917	**0.964**	0.836	0.839	0.698	**0.636**	**0.856**	**0.827**	0.702	**0.595**	0.702	0.733
S7 JSON	**0.939**	0.929	**0.881**	**0.869**	0.708	0.539	0.839	0.774	**0.721**	0.583	0.709	**0.745**

Table 9: LLMs for Zero-shot Structured EM with Different Serialization (1st **bold**, 2nd underline)

under the bi-encoder framework and report the results in Table 9. S3 *Pairwise order* is skipped as the bi-encoder framework encodes the serialized entities separately. S6 *Span typing* is designed to be paired with supervised training, which is also skipped under the zero-shot setting.

C Use of Generative AI

In this work, generative AI services are used to study several GPT-based baselines, test corner cases in entity matching, and polish writing at the sentence level.