Front. Comput. Sci., 2024, 0(0): 1–29
https://doi.org/10.1007/sxxxxx-yyy-zzzz-1

RESEARCH ARTICLE

Towards Uncertainty-Calibrated Structural Data Enrichment with Large Language Model for Few-Shot Entity Resolution

Mengyi Yan[1], Yaoshu Wang[2], Xiaohan Jiang[1], Haoyi Zhou[3], Jianxin Li(✉)[1]

1 School of Computer Science and Engineering, Beihang University, Beijing 100191, China
2 Shenzhen Institute of Computing Sciences, Shenzhen 518110, Guangdong, China
3 School of Software, Beihang University, Beijing 100191, China

Abstract

Entity Resolution (ER) is vital for data integration and knowledge graph construction. Despite the advancements made by deep learning (DL) methods using pre-trained language models (PLMs), these approaches often struggle with unstructured, long-text entities (ULE) in real-world scenarios, where critical information is scattered across the text, and existing DL methods require extensive human labeling and computational resources To tackle these issues, we propose a Few-shot Uncertainty-calibrated Structural data Enrichment method for ER (FUSER). FUSER applies unsupervised pairwise enrichment to extract structural attributes from unstructured entities via LLMs, and integrates an uncertainty-based calibration module to reduce hallucination issues with minimal additional inference cost. It also implements a lightweight ER pipeline that efficiently performs both blocking and matching tasks with as few as 50 labeled positive samples. FUSER was evaluated on six ER benchmark datasets featuring ULE entities, outperforming state-of-the-art methods and significantly boosting the performance of existing ER approaches through its data enrichment component, with a 10× speedup in uncertainty quantification for LLMs compared to baseline methods, demonstrating its efficiency and effectiveness in real-world applications.

Keywords Entity Resolution, Large Language Model, Database, Uncertainty Qualification, Entity Matching

1 Introduction

Entity Resolution (ER) aims to find and identify all tuple pairs from multiple data sources of relational

Received month dd, yyyy; accepted month dd, yyyy

E-mail: lijx@buaa.edu.cn

tables with different schema and descriptions, that refer to the same entities. Entity resolution is a central step in data integration pipelines, with the goal of integrating records from multiple datasets. Typically, ER task can be divided into two main components, namely entity blocking (EB) and entity matching (EM). Entity blocking aims to coarsely filter and rank all potential matches tuples, avoiding the quadratic computational cost for the subsequent matching phase. Entity matching aims to verify whether these tuple pairs refer to the same entities.

Nowadays, deep learning based ER models that integrate pre-trained language model (PLMs, *e.g.,* RoBERTa [1]) typically deliver superior performance. PLMs are pre-trained on extensive corpora and offer a deeper understanding of language compared to traditional word embedding techniques, allowing PLMs to enhance model generalization across new tasks and datasets more effectively.

However, these methods often cost substantial human effort to label data as matches or mismatches, alleviating overfitting issue by expanding the training set. They are mostly designed to handle relational data with well-organized schemas, so that the feature alignment relations are learned, *e.g.,* for a pair of tuples in Product domain, if they have similar Brand and SKU attributes, they may refer to the same entity with high probability. Furthermore, since PLMs are limited by input token lengths (typically no larger than 512 tokens [1–5]), they are limited to applying in short text datasets.

Example 1. Figure 1 shows two real-world examples, where tuples are both unstructured and long texts. The left part of Figure 1 is derived from the dataset Company [6], where Entity 1 represents a company's Wikipedia page and Entity 2 represents the homepage of certain company's website, both of which exceed 2000 words in length. The ER task requires one to determine whether these two entities refer to the same company or not. The right part of Figure 1 is derived from the dataset semi-text-w [7], where Entity 1 and Entity 2 represent different webpages for a watch on different e-commerce websites. The ER task here is to determine whether the two web pages describe the same product.

As discussed in Example 1, in real-world scenarios, we have to solve the ER problem based on **Unstructured** and **Long-text Entities** (denoted as ULE-ER for short). Existing ER methods struggle with the aforementioned unstructured long texts for the following reasons: (1) **Input Length Constraint:** Present PLMs-based ER approaches commonly limit the input length to 512 tokens at most, with ER performance declining as the input text gets longer. Consequently, these methods depend on truncation or summarization using TF-IDF, leading to the loss of essential information and causing errors; (2) **Needle in the Haystack [8]:** This term describes the challenge of extracting a small piece of valuable information (the Needle) hidden within a large amount of irrelevant text (the Haystack). Conventional ER techniques typically depend on schema information to align key features. However, in ULE-ER scenarios, it becomes challenging to extract and align relevant information from extensive texts in the absence of schema, and (3) **Insufficient Domain Knowledge:** Each domain uses particular abbreviations and acronyms, and a lack of understanding in the specific field may cause the ER model to focus on irrelevant noise, missing essential details.

It is worth noting that although several related works [9, 10] have introduced Large Language Models (LLMs) into ER tasks, directly replacing

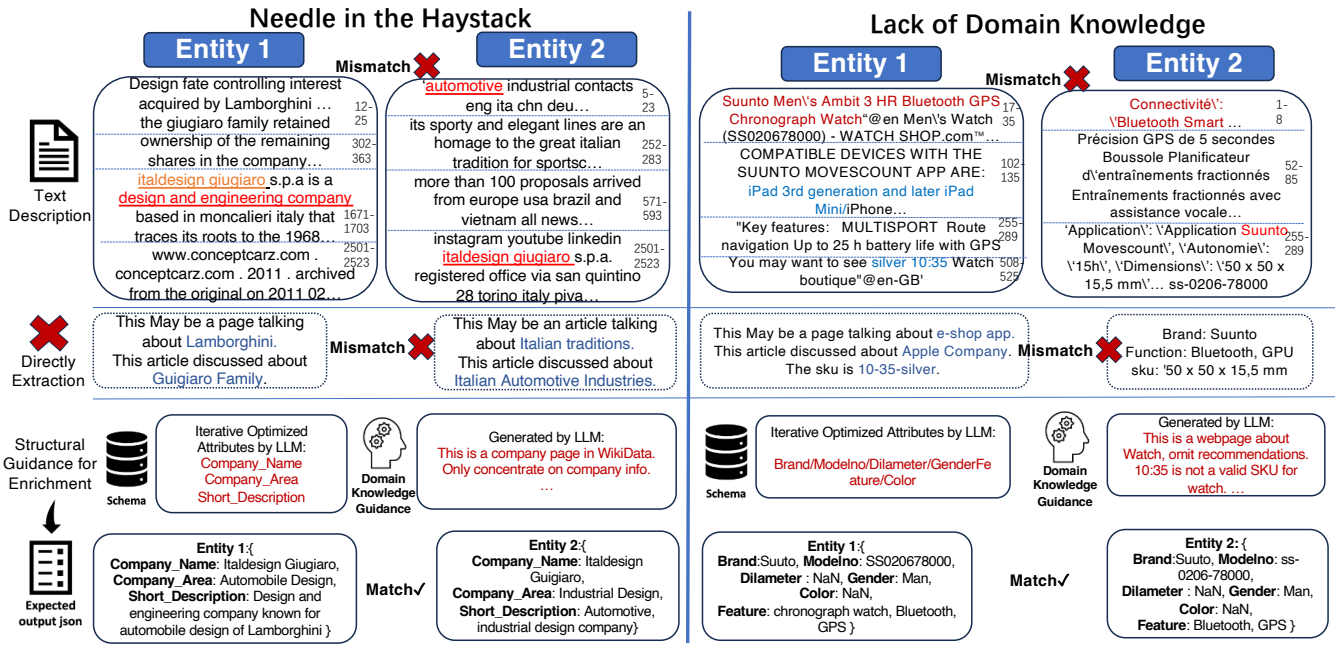

Fig. 1 Illustration of challenges in structured data extraction for unstructured long-text entities. **Entity 1 and Entity 2 refer to the same entity**. We sample multiple text chunks, and denote their token range at **the upper right corner**. Among these chunks the relevant context for ER are marked as Red, and the irrelevant context as Blue. As discussed in introduction, we summarize two main difficulties for long-text ER as *Needle in the Haystack*, which means relevant information hides in any place of a long text; and *Lack of Domain Knowledge*, which means a given LLM cannot pay attention to the relevant attributes to extract and compute.

the PLM backbone of ER methods with an LLM backbone, which has a longer input window and more model parameters, is infeasible, since such approach does not address the aforementioned limitations essentially. Furthermore, in few-shot setting, LLMs have emerged new issues, including lost in the middle, overfitting problem and high training cost. We provide a detailed experimental discussion in Section 3.3.

To solve the above challenges, a straightforward way is to extract and enrich structured information from ULE, which is also known as information extraction [11]. As shown at the bottom of Figure 1, the purpose of **structural data enrichment** (SDE for short) is to extract and fill a set of structured attributes with a clear schema from ULE to improve data quality, effectively shorten the entity text length without significant information loss for conducting ER. Due to the absence of annotated data for structured extraction, we use LLMs to generate extraction results, which is proven to be effective and robust in such task [12].

However, LLMs are proven to be vulnerable to widely known reliability issues, such as generating factual errors, also known as hallucination [13]. As shown in the middle part of Figure 1(Directly Extraction), directly querying LLMs for SDE is infeasible: Without schema constraints and domain knowledge guidance, LLMs are prone to extract incorrect attributes, leading to generating factual errors and reversely hampering data quality. Also LLMs are not efficient enough to handle large data collections of tuples.

Based on the above observations and concerns, we propose a Few-shot Uncertainty-calibrated Structural data Enrichment method for ER, FUSER for short.

To improve the quality for structural data en-

richment, FUSER employs unsupervised pairwise entity enrichment to extract well-organized structural attributes from unstructured entities using LLMs, incorporating with LLM-generated additional schema and domain knowledge guidance as constraints.

To prevent LLM from generating factual errors and hallucinations, FUSER integrates an efficient two-tier uncertainty-based calibration modules, determining reliable enrichment solutions from a variety of generated candidates above by LLM, considering both attribute- and structural-level uncertainty qualification.

Furthermore, FUSER implements a lightweight ER pipeline, capable of performing both blocking and matching tasks efficiently with up to 50 labeled positive samples. Experimental results demonstrate FUSER's superior performance in both blocking and matching tasks in ER. Our contributions can be summarized as the following:

- We propose the few-shot ULE-ER problem, addressing a real-world issue that has long been overlooked in related research, highlighting the critical role of structural data in ER.
- We propose a structural data enrichment framework, adopting pairwise enrichment method to extract well-organized structural attributes from unstructured entities using LLMs, explicitly extracting and aligning features across different data sources.
- We adopt a two-tier uncertainty qualification module, which both considers the generation quality of attribute-level and structural-level, to rank and select reliable generated attributes.
- We propose a LLM-based few-shot ER pipeline, capable of performing both blocking and matching tasks efficiently with up to 50 labeled positive samples.

The rest of this paper is organized as follows: Section 2 introduces the related work for ER and uncertainty qualification (UQ for short). Section 3 first provide the definition of the ULE-ER problem, then discuss the challenge of directly applying LLMs for ULE-ER. Section 4-Section 6 sequentially introduce the main modules in FUSER for solving few-shot ULE-ER problem. Section 7 reports a series of comparative experiments. Section 8 concludes the paper.

2 Related Work

In this section, we classify ER into entity blocking and entity matching tasks, and review existing uncertainty qualification works for LLM.

2.1 Entity Blocking

Blocking, a key step in tackling the inherent quadratic complexity of entity resolution, has been addressed through various methods [14]. We classify entity blocking into the following categories.

(1) Rule-based methods. There are methods that design handcrafted rules, *e.g.,* Matching Dependencies (MD) [15], DNF [16] and meta blocking rules [17]. (reference to [18] for more details); Besides, some works learns entity blocking rules with a few labeled training instances [19–21]. Although rule-based methods are transparent and easy to understand, they lack of enough ability of expression to retrieve tuples with the same semantic meanings;

(2) Traditional ML models. A host of works [22, 23] adopt active learning to enhance the quality of the training data so that the entity blocking models are trained more accurately;

(3) Deep learning based models. Most of works [5, 24–28] rely on representation models

and contrastive learning strategies with hard negative sampling to efficiently retrieve top-K nearest neighbours from large-scale collections of tuples.

Differing from existing works, we focus on enhancing the quality of training data by leveraging data enrichment to improve the performance of downstream entity blocking models.

2.2 Entity Matching

As the matching step of entity resolution, the entity matching is classified into the following categories:

(1) Rule-based methods [29–32] design logical rules to predict whether pairs of tuples are matched or not;

(2) traditional ML models [33–35] adopt traditional machine learning models, *e.g.,* tree-based models, SVM or Gaussian Mixture Model, and transform the EM to a binary classification task in the supervised or unsupervised manner;

(3) deep learning methods [2, 6, 36–39] that design neural networks for the EM task, *e.g.,* LSTM or transformer based neural network architectures.

Furthermore, a few works design learning strategies to boost the performance of EM, including data augmentation [3], unsupervised learning [40], transfer learning [41–44], semi-supervised learning [4] and multi-task learning [45]. There are also works that integrate entity blocking into EM, *e.g.,* [25, 27, 46] and adopt LLMs [9, 47–49].

Compared with existing works, we mainly focus on leveraging the capabilities of LLMs and generating reliable information for tuple pairs so that downstream entity matching models could have better performance.

2.3 Uncertainty Qualification for LLM

Uncertainty quantification (UQ) is a fundamental concept in machine learning and statistics, signifying that model predictions carry a certain level of variability due to incomplete information [50].

Although UQ has been well studied in the area with distinct labels, such as classification tasks in computer vision [51], NLP [52] and graph learning [53] areas, the effective UQ for open-form LLMs, *e.g.,* GPT series [54], LLaMA [55], Mistral [56] is still lack of full investigation. LLM-based open-form generation domains are flexible and effectively infinite, *i.e.,* any generations at any length that are semantic consistent with the right answer can be regarded as true.

ECE [57] calculated large scale empirical evaluations on how the model configuration(*e.g.,* model parameter size, model architecture, training loss) of LLMs affect uncertainty. Based on such finding, several works [52, 58] target at quantifying uncertainty by directly prompting the language model to generate uncertainty scores regarding to their own generations. SelfCheckGPT [59] measures the faithfulness of generations by quantifying the consistency of multi-sampled generations, *i.e.,* different generations should be consistent if the model really captured the concept of the input query. Malinin et al. [60] estimates the free-form LLMs uncertainty level by calculating the accumulative predictive entropies over multiple generations.

Among these methods, token-level UQ has proven efficient in hallucination detecting on various NLP tasks [61], however such methods fall short in evaluating semantic relations among different but consistent generation results. To compensate such shortage, there are also multiple works that focus on sentence-level UQ. Semantic Entropy(SE) [62] is presented to estimate the ”semantic equivalence” difficulty in UQ. SE gathers generations sharing the same semantics into clusters, and calculate cluster-wise predictive entropy as the un-

certainty measurement with DeBERTa-based classification model. Recent studies [63, 64] also extend UQ to both token and sentence level for more fine-grained qualification.

We aim to design metrics from multiple structural generations to characterize the uncertainty of LLMs. Based on ER scenario, our solution focuses on structural LLM output, regarding the attribute- and structural-level generative uncertainty, which are not fully explored by prior related works.

3 Problem Definition and Framework

In this section, we present the prior knowledge of uncertainty qualification with LLM in Section 3.1, formalized the studied problem in this paper in Section 3.2, discuss the challenges of directly applying LLMs for addressing the problem in Section 3.3, and present our proposed framework FUSER in Section 3.4.

3.1 Uncertainty Qualification with LLM

Large language models (LLMs) [54, 55], which typically contain billions (or more) of parameters and pre-trained on massive text data [65], have demonstrated surprising emergent behaviors and good zero-shot generalization to new tasks. Most decoder-only LLMs, *e.g.,* LLaMA [55] and Mistral [56], output generations in a free-form and auto-regressive manner, such that the probability distribution of the next token can be progressively predicted.

Here we classify LLMs as (a) black-box if they are close-source, *e.g.,* GPT series; for such models, we can only retrieve its output text without initial parameters; and (b) white-box LLMs, *e.g.,* LLaMA and Mistral, if they are open-source; this said, we can deploy them locally and get its output and initial parameters, *e.g.,* embedding and generation probability per token (see below). In this paper we mainly focus on UQ with white-box LLMs.

We denote by x and $\mathbf{s}$ the input prompt and the input query with N tokens, respectively; here LLM has to generate output $\mathbf{s}$ regarding prompt x, where $\mathbf{s}$ is an output sentence $\mathbf{s} = \{z_1, \cdots, z_N\}$ containing N tokens z_i $(1 \leq i \leq N)$. For a given LLM, denote by $p(z_i|s^{<i}, x)$ the probability of generating z_i under x, where $s^{<i} \in \mathbf{s}$ refers to the the generated tokens $\{z_1, \cdots, z_{i-1}\}$ prior to $\mathbf{s}$.

Given the generation probability p of sentence $\mathbf{s}$, UQ for LLM is to evaluate the uncertainty score $u(\mathbf{s})$ for each generation of LLM based on p, in its parameter level. The uncertainty score u is inversely proportional to the confidence of the generated text for LLM. A widely-adopted UQ method is Predictive Entropy [66](PE), defined as the entropy over the whole sentence $\mathbf{s}$:

$$u(\mathbf{s}) = -\log p(\mathbf{s}|x) = \sum_{i=1}^{N} -\log p(z_i|s^{<i}, x) \quad (1)$$

$u(\mathbf{s})$ is the accumulation of the token-wise entropy.

3.2 Task Definition

Tables. We denote T_l and T_r as two (left and right) collections (tables) of entity records (tuples) with attribute schema $R_l = \{A_1^l, \cdots, A_k^l\}$ and $R_r = \{A_1^r, \cdots, A_k^r\}$, respectively, where R_l and R_r may be different. Each record $t_l \in T_l$ (resp. $t_r \in T_r$) is a set of key-value pairs $\{A_i^l, v_i^l\}$ (resp. $\{A_i^r, v_i^r\}$), where v_i^l (resp. v_i^r) is the value of the i-th attribute A_i^l (resp. A_i^r) of tuple t_l (resp. t_r). For simplicity, we omit superscripts l and r unless specifically mentioned.

Entity resolution (ER). Denote $\mathcal{D} \subset T_l \times T_r$ as the set of pairs of tuples from T_l and T_r, respectively. A typical ER pipeline consists of two steps: blocking

and matching. The blocking step generates a candidate set $C \subset \mathcal{D}$ with a high recall by removing unnecessary comparison candidates. The matching step is then to determine whether the candidate pair $(t_l, t_r) \in C$ match or not. Existing DL-based ER methods usually rely on training set with labels, denoted as $(\mathcal{D}, \mathcal{Y}) = \{(t_l, t_r, y)\}$, where $\mathcal{Y}$ is the set of labels of tuple pairs in $\mathcal{D}$ (groundtruth). We use $\mathcal{D}$ to represent $(\mathcal{D}, \mathcal{Y})$ for short.

Blocking. Most DL-based blocking methods follow the dense retrieval paradigm [24, 25, 28] to design models. These models, denoted as $\mathcal{M}_{\text{embed}}$, serialize a tuple t as textual description, and transform it into a dense embedding vector. One can obtain the candidate set C based on the similarity between embeddings $\mathcal{M}_{\text{embed}}(t_l)$ and $\mathcal{M}_{\text{embed}}(t_r)$ of tuples t_l and t_r from T_l and T_r, respectively. Meanwhile, $\mathcal{M}_{\text{embed}}$ is further fine-tuned with training set $\mathcal{D}$ to align features between tuples in T_l and T_r.

Matching. DL-based matching models, denoted as $\mathcal{M}_{\text{match}}$, take the serialized embedding of candidate pairs (t_l, t_r) as input, and outputs a prediction $\hat{y}$ to decide whether (t_l, t_r) is match or not. Similar to $\mathcal{M}_{\text{embed}}$, $\mathcal{M}_{\text{match}}$ is trained on $\mathcal{D}$, and its parameters are updated such that it can correctly determine whether the input tuple pairs match or not.

Next, we define the unstructured long-text few-shot ER problem (ULE-ER for short). As briefly discussed in Section 1, in such setting, (a) $t_l \in T_l$ and $t_r \in T_r$ are both long text descriptions of certain entity without any schema information (*i.e.,* R_l and R_r are not available); and (2) the size of labeled set $|\mathcal{D}|$ is limited in few-shot setting, where only a small number of positive tuple pairs are available.

Definition 3.1. (few-shot ULE-ER) Given two relational tables T_l and T_r that contain long-text tuples t without any schema information, and a small set $\mathcal{D}$ of matched pairs of tuples from T_l and T_r, respectively, the objective of few-shot ULE-ER is to identify all matching tuple pairs from $T_l \times T_r$. □

3.3 Challenges of directly applying LLMs for ER

In this subsection, we outline the challenges of directly applying LLMs for addressing ER tasks. Moreover, we motivate this work by demonstrating the unsatisfied experimental result of the straightforward idea above; in particular, we list a few representative results in Figures 2 and 3 (see FUSER w/o Enrichment in Table 9 for more results).

Challenges. We list the challenges of fine-tuning LLM for ULE-ER in few-shot setting as follows:

(a) Lost in the middle [67]. It is a common phenomenon when employing LLMs on tasks with long text, *i.e.,* LLM tends to focus on information presented in the beginning/end of a long text, while neglecting (possibly) relevant information hidden in the middle (*"Needle in the Haystack"*).

(b) Overfitting issue. LLM, with a huge size of parameters, is prone to memorize and overfit to few-shot training examples $\mathcal{D}$, even worse than PLMs.

(c) High Training cost. When the size of input length increases, it leads to a rapidly increased training cost, *e.g.,* GPU memory and training time; and a larger size of labelled $\mathcal{D}$ to solve ULE-ER.

The performance of directly employing LLM on ULE-ER. In the sequel, we show that it is impractical to directly employ LLM on ULE-ER, with the unsatisfactory experimental results on datasets Company (CO) and semi-text-w (SW); on average the token length of each entity in CO and SW is 1753 and 500, up to 6755 and 3155, respectively. Moreover, we show few experimental results of our

proposed framework (FUSER in Section 3.4), illustrating that above challenges can be well addressed if we do it right. We show the results in Figures 2 and 3; see more results in Section 7.

Settings In Figure 2, we show the results of (a) LLM with Enrich (FUSER), which applies both LLM-based enrichment and pseudo-label generation for training LLM-based ER model, (b) LLM w/o pseudo-label, which is trained only on few-shot $\mathcal{D}$ with enrichment, and (c) LLM w/o Enrich, which is the straightforward LLM-based ER solution without enrichment. Moreover, in Figure 3, the input token length for LLM is extend up to 4096, while PLM w/o Enrich uses the same data $\mathcal{D}_{\text{ER}}$ with input token length of 512, and truncate exceeding text.

We report our findings below.

Effectiveness. Figure 2 represents the EM performance in 100-scale F1 score for the above four methods. Directly applying LLM for ULE-ER suffers a significant performance drop. The reason lies on that changing backbone model from PLM to LLM does not essentially solves the issues above.

Comparing the performance of LLM with and w/o Enrich, LLM suffers from calibrating its attention to the right place in long-text scenarios, resulting in "*lost in the middle*" phenomenon. PLM-based methods have a more significant performance drop, since they have to truncate useful information that exceeds its inherit input length.

Comparing LLM with and w/o pseudo-label, we can see that even if the data is well-structured, the few-shot $|\mathcal{D}|$ still hampers LLM-based ER performance, leading to significant overfitting issue.

Efficiency. Figure 3 illustrates the training cost of dataset CO under different input length (*w.r.t.* cutoff length) when fine-tuning LLM. We can see that a shorter input token length, *e.g.,* 1024, cannot cover enough relevant information; worse still, a larger input token length window brings out increasing training cost in both training time and GPU memory usage.

According to the above observations, we can see that directly apply LLM for ULE-ER is impractical, both in effectiveness and efficiency.

3.4 The FUSER Framework

To solve the aforementioned concerns, we propose FUSER, a framework with three sequentially executed components: structural data enrichment (Section 4), uncertainty qualification module (Section 5), and few-shot ER with LLM (Section 6).

Structural data enrichment (SDE). This component takes ULE in T_l and T_r as input, conducts pairwise enrichment with recursive structure-aware chunking strategy, and outputs enriched entities with summarized and structural format. SDE is designed to solve *Needle in the Haystack* problem by leveraging RAG-based LLM generation with structural constraint (in response to challenge (a)).

Uncertainty qualification module (UQ). This module calibrates and ranks multiple generated candidates from the previous SDE part; it is to ensure the semantic consistency of the enrichment results with minimal additional computing cost (in response to challenge (c)).

Few-shot ER with LLM. It takes the enriched and calibrated high-quality data as input, significantly shortens the input length, and reduces the training cost of both PLM and LLM-based ER solutions. Moreover, it cooperates with UQ-based sampling strategy to extend the training set with pseudo-labeled positive/negative pairs, such to alleviate the overfitting issue (in response to challenge (b)).

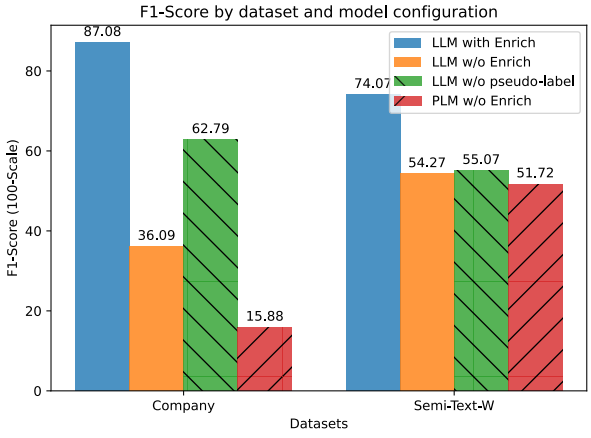

Fig. 2 Entity matching performance of LLMs and PLM-based ER solutions.

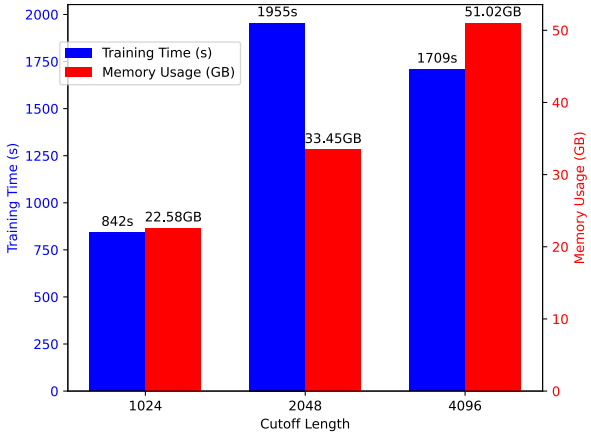

Fig. 3 Training cost of LLMs in different maximum input token lengths. We apply the contamination-free packing [68] method to merge/pack short inputs into a single input at cutoff length, so the sample number in 4096 is fewer than that in 2048, leading to less training time.

4 Retrieval-Augmented Pairwise Data Enrichment with LLMs

In this section, we present the data enrichment part of FUSER with LLMs; it has a simple but effective paradigm, with the overall pipeline in Figure 4.

Given two relational tables T_l and T_r without any structural information, FUSER sequentially conducts data enrichment with three modules:

(1) FUSER applies a RAG component to retrieve and select tuple pairs $\mathbf{T} \subset T_l \times T_r$, maximizing common information. (Section 4.1)

(2) FUSER iteratively selects additional enriched schema R for $\mathbf{T}$, as well as domain-related prompt guidance g, which are both generated by LLMs and validated by matching relevance with label; in particular, $\mathbf{T}, R, g$ will be concatenated and queried by a local LLM. (Section 4.2)

(3) it takes tuple pairs set $\mathbf{T}$, selected extra schema R_{extend} and optimized prompt set g as input, and generates structured enrichment output results $\mathbf{T_s}$ in schema R_{extend}. (Section 4.3, 4.4)

Table 1 General notations with corresponding descriptions.

Symbol	Description
ULE	Unstructured Long-Text Entities
UQ	Uncertainty Qualification
SDE	Structural Data Enrichment
PLM	Pre-Trained Language Model, *e.g.,* RoBERTa
LLM	Large Language Model, *e.g.,* LLaMA, Mistral
G	Generative model, represent LLM
T_l, T_r	Left and right relational table for ER
t_l, t_r, y	Tuple/record pair with match/mismatch label y
$A_i \in R$	Single attribute A_i in schema R
$v \in \mathcal{V}_A$	Attribute value v and value set $\mathcal{V}$ for attribute A
$\mathcal{M}_{\text{embed}}$	Embedding model
$\mathbf{T}$	Selected tuple pairs, $(t_l, t_r) \in \mathbf{T} \subseteq T_l \times T_r$
$\mathcal{D}$	Train set $\mathcal{D}$ with a few positive pairs
$\mathcal{D}_{\text{update}}$	$\mathcal{D}$ adding self-labeled tuple pairs
R_{extend}	Extend schema generated by LLM
$g, g_{\mathbf{T}}$	Domain knowledge guidance/prompt for $\mathbf{T}$
$\mathcal{S}(t)$	Structured enrichment result for tuple t
$E_t, u(t)$	Uncertainty qualification for tuple t
$\text{sim}(\cdot)$	Semantic similarity for any input pairs
$p(\mathbf{s})$	generation probability for sequence $\mathbf{s}$

4.1 Tuple Pairs Selection with RAG

Information retrieval from unstructured long texts (ULE) has been a long-standing research issue. In the context of ER scenarios, the challenges of retrieval and enrichment are even more daunting. These challenges can be outlined as follows:

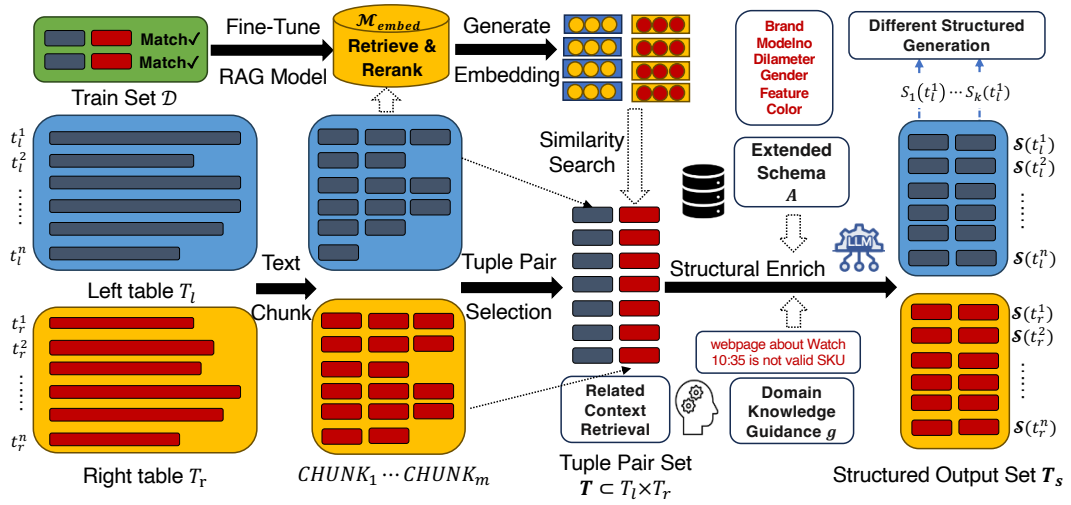

Fig. 4 Illustration of structural data enrichment component in FUSER.

1. **Needle in the Haystack:** Effective information is scattered throughout the long text, but directly embedding the entire text without segmentation into a single dense embedding vector results in significant information loss.
2. **Distribution Discrepancy Problem:** In ER, the left table T_l and the right table T_r typically originate from different data sources. An un-tuned dense retrieval model struggles to bridge the domain gap.
3. **Hallucination Issue:** Data enrichment focused solely on a single tuple often leads to LLMs producing factual errors or irrelevant information, due to the lack of contextual information.

We propose a series of optimization methods to address the aforementioned challenges. focusing on both enhancing the granularity of text segmentation and improving the robustness of data alignment across different sources. Please check the left part of Figure 4 for a brief illustration.

Text Chunk Firstly, to tackle the "*Needle in the Haystack*" problem, we apply the recursive structure-aware chunking strategy [69, 70], a hybrid solution combining fixed-size sliding window and structure-aware splitting. It attempts to balance fixed chunk sizes with linguistic boundaries, offering precise context control. In detail, such method tends to divide the initial long text for ULE t under several structural separators, *e.g.,* newline characters or HTML symbols. Next we initialize the sliding window, denoted as CHUNK_i for i-th sliding window, covering several consistent structural segments within the input window size for embedding model $\mathcal{M}_{\text{embed}}$. Please check Figure 5 for illustration.

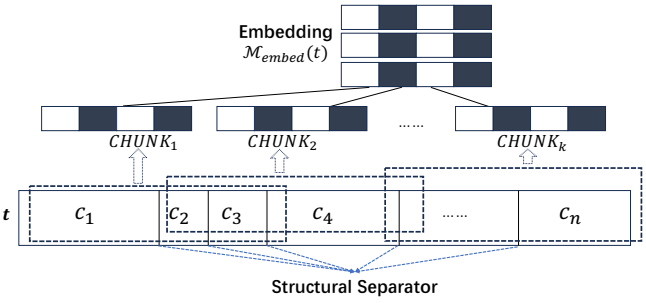

Fig. 5 Illustration of recursive structure-aware chunk in FUSER.

For example, the first chunk covers l_1 segments, which sum do not exceed the input window size:

$$\text{CHUNK}_1 = \{c_1, \cdots, c_{l_1}\} \tag{2}$$

As the sliding window moves forward, next chunk cover l_2 segments:

$$\text{CHUNK}_2 = \{c_{m+1}, \cdots, c_{m+l_2}\} \tag{3}$$

where the discard segments $\{c_1, \cdots, c_m\}$ is approximately the step length, and the covered segments $\{c_{m+1}, \cdots, c_{m+l_2}\}$ do not exceed the input window size. Such procedure goes on until the last segment reaches the end of text t.

Such method effectively keeps the structure granularity and semantic integrity, avoiding the risk of segment relevant information into distinct chunks. It will also keep the coherence of context, and highlight the middle text in t, by overlapping it with multiple chunks, avoiding "lost in the middle" phenomenon.

RAG Model To address the issue of domain gap between the left table T_l and the right table T_r, we use a lightweight Sentence-Bert [5] model as the initial RAG embedding model $\mathcal{M}_{\text{embed}}$, based on the train set $\mathcal{D}$ which contains multiple matched tuple pairs. Furthermore, we apply the initialized $\mathcal{M}_{\text{embed}}$ to mine hard negative samples across tables for data labeled as positive in $\mathcal{D}$.

For example, if $(t_l, t_r) \in \mathcal{D}$ is labeled to be match, then $\mathcal{M}_{\text{embed}}$ will search top-k similar tuples in T_r for t_l exclude t_r as negative tuples, and vice versa. We denote the updated train set from $\mathcal{D}$ as $\mathcal{D}_{\text{update}}$, containing positive and self-labeled negative samples. Subsequently, we fine-tune $\mathcal{M}_{\text{embed}}$ using contrastive loss [71] with $\mathcal{D}_{\text{update}}$. This process aims to minimize the domain gap between T_l and T_r.

Finally, based on the chunked text set, we embed all tuples of T_l and T_r using the fine-tuned embedding model $\mathcal{M}_{\text{embed}}$, and for each tuple t_1 in the left table, we select the top-k most similar tuples from the right table $\{t_2^1, \cdots, t_2^k\} \in T_r$, iterating over the left table T_l to form the tuple pair set $\mathbf{T}$, and vice versa. We further denote the organized tuple pairs set as $\mathbf{T}$. This design is based on a simple assumption that semantically similar but structurally dissimilar tuple pairs can complement each other's information with high confidence, thereby avoiding the hallucination issue caused by in-context guidance.

4.2 Extended Schema and Domain Knowledge Generation

Directly using LLM for enrichment of the tuple pair set $\mathbf{T}$ is considered risky, as $\mathbf{T}$ may contain tens of thousands of tuple pairs without schema information, and lacks effective domain knowledge to restrict the generation quality. However, recall the updated labeled training dataset $\mathcal{D}_{\text{update}}$ from Section 4.1, which contains a comparatively smaller volume of data with high-quality annotations. Based on $\mathcal{D}_{\text{update}}$, we query LLM to determine the schema and domain knowledge for $T_l \cup T_2$.

Domain Knowledge Generation Firstly, determining the domain knowledge guidance is relatively straightforward. We sample and serialize several tuple pairs (t_l, t_r, y) from $\mathcal{D}_{\text{update}}$, and input it into the generative LLM G, allowing it to generate domain knowledge guidance based on the provided tuples. We apply the global-level UQ module (detailed later in Section 5) to select the most convincing k guidance for the final injected domain knowledge as prompt, denoted as $g_{\mathbf{T}}$.

Extended Schema Generation The schema generation for $\mathbf{T}$ entails calculating the correlation between the extended attribute value set $\mathcal{V} = \{v_1, \ldots, v_n\}$ and the label set $\mathcal{Y}$. Specifically, tuples (t_l, t_r, y) extracted from $\mathcal{D}_{\text{update}}$ enable LLM to generate various possible additional attribute candidates R_{all}, enhancing the ER tasks with labels y.

Using LLM, we query tuple pairs from $\mathcal{D}_{\text{update}}$ to generate enriched values based on R_{all} and measure the correlation between these values and $\mathcal{Y}$. We select the top 5 attributes with the highest correlation scores from R_{all} to form the final extended schema R_{extend}, as these are most relevant to the EM task performance.

In detail, to evaluate the quality of extended attribute R_{extend}, for each tuple pair $(t_l, t_r, y) \in \mathcal{D}_{\text{update}}$ and attribute $v \in \mathcal{V}_A$ for attribute R_{extend} generated by LLMs, we utilize $\mathcal{M}_{\text{embed}}$ to transform v into embeddings $\mathcal{M}_{\text{embed}}(v)$. We utilize $\text{Sim}(v, v')$ to evaluate the semantic similarity of two attribute values v, v' in embedding level. To assess the correlation between $v \in \mathcal{V}_a$ and the labels $\mathcal{Y}$, we apply the Point-Biserial Correlation [72]:

$$\text{PBC}(v, \mathcal{Y}) = \frac{\bar{y}_1 - \bar{y}_0}{s} \cdot \sqrt{\frac{n_1 n_0}{n(n-1)}} \tag{4}$$

where $\bar{y}_0$ and $\bar{y}_1$ represent the average values of attribute v under labels True (*w.r.t.* match pairs) and False (*w.r.t.* mismatch pairs) respectively in $\mathcal{Y}$, s is the standard deviation of $\{\text{Sim}(v, v') | v' \in \mathcal{V}_A, v' \neq v\}$ within $\mathcal{D}_{\text{update}}$, n_0 and n_1 are the counts of mismatched and matched instances, and $n = |\mathcal{D}_{\text{update}}|$. The top 5 attributes in R_{all} with the highest PBC values are selected for the extended schema R_{extend}, R for short.

The aforementioned methods are capable of both black-box and white-box LLMs. For consistency, we utilizes the same white-box local LLMs to handle the above steps.

4.3 Structured Data Enrichment

After acquiring the tuple pairs set $\mathbf{T}$, domain knowledge guidance g, and extended schema R, we serialize and concatenate g, each tuple pair $(t_l, t_r) \in \mathbf{T}$, and R, and query the white-box LLM G, denoted as $G(g, (t_l, t_r), R)$. Such query for LLM leads to generate JSON structured data, denoted as $\mathcal{S}(t_l)$ and $\mathcal{S}(t_r)$, both of which are multiple dictionaries formatted outputs $\{S_i(t)\}_{i=1}^{k}$ under schema R. Different from existing multi-step retrieval methods [73], FUSER generate $S_i(t_l), S_i(t_r)$ with all attributes in R through single-step generation query for tuple pair t_l, t_r.

Additionally, we record the token-level generation probabilities for single structured outputs $\mathcal{S}(t)$ as $P(t)$. These probabilities represent the generation likelihood of each token within $\mathcal{S}(t_l)$ and $\mathcal{S}(t_r)$, further utilized for uncertainty qualification.

4.4 Implementation

Ensuring stable JSON output from a white-box LLM is non-trivial. To address this challenge, we employed the lm-format-enforcer [74] method to guarantee consistent JSON formatting by the LLM. This method integrates a character-level parser with a tokenizer prefix tree to establish a sophisticated token filtering mechanism. Additionally, we leveraged vLLM [75]-based efficient inference method, incorporating optimizations through KV cache and similar query aggregation to accelerate the enrichment process. In our experiments, with 2 A800 GPUs, we complete the whole enrichment of 10,000 tuple pairs within an average of 535 seconds, with a parsing failure rate of less than 0.1% and an average generation speed exceeding 9,000 tokens per second.

5 Two-Tier Uncertainty Qualification

In this section, we present the UQ part of FUSER, which aims to select and calibrate high-quality enrichment results from LLM-generated candidates

$\mathbf{T_s}$. We present an illustration figure of UQ in Figure 6, and UQ calculating demonstration in Figure 7.

In summary, in Section 5.1, we analyze the necessity of applying UQ to measure reliable outputs for different generations of LLMs, as well as the limitations of existing UQ methods for structured outputs. To address such concern, in Sections 5.2 and 5.3, we propose attribute-level and entity-level UQ calculation methods, focusing on the local and global levels, respectively. Section 5.4 summarizes the above methods and introduces strategies for selecting generated outputs based on UQ results. Finally, Section 5.5 concludes by discussing how we accelerate the UQ calculating process.

5.1 Analyzing exist UQ methods

Although Section 4 adopts SDE to generate a set $\mathcal{S}(t)$ of multiple structured output results for each tuple t, the quality measurement and selection of these different generated results presents a non-trivial problem. Due to differences in pairwise context and prompt guidance, for the same attribute $A \in R$, the enriched attributes value $\mathcal{V}_A$ for a given tuple t can vary and be contradictory. While several works on UQ for LLM propose various metrics for measuring output quality, these methods are not directly applicable to the pairwise enrichment scenario. Figure 6 provides an illustrative example of this issue.

Example 2. As shown in Figure 6 (follows the left example in Figure 1), for the input tuple t on the left, to enrich the attribute *Company-Name*, LLM answers multiple contradictory results due to differing contexts. If one were to solely rely on the token probability associated with each generated result, as estimated by the PE method described in Section 3.1, an erroneous selection "*Lamborghini*" might be chosen as the result because it appears earlier in the text, thus exhibiting lower uncertainty and high probability.

To solve such *False but Convincing* hallucination by LLM, recent works, *e.g.,* Semantic Entropy [62], SAR [63] point out that semantic consistency is key to calibrate UQ results. To achieve this, they sample LLM generation multiple times for single query, and additionally compare the semantic correlation within these answers. Generations with higher semantic correlation with other outputs will be provided with a higher weight, and vice versa.

However, existing UQ methods predominantly cater to open-form generation with LLMs and primarily focus on token-level and sentence-level estimations and selections. These approaches are unsuitable for the ULE-ER scenario in FUSER due to several critical reasons.

First, token-level selections are inappropriate for constrained structural outputs as they may disrupt the inherent structure of the data, rendering the uncertainty measurements meaningless in the context of structured data.

Similarly, sentence-level UQ is not applicable either, as a single sentence may encompass multiple structured outputs, complicating the clarity and utility of such measurements in this context.

In response to these limitations, we propose a novel UQ approach that operates at both the attribute-level and entity-level. This method provides a finer granularity in calculating uncertainty, significantly enhancing the relevance and applicability of UQ in structured data environments, with high efficiency.

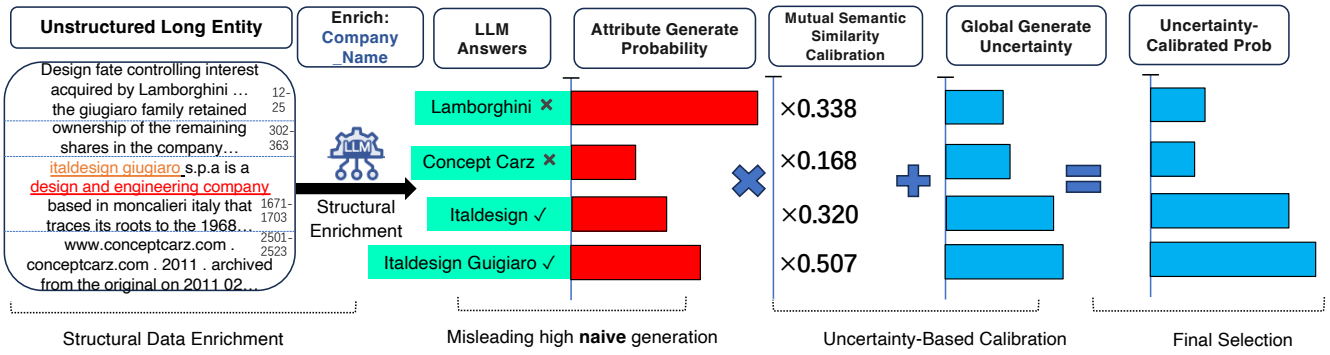

Fig. 6 Illustration of uncertainty calibration component in FUSER, where the example at left corner is extracter from Figure 1. The wrong answer for enriching attribute *Company-Name* is marked as ×, and right answer as ✓. After structural data enrichment process in Section 4, LLM may generate multiple candidates for single entity. Uncertainty calibration component aims to rank and select the right answer, alleviating hallucination issue.

5.2 Attribute-level Uncertainty Qualification

As discussed in [76], attribute values that have higher relevance scores, *i.e.,* semantically consistent, are more convincing than others. Therefore, we simply reduce sentence uncertainty by enlarging its generative probability with a relevance-controlled quantity.

Given a tuple t with attribute $A \in R$, in which containing different enriched attribute values $\mathcal{V}_A(t) = \{v_A^1, \cdots, v_A^k\}$, while the domain knowledge guidance over t is remarked as g for short, the attribute-level UQ score $\{u(v_A^i) | v_A^i \in \mathcal{V}_A(t)\}$ is calculated as:

$$u(v_A^i) = E_t(v_A^i, \mathcal{V}_A(t), g) = -\log\Big(p(v_A^i|g)\Big) + \underbrace{\frac{\sum_{i\neq j} \mathrm{sim}(v_A^i, v_A^j) p(v_A^i|g)}{t}}_{\text{attribute relevance}} \tag{5}$$

where $p(v_A^i|g)$ is the generative probability of v_A^i, and t is the temperature used to control the relevance shifting scale. And the $\mathrm{sim}(v_A^i, v_A^j)$ is the semantic similarity between attribute pair (v_A^i, v_A^j), calculated by the embedding model $\mathcal{M}_{\text{embed}}$ in Section 4.

It is worth nothing that Eq. 5 shares a similar form with SE [62] and SAR [63], both of which aims to reduce the uncertainty of semantically consistent sentences. SE conduct such component with bi-directional entailment prediction, while SAR achieves this with weighted relevance scores, both on sentence level. FUSER calculate E_t in attribute level. We denote $u(v_A^i) = E_t(v_A^i, \mathcal{V}_A, g)$ as the uncertainty score for generated attribute value $v_A^i \in t$ over attribute A. The middle part of Figure 6 demonstrate a simple illustration of attribute-level calibration.

5.3 Entity-level Uncertainty Qualification

Different from existing sentence-based UQ methods, FUSER also pay attention to entity-level UQ calibration, denoted as $u(S(t))$, qualifying the generation quality of single SDE $S(t) \in \mathcal{S}(t)$. It is worth noting that $S(t)$ is constrained in JSON structure with tokenizer prefix tree, so the token-level generation probability $p(z_i|S(t)^{<i}, x)$ for $S(t)$ may be biased from the traditional open-form LLM generation condition.

To avoid such impact for structural output, we calculate the generation probability of $u(S(t))$ by the sum of its all enriched attributes in extended schema $R = \{A_1, \cdots, A_n\}$. We mark $v_{A_j}^i$ as single enriched value for attribute A_j in $S_i(t)$, while $\mathcal{V}_{A_j}(t)$

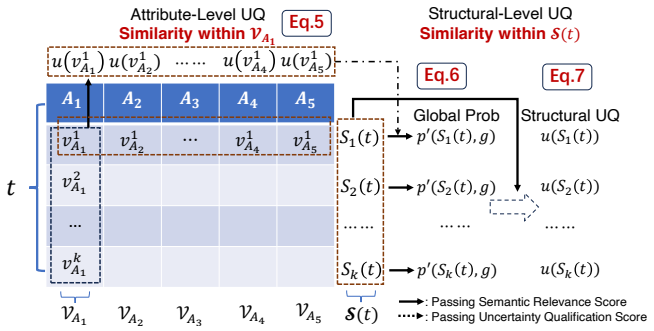

Fig. 7 Illustration of attribute- and entity-level UQ calculation for FUSER.

as all enriched value set for attribute A_j in all enriched solutions $\mathcal{S}(t) = \{S_i(t)\}_{i=1}^{k}$, representing different k enrichment structured results for tuple t. The generation probability for $S_i(t)$ is:

$$p'(S_i(t), g) = \exp\left(-\sum_{j=1}^{n} E_t(v_{A_j}^i, \mathcal{V}_{A_j}, g)\right) \tag{6}$$

As presented in Figure 7, if we expand $\mathcal{S}(t)$ to a relational table, where i-th row represents structured output $S_i(t)$ and j-th column represents attribute $A_j \in R$, in this perspective Eq.5 can be regarded as vertical comparison across different values for the same attribute A_j, and Eq.6 can be regarded as horizontal estimation across different attributes in R for the same output $S_i(t)$.

Based on Eq.6, we slightly modify Eq.5 to estimate UQ for $S_i(t) \in \mathcal{S}(t)$:

$$E_t(S_i(t), \mathcal{S}(t), g) = -\log\left(p'(S_i(t)|g)\right) + \underbrace{\frac{\sum_{i\neq j}^{1\le j\le k} \mathrm{sim}(S_i(t), S_j(t)) p'(S_j(t)|g)}{t}}_{\text{structural relevance}} \tag{7}$$

Eq.6 also applies $\mathcal{M}_{\text{embed}}$ to calculate semantic relevance, denoted as $\mathrm{sim}(S_j(t), S_i(t))$. The right part of Figure 6 illustrate the entity-level uncertainty calibration. We highlight that calculating Eq.7 only requires additional comparison across $\mathcal{S}_t$, since attribute-level comparison result across $\mathcal{V}_{a_j}$ has been computed and cached in the previous step.

Recall the domain knowledge generation phase in Section 4.2, we can also apply Eq.7 to calculate the UQ score $u(g)$ for various $g \in \mathcal{G}$, by replacing p' with generation probability for g.

5.4 Overall Measurement and Enrichment Selection

After calculating attribute-level UQ $u(v_A^i)$, and its entity-level contextual UQ $u(S_i(t))$ where $v_A^i \in S_i(t)$, the final uncertainty estimation for v_A^i is denoted as

$$U(v_A^i) = (1-\lambda)u(v_A^i) + \lambda u(S_i(t)) \tag{8}$$

where λ is the hyperparemeter controlling the relation between attribute and entity level UQ.

Recall that a lower certainty indicates a better quality result, given UQ scores $\{U(v_A^1), \cdots, U(v_A^k)\}$ for k values $\{v_A^1, \cdots, v_A^k\} \in \mathcal{V}_A$, we have two different sampling strategy to generate the final selection for $t[A]$.

One is greedy-based selection with certainty, marked as Certain($\mathcal{S}(t)$), where we always select v_A^i with the lowest UQ score $U(v_A^i)$. Such solution is similar to the greedy decoding strategy for LLM, which only generate results with the highest probability, however hampers data diversity.

Another strategy is uncertainty-based sampling method, denoted as Prob($\mathcal{S}(t)$) ,where for $t[A]$, $\{v_A^1, \cdots, v_A^k\}$ is selected by probability $\mathrm{Softmax}(-U'(v_A^1), \cdots, -U'(v_A^k))$, and $U'(v_A^i)$ is normalized from $U(v_A^i)$. Such selection aims to upsample various diversified and reliable data from LLM-generated results, similar with the speculate decoding strategy [77] for LLM.

5.5 Implementation Optimization

Previous work on LLM open-form generation for calculating UQ typically involves high computational complexity. According to the analysis in existing literature [63, 78], computing UQ generally consists of three parts: LLM generation, Logits Computing, and Semantic Relevance Calculation.

Among these process, LLM generation part requires generating multiple different responses for the same input query. Fewer response generations can impair the accuracy of UQ, while a larger number of responses significantly increases computational costs [78]. Logits Computing part involves parsing these responses, extracting relevant information, and calculating the generation probability p. Semantic Relevance requires frequently calculating the semantic relevance score across generated outputs. Previous research lacks decoupling of these processes and batch computation optimization, leading to high computational complexity in the enrichment and UQ process.

To improve the computational efficiency for UQ, we have decoupled above three parts and implemented batch inference. In the generation and logits computing phases, we have accelerated the LLM generation speed with vLLM, and increase the efficiency of parsing structural data (detailed in Section 4.4). In the semantic relevance phase, we have aggregated and batch-computed the similarity score $\mathrm{Sim}(\cdot)$ at both the attribute level and the entity level, avoiding replication computations of same value pairs across different entities.

Combining such optimization strategy, as detailed in Section 7.4 and Table 8, FUSER achieves a non-trivial 10× speedup than baseline UQ solutions.

6 Workflow for Few-Shot ER with LLM

We present a light-weighted workflow that solves blocking and matching tasks coherently, based on fine-tuning local LLMs in RAG paradigm.

6.1 Dense-Retrieval based Entity Blocking Model

The training process and backbone for the blocking model, denoted as $\mathcal{M}_{\text{block}}$, is highly similar with $\mathcal{M}_{\text{embed}}$, which is extensively discussed in Section 4. However, a critical distinction lies in the nature of the training data. Instead of using the text chunks for t as in $\mathcal{M}_{\text{embed}}$, $\mathcal{M}_{\text{block}}$ employs a more compact yet information-dense structured output $\mathcal{S}(t)$.

The initial dataset $\mathcal{D}_{\text{ER}}$ for training comprises a small set of manually annotated match tuple pairs (t_1, t_2, y). In an effort to robustly augment $\mathcal{D}_{\text{ER}}$, the positive samples within $\mathcal{D}_{\text{ER}}$ are derived from the combined sets $\mathsf{Prob}(\mathcal{S}(t_1)) \times \mathsf{Prob}(\mathcal{S}(t_2))$. Conversely, the hard negative samples are generated from $\mathsf{Prob}(\mathcal{S}(t_1))$, and are then sampled based on embedding similarity from $\mathsf{Prob}(\mathcal{S}(t_i))$ for $i \neq 1, 2$.

Such upsampling strategy is designed to mitigate the risk of the model overfitting to the biased distribution of a limited number of highly structured annotated samples. Upon completing the training of $\mathcal{M}_{\text{block}}$ with contrastive learning loss, during the inference phase, $\mathsf{Certain}(\mathcal{S}(t))$ is employed to sample and generate embedding vector for the tuple t. This ensures the stability and reliability of the inference results.

6.2 LLM-based few-shot Entity Matching Model

The training set $\mathcal{D}_{\text{ER}}$ is also applied for the LLM-based matching model $\mathcal{M}_{\text{match}}$. However, when

sample $(S(t_1), S(t_2), y) \in \mathcal{D}_{\text{ER}}$, the serialize input of the sample for fine-tuning LLM contains multiple values:

$$\text{Serial}(t_1, t_2) = \{g, [S(t_1), S(t_2)], ICL[S(t_1), S(t_2)], y\} \quad (9)$$

In Eq.9:

- g means domain knowledge guidance, that are selected in Section 4, and acts as prompt.
- $[S(t_1), S(t_2)]$ denote serialized structural output for t_1, t_2, represents as input.
- $ICL[S(t_1), S(t_2)]$ means multiple in-context demonstrations $\{t_i, t_j, y\} \in \mathcal{D}_{\text{ER}}$ with match and mismatch examples, which is selected base on semantic similarity derived from $\mathcal{M}_{\text{block}}$

We apply supervised fine-tuning for a white-box local LLM to conduct EM task with LoRA [79]. Similarly, during inference with trained $\mathcal{M}_{\text{match}}$, $\text{Certain}(\mathcal{S}(t))$ is employed to sample enrich result.

7 Experimental Results

We empirically evaluated the performance of our method, FUSER, on 6 benchmark datasets. We aim to answer the following questions:

(1). In general, can LLM-based FUSER effectively solve ULE-ER problem in few-shot setting, compared to that of PLM-based ER baselines?

(2). How is the effectiveness of data enrichment in FUSER? Can it outperform existing approaches based on text augmentation and summarization?

(3). Does the UQ component in FUSER outperform other LLM-based UE methods *w.r.t.* computational efficiency, without degrading effectiveness?

We presented our experimental settings in Section 7.1. Section 7.2 reports our results and findings in both entity blocking and entity matching tasks, regarding Question 1; Section 7.3 compare the effectiveness of data enrichment module in FUSER with 4 data augmentation baselines for ER, answering Question 2; Section 7.4 compares the UQ module of FUSER with 3 widely-adopted UQ baselines in both effectiveness and efficiency, in response to Question 3. Section 7.5 and 7.6 provides ablation study and hyperparameter sensitivity analysis for FUSER.

7.1 Experimental Settings

We start with our settings.

Datasets. We conducted experiments on following 6 benchmark datasets, as shown in Table 2.

Company. [80] A benchmark dataset applied for Ditto [2]. The left and right tables are the Wikipedia page and the website homepage of some companies, respectively, both of which are unstructured.

semi-text-w. [7] A benchmark dataset applied for PromptEM [4]. The left table is semi-structured data, while the right table is unstructured webpage content for certain watch from different data sources.

semi-text-c. [7] A benchmark dataset from machamp [7], applied for PromptEM [4]. The left table is schema-free structured data, while the right table is unstructured webpage content for certain electronic product from different data sources.

wdc-all-small [81]. A benchmark dataset from WDC dataset [81]. Both left and right tables in it are unstructured text description for certain product from different data sources.

Walmart-Amazon-Dirty [80]. A benchmark dataset applied processed from Ditto [2]. Both left and right table are unstructured description of product

from different sources.

Abt-Buy-Dirty [80]. A benchmark dataset applied and processed from Ditto [2]. Both left and right table are unstructured description of product from different sources.

Following Ditto, we obtained a few-shot labeled training dataset $\mathcal{D}$ by randomly sampling 50 positive tuple pairs, and retained the left and right tables as unlabeled. The statistics of all datasets are summarized in Table 2.

Baselines For entity blocking task, we select the following 3 widely-adopted baselines:

- DeepBlocker [24]: a transformer-based entity blocking model in the self-supervised strategy;
- Sudowoodo [25]: a self-supervised contrastive learning framework that combines blocking and matching.
- STransformer [5]: a pre-trained BERT-based model that converts sentences into dense embedding vectors. STransformer applies the same backbone model and training loss function with FUSER, so STransformer can be regarded as an ablation of FUSER without data enrichment.

For entity matching task, we select the following 5 baselines:

- Ditto [2]: an entity matching model based on pre-trained RoBERTa [1] model, and uses various data perturbation, domain knowledge injection and text summarize strategies to augment data.
- Rotom [3]: an entity matching model leveraging PLMs and select various data augmentation operators through reinforcement learning.
- PromptEM [4]: a PLMs-based entity matching model, which is fine-tuned with prompt-tuning. PromptEM leverages a student-teacher training framework, and shows better performance in few-shot learning scenarios.
- Unicorn [45]: a multi-task data matching framework with mixture-of-experts architecture, and pre-trained on various task-relafed corpus.
- JellyFish [9]: a LLM-based entity matching model using LoRA-based instruction-tuning method.

For uncertainty qualification efficiency evaluation, we also apply the following unsupervised baselines, selected from lm-polygraph [78]:

- PE [66]: predictive entropy method, which is defined as the entropy over the whole sentence, detailed in Eq.1.
- SE [62]: semantic entropy method, which introduce the 'semantic entropy' difficulty in UQ of open-form LLMs, and tackles this issue by gathering sentences containing the same meaning into clusters and calculating clusters-wise entropy with DeBERTa [82] model, predicting entailment relations.
- SAR [63]: semantic entropy method with shifting attention to relevance improvement, which pay attention to both token and sentence-level UQ. SAR evaluate sentence relevance with cross-encoder model STSB-RoBERTa, directly calculate sentence pair similarity.

Evaluation Metrics For entity blocking, following existing work [24], we report top-k recall and top-k precision. Denote left table as T_l, while right table as T_r, all of which exclude duplicated records, and their match record pair set is marked with T_{gt} as

Table 2 Datasets used in our experiments, # means Number of, T_l, T_r represents left/right table in ER; for schema type, unstructured means each tuple in this table only contains text description without schema, while semi-structured means each tuple in this table contains various schema.

Dataset	Domain	# All	# Match	Schema Type of T_l	Schema Type of T_r	# $\lvert T_l\rvert$-#$\lvert T_r\rvert$	$\lvert\mathcal{D}\rvert$/ (# All)
Company(CO) [80]	Company	112,632	28,200	Unstructured	Unstructured	28,200-28,200	0.07%
Semi-Text-W(SW) [7]	Watch	9,234	1,089	Semi-Structured	Unstructured	9,234-9,234	0.54%
Semi-Text-C(SC) [7]	Electronic	20,897	2,940	Semi-Structured	Unstructured	20,897-20,897	0.24%
WDC-All-Small(WS) [81]	Product	13,436	3,516	Unstructured	Unstructured	7,437-8,091	0.77%
Abt-Buy(AB) [80]	Product	9,575	1,028	Unstructured	Unstructured	1,081-1,092	0.87%
Walmart-Amazon(WA) [80]	Product	10,242	962	Unstructured	Unstructured	2,554-22,074	0.81%

groundtruth . For each tuple in T_l, top-k prediction means predict k most relevant tuples from T_r, and the prediction set is denoted as T_{predict}. Then top-k recall and precision stands for

$$\text{Recall}(k) = \frac{|T_{\text{gt}} \cap T_{\text{predict}}|}{|T_{\text{gt}}|} \tag{10}$$

$$\text{Precision}(k) = \frac{|T_{\text{gt}} \cap T_{\text{predict}}|}{|T_{\text{predict}}|} \tag{11}$$

For entity matching, following existing work [2], we report precision(P), recall(R) and F1-score(F) here. All results are reported in 100-scale.

Configurations We select Mistral-7B [56] as the backbone model of $\mathcal{M}_{\text{match}}$ and G, bge-rerank-large [83] as the pairwise similarity calculation model for UQ in Eq.5 and Eq.7, bge-large-en [83] as the embedding model for $\mathcal{M}_{\text{embed}}, \mathcal{M}_{\text{block}}$. We fine-tune the $\mathcal{M}_{\text{embed}}, \mathcal{M}_{\text{block}}$ with FlagEmbedding [84] framework, and the LLM-based $\mathcal{M}_{\text{match}}$ with llama-factory [79].

Following existing UQ works [63], we set the top-k sampling parameter for data enrichment as 5, relevance parameter t in Eq.5 and Eq.7 as 1e-3, and the learning rate of $\mathcal{M}_{\text{embed}}, \mathcal{M}_{\text{match}}$ as 1e-5, 1e-4 separately. Following the findings in [69, 85], we set the sliding window size as 512, and step size limit as 256 for text chunk.

We conduct our experiment on a single machine powered by 1.5TB RAM and 128 processors with Intel(R) Xeon(R) Platinum 8358 CPU @2.60GHz and 2 NVIDIA A800 GPUs. Each experiment was conducted twice, averaging the results reported here.

7.2 Entity Resolution Results

We next report our findings.

Entity Blocking First we report the result of entity blocking. Table 3 report the top-1 recall and precision of all models, which is crucial in real-world RAG scenarios, since user may only want to check and use the first item without the willing of selecting from multiple options, while top-1 recall can be roughly regarded as the matching proportion of the first prediction.

It is evident that our method FUSER significantly surpasses all baselines in top-1 recall. For the datasets CO, SW, and SC, where entities predominantly consist of lengthy texts exceeding the representational capacities of the baseline models such as DeepBlocker and Sudowoodo, the embedding vectors generated by these methods fail to capture the implicit features of entities effectively, resulting in inadequate extraction of key information. Although the STransformer baseline employs the same contrastive learning methods, backbone

Table 3 The Performance of Entity Effectiveness with precision, recall and top-K. Here we fix $K = 1$ to evaluate top-1 retrieval ability of each blocking model.

Datasets	DeepBlocker [24]		Sudowoodo [25]		STransformer [5]		FUSER	
	Recall / Precision	K	Recall / Precision	K	Recall / Precision	K	Recall / Precision	K
Company(CO)	24.75 / 24.75	1	39.24 / 39.24	1	67.43 / 67.43	1	**78.67 / 78.67**	**1**
semi-text-c(SC)	0.20 / 0.47	1	0.92 / 2.13	1	12.40 / 28.36	1	**14.97 / 34.24**	**1**
semi-text-w(SW)	0.18 / 0.46	1	0.18 / 0.46	1	11.79 / 25.69	1	**16.15 / 35.18**	**1**
wdc-all-small(WS)	7.93 / 2.07	1	7.45 / 1.95	1	7.11 / 1.86	1	**32.39 / 8.47**	**1**
walmart-amazon(WA)	57.79 / 21.77	1	63.51 / 23.92	1	80.24 / 30.22	1	**82.01 / 30.89**	**1**
abt-buy(AB)	38.22 / 36.35	1	69.84 / 66.42	1	89.59 / 85.20	1	**94.06 / 89.45**	**1**

,

Table 4 The Performance of Entity Effectiveness with precision, recall and top-K. Following [28], we report the smallest recall/precision/K when the corresponding top-K recall achieves 90. If not, we will report recall/precision at top-20.

Datasets	DeepBlocker [24]		Sudowoodo [25]		STransformer [5]		FUSER	
	Recall / Precision	K	Recall / Precision	K	Recall / Precision	K	Recall / Precision	K
Company(CO)	47.85 / 2.39	20	57.39 / 4.46	20	83.25 / 4.16	20	**86.42 / 4.32**	**20**
semi-text-c(SC)	5.57 / 0.64	20	8.57 / 1.16	20	83.35 / 9.53	20	**89.31 / 10.22**	**20**
semi-text-w(SW)	1.92 / 0.24	20	5.23 / 0.67	20	64.93 / 7.07	20	**80.55 / 8.77**	**20**
wdc-all-small(WS)	55.00 / 0.72	20	53.04 / 0.92	20	57.39 / 0.65	20	**90.35 / 1.39**	**17**
walmart-amazon(WA)	90.12 / 3.39	20	90.54 / 8.53	4	86.38 / 1.63	20	**93.76 / 17.65**	**2**
abt-buy(AB)	75.19 / 3.57	20	90.37 / 28.73	3	74.32 / 3.53	20	**94.06 / 89.45**	**1**

model, and text input lengths as FUSER, its inability to leverage data enrichment for extracting effective attributes still fails to address '*needle in the haystack*' issue.

The conclusion for datasets WS, WA, and AB differs from previously discussed datasets. These datasets typically feature shorter texts, yet they include many specific abbreviations and terms, such as SKUs, which pose challenges for baseline models in segmentation and tokenization. This difficulty results in a misalignment between the left table T_l and the right table T_r. In contrast, FUSER mitigates this issue by unifying schema of T_l and T_r after enrichment and incorporating domain knowledge to interpret and populate the corresponding values, thereby significantly enhancing the retrieval quality of $\mathcal{M}_{\text{block}}$.

We additionally report the lowest recall, precision, and K values when the top-K recall reaches 0.9, as shown in Table 4. If this condition is not met, we will provide recall/precision for the top-20. Table 4 emphasizes the long-tail effect of the blocking model, assessing whether it can identify the correct entity within a specified budget limitation, *e.g.,* 20 in this scenario. It is evident that FUSER is the only method to achieve a recall above 0.8 within 20 candidates across all datasets. Additionally, FUSER is also the most cost-efficient blocking model, meaning it achieves the same performance with the fewest candidates K.

Entity Matching In Table 5, we report the main result of entity matching, comparing FUSER with different baseline models, containing PLMs and LLMs-based models.

Table 5 Entity matching performance in comparison to the baselines, n/a means the method cannot be terminated within 10 hours nor generate valid predictions. Unicorn and JellyFish are also pre-trained on various EM datasets for better performance, as discussed in their paper.

Datasets	FUSER			Ditto [2]			Rotom [3]			Unicorn [45]			PromptEM [4]			JellyFish [9]		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
WA	87.36	86.01	**86.68**	78.43	20.72	32.78	11.80	73.10	20.30	89.99	60.62	72.44	93.55	30.05	45.49	80.09	93.78	86.39
AB	87.44	91.26	**89.31**	97.24	51.45	67.30	14.60	42.70	21.70	97.11	49.02	65.16	98.04	48.54	64.94	99.38	78.15	87.50
CO	98.52	78.01	**87.08**	25.06	100.00	40.08	n/a	n/a	n/a	77.60	8.84	15.88	n/a	n/a	n/a	96.43	22.07	35.92
WS	91.55	94.99	**93.24**	71.89	20.28	31.64	27.4	75.00	40.2	95.97	43.73	60.09	91.55	28.05	42.94	96.64	52.92	68.39
SW	92.85	61.61	**74.07**	11.43	100.00	20.51	21.70	4.70	7.80	90.93	35.06	51.72	11.69	17.06	13.87	49.41	60.19	54.27
SC	81.85	71.67	**76.42**	13.85	100.00	24.34	23.40	16.20	19.20	99.99	39.33	56.46	17.00	13.30	14.92	77.79	74.43	76.08

When compared to PLM-based baseline solutions such as Ditto, Rotom, Unicorn, and PromptEM, FUSER consistently outperforms all baselines, achieving an improvement of over 20% in F1 score. These results underscore the significance of data enrichment and UQ selection techniques utilized by FUSER. In the few-shot setting, most baselines face challenges related to overfitting and unbalanced sample distribution. However, FUSER mitigates these issues by upsampling positive tuple pairs with structural data enrichment under UQ. This approach allows FUSER to upsample positive pairs in the training set $\mathcal{D}$, for example, *e.g.,* transforming 50 positive samples into 500 pairs remaining data distribution diversity. This addresses the unbalance data distribution issue in few-shot learning. Additionally, FUSER incorporate blocking model $\mathcal{M}_{\text{block}}$ to generate self-labeled hard negative samples. The above methods ensure that FUSER maintains leading performance across various scenarios.

Compared to LLM-based EM methods, such as JellyFish backboned with a 13B LLM model, FUSER shows an average F1-score improvement of over 10%, particularly in typical ULE-ER scenarios like the CO, SW, and WS datasets with 45% fewer parameter size and 20% lower training costs. These results underscore the significance of UQ-controlled data enrichment.

Regarding efficiency, as demonstrated in Figure 3, for ULE-ER process, JellyFish needs to extend the input window size to 4096 tokens for optimal performance, while FUSER can perform effectively with 2048 tokens for datasets CO, SW, and SC, and can reduce further to 1024 tokens for WA, AB, and WS datasets. We assert that FUSER strikes a superior balance between EM effectiveness and efficiency, demonstrating significantly higher data usage efficiency compared to other LLM-based EM methods.

7.3 Data Enrichment Results

In this section, we compare the data enrichment method in FUSER with existing data augmentation baselines for ER. Main result is listed in Table 6.

Since existing information retrieval method cannot be directly applied in our setting, we introduce the following data augmentation baselines, which are widely adopted among existing ER solutions.

- DK+DITTO: domain knowledge injection method applied by Ditto. Such method applies Named-Entity Recognition (NER) model Spacy [86] to identify and insert ex-

ternal attributes to the original model. Following Ditto [2], we set DK type to **Product**, which contains Product ID, Brand, Configurations (num.).

- Summarize+DITTO: summarize method applied by Ditto, based on TF-IDF summarize technology, which retains the non-stopword tokens with the high TF-IDF scores.
- InvDA + Rotom: generative data augmentation methods applied by Rotom. InvDA require to fine-tune a PLM model T5 [87] with $\mathcal{D}$ on each dataset, then inference such fine-tuned model to generate augmentation data.
- PTuning + PromptEM: prompt-tuning method applied by PromptEM. PromptEM incorporates a mixture of augmentation strategies for few-shot EM task, including prompt optimization, uncertainty-based pseudo-label sample generation and prompt-tuning technique for fine-tuning PLM.

To ensure a fair comparison, we substitute $\mathcal{M}_{\text{match}}$ in FUSER from Mistral-7B to PLM-based Unicorn, while preserving the enrichment results and self-labeled training dataset $\mathcal{D}_{\text{ER}}$. All methods in Table 6 utilize PLMs as backbone model with similar parameter size among all methods. We select three typical ULE-ER datasets: CO, SW and SC, highlight the necessity of data augmentation.

As presented in Table 6, the structural data enrichment method for FUSER achieve the leading performance, with more than 30% performance improvement in F1 score on average.

DK retrieves domain knowledge by pre-defined attribute and corresponding pattern, however it rely on pre-defined pattern to extract additional attributes, which lead to a high failure rate in unstructured text, where pattern may be vary in different entities. In not-defined domains, *e.g.,* CO dataset, DK may inject irrelevant information and result in 0.16 F1 performance drop. So DK cannot address "Lack of Domain Knowledge" issue well.

TF-IDF-based summarization was ineffective in the ULE-ER scenario. The primary issue is the "Needle in the Haystack" problem, where TF-IDF scores fail to accurately estimate useful information. Effective summarization necessitates a comprehensive understanding of the text, which TF-IDF cannot achieve. Although we observe a minor performance boost in summarization for DITTO, this method is not applicable in other contexts.

The generation-based method InvDA also falls short in the ULE-ER setting. InvDA requires a separate training of the generation model based on the training set $\mathcal{D}$, and its generation process is slower (taking an average of 4320s to generate 1000 samples, 30-50 times slower than FUSER). However, the contribution of its generated results to the improvement of EM performance is limited. This is because InvDA lacks structural restrictions and domain knowledge guidance. The underlying reason is that the generative capabilities of the PLM T5 are significantly lower than current LLMs.

PTuning focuses on improving EM performance at the prompt optimization level. However, we argue that the difficulty in the ULE-ER problem lies in effective information retrieval and extraction, rather than better methods of model training. This also proves that the structural data enrichment method of FUSER is independently viable, effectively enhances data quality at the data level.

7.4 Uncertainty Qualification Efficiency

In this section, we compare the UQ efficiency and their performance with 3 widely-adopted UQ baselines, namely Predictive Entropy(PE) [66], Semantic Entropy(SE) [62], and Shifting Attention Rele-

Table 6 Data enrich performance in comparison to baseline data augmentation methods in ER. All EM model are backboned with RoBERTa for fair comparison.

Datasets	FUSER + Unicorn			DK + DITTO			Summarize+ DITTO			InvDA+ Rotom			PTuning+ PromptEM		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
CO	93.41	77.87	**85.56**	92.00	13.18	24.12	73.88	35.23	47.70	41.70	3.60	6.60	28.79	2.62	4.81
SW	92.57	62.55	**65.83**	11.43	100.00	20.51	11.43	100.00	20.51	34.60	8.50	13.70	15.70	46.90	23.60
SC	90.75	66.14	**76.52**	13.85	100.00	24.34	13.85	100.00	24.34	74.20	20.90	32.60	40.80	29.00	33.90

Table 7 Uncertainty Qualification(UQ) performance in comparison to baseline UQ methods in EM task. We apply Unicorn as the EM method $\mathcal{M}_{\text{match}}$ for all methods.

Datasets	FUSER + Unicorn			PE + Unicorn			SE + Unicorn			SAR + Unicorn		
	P	R	F	P	R	F	P	R	F	P	R	F
AB	86.17	78.64	**82.23**	83.00	72.45	77.37	77.93	80.58	79.23	76.01	81.55	78.68
SW	92.57	62.55	**65.83**	56.88	60.66	58.71	48.42	65.40	55.65	51.96	62.55	56.77
SC	90.75	66.14	76.52	73.60	75.57	74.87	79.56	69.25	74.05	83.70	70.98	**76.82**

Table 8 Computational Cost for UQ of each tuple t on average, and k represents number of generations per entity.

Method	k	Generation	Logits	Semantic	Sum
PE [66]	5	1.36s	0.08s	0s	1.42s
SE [62]	5	1.36s	0.08s	0.22s	1.66s
SAR [63]	5	1.36s	0.08s	1.88s	3.32s
FUSER (1-GPU)	3.36	0.114s	0.0027s	0.013s	0.1297s
FUSER (2-GPU)	3.36	0.058s	0.0027s	0.007s	0.0684s

vance(SAR) [63]. Due to the limitaion of baseline methods, which can only estimate UQ on token-level and sentence-level, we uniformly apply baseline UQ methods on sentence-level, predicting 5 times over all entities $t \in T_l \cup T_r$, denoted as $G(g,t,R)$, and select the best result with the lowest UQ score. After generation and UQ selection, we organize the self-labeled training data $\mathcal{D}_{\text{ER}}$ and select Unicorn as the EM model. Among all UQ methods, the positive and negative pairs in $\mathcal{D}_{\text{ER}}$ is kept the same, while only the extracted SDE $\mathcal{S}(t)$ changes correspondingly. To perform a fair comparison, all baseline UQ methods are conducted with framework lm-polygraph [78], with the same LLM G as FUSER.

First, we report the UQ efficiency comparison as presented in Table 8, which details the average computational cost for each tuple on single A800 GPU. As discussed in Section 5.5, FUSER has successfully disentangled the LLM generation, logits computation, and the semantic relevance calculation procedures. For each of these procedures, FUSER employs a more efficient implementation method. Specifically, vLLM [75] is used for LLM generation and FlagEmbedding [84] for calculating semantic relevance. As a result, FUSER achieves a significant speedup, approximately 10 times faster than the baseline methods, with the capability to generate up to 1024 tokens per query on single GPU. Moreover, FUSER can effectively scale up through model parallelism. For example, FUSER can employ multiple GPUs to accelerate both the generation and the similarity relevance calculation stages.

Subsequently, we discuss the effectiveness of UQ as illustrated in Table 7. It is evident that FUSER also excels in EM tasks. This success is attributable to the pairwise enrichment strategy

Table 9 Ablation Study result in EM task. For w/o LLM ablation, we select Unicorn, which achieves the best performance among all non-LLM baselines.

Methods	CO			SW			AB			WS		
	P	R	F	P	R	F	P	R	F	P	R	F
FUSER	98.52	78.01	**87.08**	92.85	61.61	**74.07**	87.44	91.26	**89.31**	91.55	94.99	**93.24**
FUSER w/o Enrichment	25.84	59.80	36.09	49.41	60.19	54.27	81.18	79.61	80.39	73.77	99.08	84.57
FUSER w/o UQ	99.34	72.21	83.63	78.94	63.98	70.68	81.57	90.29	85.71	77.20	95.58	85.41
FUSER w/o LLM	93.41	77.87	85.56	92.57	62.55	65.83	86.17	78.64	82.23	88.06	80.47	84.09

employed by FUSER, which allows LLM to comprehend a wider range of contextual information. Additionally, FUSER's two-tier UQ solution effectively manages the quality of the generated content. These results demonstrate that FUSER strikes a balance between enrichment diversity and credibility, without significantly increasing computational overhead for UQ estimation.

7.5 Ablation Study

In this section, we apply ablation study to evaluate the effectiveness of each components in FUSER. The result is reported in Table 9.

For FUSER w/o Enrichment, it means directly use the ULE text to construct $\mathcal{D}_{\text{ER}}$ for training EM model; FUSER w/o UQ means randomly select attributes within $\mathcal{S}_t$ to construct training set; FUSER w/o LLM means replacing LLM-based EM models to Unicorn, while maintain $\mathcal{D}_{\text{ER}}$ generated with FUSER unchanged. We conduct the ablation study on 4 datasets, CO/SW/AB/WS, such order is arranged in descending order of the average number of tokens contained in each entity.

First of all, we can see enrichment in FUSER is most important, especially for LLM based EM methods. As discussed in Section 3.3, although LLM can takes the long text as input, it may "lost in the middle" [67] and pay attention to the wrong part. So FUSER w/o Enrichment suffers a worse performance compare to baseline Ditto with summarize.

FUSER w/o UQ also has an average of over 5% performance drop. Such reasons lies in hallucination issue for LLM, where LLM may generate factual error or irrelevant information. Although FUSER leverages SDE and domain knowledge guidance to constrain such issue, such hallucination still exists. We claim our two-tier UQ is a valid solution without much additional resource cost.

FUSER w/o LLM also has a significantly performance drop. Such phenomenon highlights the advantage of LLMs in understanding natural guidance g and related demonstration, mitigating the overfitting and unbalanced issue in few-shot learning. Nonetheless, we also claim that FUSER enhances data quality at the data level, and can be integrated with various existing EM methods with flexibility.

7.6 Hyperparameter Sensitivity

In this section, we primarily analyze the impact of two key hyperparameters: λ and the number of extended attributes $|R_{\text{extend}}|$, as illustrated in Figure 8,9.

Figure 8 demonstrates the effect in EM performance on the SC dataset as λ varies from 0 to 1. According to Eq.8, a smaller λ tends to favor attribute-level selection, whereas a larger λ leans

towards entity-level. Firstly, FUSER is not particularly sensitive to λ, which is reflected in the overall performance not showing significant fluctuations; secondly, since the entity-level UQ score is generally estimated based on the results of both structural similarity and attribute-level UQ, a bias towards entity-level, i.e., a larger λ value, would lead to better performance, although this disturbance is insignificant. Therefore, for robustness, in our experiments, we chose $\lambda = 0.8$.

Figure 9 shows the impact of extended attribute number on the results in WS dataset. It is evident that extended attributes improve the overall performance by more than 10%. However, only a limited number of attributes decisively impact the ER performance, and as $|R_{\text{extend}}|$ increases, LLM tends to generate null values or irrelevant information, thereby causing a decline in performance. Hence, in our experiments, we controlled $|R_{\text{extend}}|$ to 5, consistent with observations from existing ER-oriented data augmentation works [88].

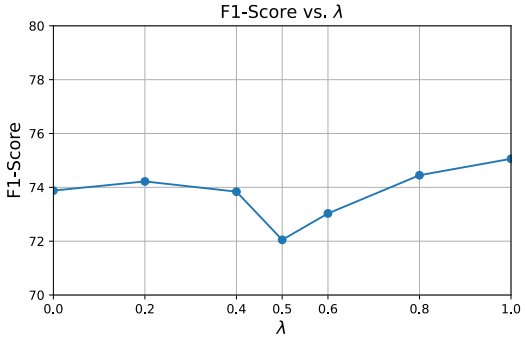

Fig. 8 Hyperparameter analysis varying λ in Eq.8 for dataset SC.

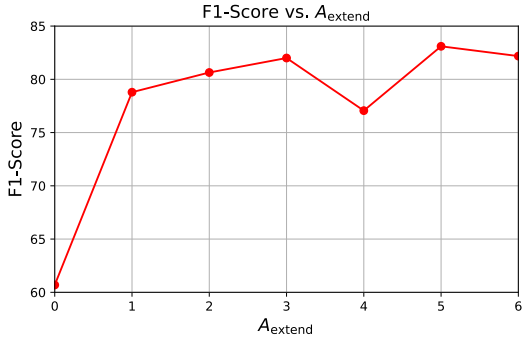

Fig. 9 Hyperparameter analysis varying the number of enriched attribute number $|R_{\text{extend}}|$ for dataset WS.

8 Conclusion

In this paper, we addressed the critical challenges posed by unstructured, long-text entities (ULE) in ER, such as input length constraints, the "Needle in the Haystack" problem, and insufficient domain knowledge, which hinder the performance of existing ER methods. To tackle these issues, we introduced FUSER, a Few-shot Uncertainty-calibrated Structural information Enrichment framework. FUSER leverages LLMs to extract and enrich structured data from unstructured entities and incorporates a two-tier uncertainty qualification module to enhance the reliability of generated attributes without additional inference cost. Our experimental results demonstrate that FUSER not only improves the data quality, but also maintains high performance in blocking and matching tasks with minimal labeled data. This work underscores the importance of SDE in ER, and establishes an efficient LLM-based methodology for handling few-shot ULE-ER problems effectively.

Acknowledgements This work was supported in part by the NSFC under Grants No.62225202, and Longhua Science and Technology Innovation Bureau 10162A20220720B12AB12. Jianxin Li is the corresponding author of this work.

References

1. Liu Y. Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692, 2019
2. Li Y, Li J, Suhara Y, Doan A, Tan W. Deep entity matching with pre-trained language models. PVLDB, 2020, 14(1): 50–60

3. Miao Z, Li Y, Wang X. Rotom: A meta-learned data augmentation framework for entity matching, data cleaning, text classification, and beyond. In: SIGMOD. 2021, 1303–1316
4. Wang P, Zeng X, Chen L, Ye F, Mao Y, Zhu J, Gao Y. Promptem: Prompt-tuning for low-resource generalized entity matching. PVLDB, 2022
5. Reimers N, Gurevych I. Sentence-bert: Sentence embeddings using siamese bert-networks. In: EMNLP. 11 2019
6. Mudgal S, Li H, Rekatsinas T, Doan A, Park Y, Krishnan G, Deep R, Arcaute E, Raghavendra V. Deep learning for entity matching: A design space exploration. In: SIGMOD. 2018, 19–34
7. Wang J, Li Y, Hirota W. Machamp: A generalized entity matching benchmark. In: CIKM. 2021
8. Chaudhury S, Dan S, Das P, Kollias G, Nelson E. Needle in the haystack for memory based large language models. arXiv preprint arXiv:2407.01437, 2024
9. Zhang H, Dong Y, Xiao C, Oyamada M. Jellyfish: A large language model for data preprocessing. arXiv preprint arXiv:2312.01678, 2023
10. Narayan A, Chami I, Orr L J, Ré C. Can foundation models wrangle your data? PVLDB, 2022, 16(4): 738–746
11. Cardie C. Empirical methods in information extraction. AI magazine, 1997, 18(4): 65–65
12. Wu H, Yuan Y, Mikaelyan L, Meulemans A, Liu X, Hensman J, Mitra B. Structured entity extraction using large language models. arXiv preprint arXiv:2402.04437, 2024
13. Huang L, Yu W, Ma W, Zhong W, Feng Z, Wang H, Chen Q, Peng W, Feng X, Qin B, others . A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. arXiv preprint arXiv:2311.05232, 2023
14. Papadakis G, Skoutas D, Thanos E, Palpanas T. Blocking and filtering techniques for entity resolution: A survey. ACM Comput. Surv., 2020, 53(2): 31:1–31:42
15. Fan W, Jia X, Li J, Ma S. Reasoning about record matching rules. PVLDB, 2009, 2(1): 407–418
16. Kejriwal M, Miranker D P. A DNF blocking scheme learner for heterogeneous datasets. CoRR, 2015, abs/1501.01694
17. Papadakis G, Koutrika G, Palpanas T, Nejdl W. Meta-blocking: Taking entity resolutionto the next level. IEEE Trans. Knowl. Data Eng., 2014
18. Papadakis G, Skoutas D, Thanos E, Palpanas T. A survey of blocking and filtering techniques for entity resolution. CoRR, 2019
19. Michelson M, Knoblock C A. Learning blocking schemes for record linkage. In: AAAI. 2006
20. Singh R, Meduri V V, Elmagarmid A K, Madden S, Papotti P, Quiané-Ruiz J, Solar-Lezama A, Tang N. Synthesizing entity matching rules by examples. Proc. VLDB Endow., 2017
21. Paulsen D, Govind Y, Doan A. Sparkly: A simple yet surprisingly strong TF/IDF blocker for entity matching. Proc. VLDB Endow., 2023
22. C. P S G, Ardalan A, Doan A, Akella A. Smurf: Self-service string matching using random forests. Proc. VLDB Endow., 2018
23. Efthymiou V, Papadakis G, Papastefanatos G, Stefanidis K, Palpanas T. Parallel meta-blocking: Realizing scalable entity resolution over large, heterogeneous data. In: IEEE BigData. 2015
24. Thirumuruganathan S, Li H, Tang N, Ouzzani M, Govind Y, Paulsen D, Fung G, Doan A. Deep learning for blocking in entity matching: A design space exploration. Proc. VLDB Endow., 2021, 14(11): 2459–2472
25. Wang R, Li Y, Wang J. Sudowoodo: Contrastive self-supervised learning for multi-purpose data integration and preparation. In: ICDE. 2023
26. Brinkmann A, Shraga R, Bizer C. Sc-block: Supervised contrastive blocking within entity resolution pipelines. In: ESWC. 2024
27. Wu S, Wu Q, Dong H, Hua W, Zhou X. Blocker and matcher can mutually benefit: A co-learning framework for low-resource entity resolution. Proc. VLDB Endow., 2023
28. Wang T, Lin H, Han X, Chen X, Cao B, Sun L. Towards universal dense blocking for entity resolution. CoRR, 2024, abs/2404.14831
29. Guo S, Dong X L, Srivastava D, Zajac R. Record linkage with uniqueness constraints and erroneous values. PVLDB, 2010, 3(1): 417–428
30. Fan W, Gao H, Jia X, Li J, Ma S. Dynamic constraints for record matching. VLDB J., 2011, 20(4): 495–520
31. Whang S E, Garcia-Molina H. Joint entity resolution on multiple datasets. VLDB J., 2013, 22(6): 773–795
32. Singh R, Meduri V V, Elmagarmid A K, Madden S, Papotti P, Quiané-Ruiz J, Solar-Lezama A, Tang N. Synthesizing entity matching rules by examples. PVLDB, 2017
33. Konda P, Das S, C. P S G, Doan A, Ardalan A, Ballard J R, Li H, Panahi F, Zhang H, Naughton J F, Prasad S, Krishnan G, Deep R, Raghavendra V. Magellan: Toward building entity matching management systems. PVLDB, 2016, 9(12): 1197–1208
34. Bilenko M, Mooney R J. Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the Ninth ACM SIGKDD International Confer-

ence on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003. 2003
35. Wu R, Chaba S, Sawlani S, Chu X, Thirumuruganathan S. ZeroER: Entity resolution using zero labeled examples. In: SIGMOD. 2020, 1149–1164
36. Ebraheem M, Thirumuruganathan S, Joty S R, Ouzzani M, Tang N. Distributed representations of tuples for entity resolution. PVLDB, 2018, 16(8): 1944–1957
37. Zhao C, He Y. Auto-EM: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In: WWW. 2019, 2413–2424
38. Li B, Wang W, Sun Y, Zhang L, Ali M A, Wang Y. Grapher: Token-centric entity resolution with graph convolutional neural networks. In: AAAI. 2020, 8172–8179
39. Fu C, Han X, Sun L, Chen B, Zhang W, Wu S, Kong H. End-to-end multi-perspective matching for entity resolution. In: IJCAI. 2019, 4961–4967
40. Zeng X, Wang P, Mao Y, Chen L, Liu X, Gao Y. Multiem: Efficient and effective unsupervised multi-table entity matching. In: 40th IEEE International Conference on Data Engineering, ICDE 2024, Utrecht, The Netherlands, May 13-16, 2024. 2024
41. Kirielle N, Christen P, Ranbaduge T. Transer: Homogeneous transfer learning for entity resolution. In: EDBT. 2022
42. Tu J, Han X, Fan J, Tang N, Chai C, Li G, Du X. DADER: hands-off entity resolution with domain adaptation. PVLDB, 2022, 15(12): 3666–3669
43. Sun C, Xu Y, Shen D, Nie T. Matching feature separation network for domain adaptation in entity matching. In: WWW. 2024, 1975–1985
44. Loster M, Koumarelas I K, Naumann F. Knowledge transfer for entity resolution with siamese neural networks. ACM J. Data Inf. Qual., 2021
45. Fan J, Tu J, Li G, Wang P, Du X, Jia X, Gao S, Tang N. Unicorn: A unified multi-tasking matching model. SIGMOD Rec., 2024
46. Li B, Miao Y, Wang Y, Sun Y, Wang W. Improving the efficiency and effectiveness for bert-based entity resolution. In: AAAI. 2021
47. Li P, He Y, Yashar D, Cui W, Ge S, Zhang H, Fainman D R, Zhang D, Chaudhuri S. Table-gpt: Table fine-tuned GPT for diverse table tasks. Proc. ACM Manag. Data, 2024
48. Wang T, Lin H, Chen X, Han X, Wang H, Zeng Z, Sun L. Match, compare, or select? an investigation of large language models for entity matching. CoRR, 2024, abs/2405.16884
49. Li H, Feng L, Li S, Hao F, Zhang C J, Song Y, Chen L. On leveraging large language models for enhancing entity resolution. CoRR, 2024
50. Gawlikowski J, Tassi C R N, Ali M, Lee J, Humt M, Feng J, Kruspe A, Triebel R, Jung P, Roscher R, others . A survey of uncertainty in deep neural networks. Artificial Intelligence Review, 2023, 56(Suppl 1): 1513–1589
51. Kendall A, Gal Y. What uncertainties do we need in bayesian deep learning for computer vision? Advances in neural information processing systems, 2017, 30
52. Kadavath S, Conerly T, Askell A, Henighan T, Drain D, Perez E, Schiefer N, Hatfield-Dodds Z, DasSarma N, Tran-Johnson E, others . Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221, 2022
53. Zhao X, Chen F, Hu S, Cho J H. Uncertainty aware semi-supervised learning on graph data. Advances in Neural Information Processing Systems, 2020, 33: 12827–12836
54. Brown T, Mann B, Ryder N, Subbiah M, Kaplan J D, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, others . Language models are few-shot learners. Advances in neural information processing systems, 2020, 33: 1877–1901
55. Touvron H, Lavril T, Izacard G, Martinet X, Lachaux M A, Lacroix T, Rozière B, Goyal N, Hambro E, Azhar F, others . Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023
56. Jiang A Q, Sablayrolles A, Mensch A, Bamford C, Chaplot D S, Las Casas d D, Bressand F, Lengyel G, Lample G, Saulnier L, Lavaud L R, Lachaux M, Stock P, Scao T L, Lavril T, Wang T, Lacroix T, Sayed W E. Mistral 7b. CoRR, 2023, abs/2310.06825
57. Xiao Y, Liang P P, Bhatt U, Neiswanger W, Salakhutdinov R, Morency L P. Uncertainty quantification with pre-trained language models: A large-scale empirical analysis. arXiv preprint arXiv:2210.04714, 2022
58. Lin S, Hilton J, Evans O. Teaching models to express their uncertainty in words. arXiv preprint arXiv:2205.14334, 2022
59. Manakul P, Liusie A, Gales M J. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. arXiv preprint arXiv:2303.08896, 2023
60. Malinin A, Gales M. Uncertainty estimation in autoregressive structured prediction. arXiv preprint arXiv:2002.07650, 2020
61. Huang Y, Song J, Wang Z, Zhao S, Chen H, Juefei-Xu F, Ma L. Look before you leap: An exploratory study of uncertainty measurement for large language models. arXiv preprint arXiv:2307.10236, 2023
62. Kuhn L, Gal Y, Farquhar S. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural

language generation. In: ICLR. 2023
63. Duan J, Cheng H, Wang S, Zavalny A, Wang C, Xu R, Kailkhura B, Xu K. Shifting attention to relevance: Towards the predictive uncertainty quantification of free-form large language models. In: ACL. 2024, 5050–5063
64. Yin Z, Sun Q, Guo Q, Wu J, Qiu X, Huang X. Do large language models know what they don't know? arXiv preprint arXiv:2305.18153, 2023
65. Sui Y, Zhou M, Zhou M, Han S, Zhang D. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In: Proceedings of the 17th ACM International Conference on Web Search and Data Mining. 2024, 645–654
66. Kadavath S, Conerly T, Askell A, Henighan T, Drain D, Perez E, Schiefer N, Hatfield-Dodds Z, DasSarma N, Tran-Johnson E, others . Language models (mostly) know what they know. arXiv preprint arXiv:2207.05221, 2022
67. An S, Ma Z, Lin Z, Zheng N, Lou J G. Make your llm fully utilize the context. arXiv preprint arXiv:2404.16811, 2024
68. Krell M M, Kosec M, Perez S P, Fitzgibbon A. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance, 2022
69. Luo K, Liu Z, Xiao S, Liu K. Bge landmark embedding: A chunking-free embedding method for retrieval augmented long-context large language models. arXiv preprint arXiv:2402.11573, 2024
70. Liu J. Llamaindex. https://github.com/jerryjliu/llama_index, 2023
71. Chen T, Kornblith S, Norouzi M, Hinton G. A simple framework for contrastive learning of visual representations, 2020
72. Kornbrot D. Point biserial correlation. Wiley StatsRef: Statistics Reference Online, 2014
73. Brinkmann A, Shraga R, Bizer C. Product attribute value extraction using large language models. arXiv preprint arXiv:2310.12537, 2023
74. lm-format-enforcer. https://github.com/noamgat/lm-format-enforcer, 2024
75. Kwon W, Li Z, Zhuang S, Sheng Y, Zheng L, Yu C H, Gonzalez J E, Zhang H, Stoica I. Efficient memory management for large language model serving with pagedattention. In: Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles. 2023
76. Farquhar S, Kossen J, Kuhn L, Gal Y. Detecting hallucinations in large language models using semantic entropy. Nature, 2024, 630(8017): 625–630
77. Leviathan Y, Kalman M, Matias Y. Fast inference from transformers via speculative decoding. In: International Conference on Machine Learning. 2023, 19274–19286
78. Fadeeva E, Vashurin R, Tsvigun A, Vazhentsev A, Petrakov S, Fedyanin K, Vasilev D, Goncharova E, Panchenko A, Panov M, others . Lm-polygraph: Uncertainty estimation for language models. arXiv preprint arXiv:2311.07383, 2023
79. Zheng Y, Zhang R, Zhang J, Ye Y, Luo Z, Feng Z, Ma Y. Llamafactory: Unified efficient fine-tuning of 100+ language models. In: ACL. 2024
80. Das S, Doan A, G. C. P S, Gokhale C, Konda P, Govind Y, Paulsen D. The magellan data repository. https://sites.google.com/site/anhaidgroup/projects/data, 2020
81. Primpeli A, Peeters R, Bizer C. The WDC training dataset and gold standard for large-scale product matching. In: Companion of The 2019 World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019. 2019
82. He P, Liu X, Gao J, Chen W. Deberta: Decoding-enhanced bert with disentangled attention. arXiv preprint arXiv:2006.03654, 2020
83. Zhang P, Xiao S, Liu Z, Dou Z, Nie J Y. Retrieve anything to augment large language models, 2023
84. Zhang P, Xiao S, Liu Z, Dou Z, Nie J. Retrieve anything to augment large language models. CoRR, 2023
85. Wang X, Wang Z, Gao X, Zhang F, Wu Y, Xu Z, Shi T, Wang Z, Li S, Qian Q, Yin R, Lv C, Zheng X, Huang X. Searching for best practices in retrieval-augmented generation, 2024
86. Honnibal M, Montani I, Van Landeghem S, Boyd A. spaCy: Industrial-strength Natural Language Processing in Python. 2020
87. Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou Y, Li W, Liu P J. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023
88. Yan M, Fan W, Wang Y, Xie M. Enriching relations with additional attributes for er. Proceedings of the VLDB Endowment, 2024, 17(11): 3109–3123

Mengyi Yan is currently working toward his PhD degree in the School of Computer Science and Engineering at Beihang University, China. His research interests include Large Language Models, Database and Data Mining.

Yaoshu Wang received the PhD degree in computer science from the University of New South Wales in 2018. He is currently a senior researcher in Shenzhen Institute of Computing Sciences. His research interests include Data Quality, Machine Learning and Big Data.

Xiaohan Jiang is currently working toward her PhD degree at the School of Computer Science and Engineering, Beihang University, China. Her research interests include Natural Language Processing, Time Series Analysis and Data Mining.

Haoyi Zhou received the PhD degree from the School of Computer Science and Engineering, Beihang University, China, in 2021. He is currently an Associate Professor with the School of Software, Beihang University, His research interests include Machine Learning and time-series analysis.

Jianxin Li is currently a professor with the School of Computer Science and Engineering, Beihang University, Beijing, China, and a senior researcher with Beijing Advanced Innovation Center for Big Data and Brain Computing. His current research interests consist of social networks, machine learning, Big Data, and trustworthy computing.