Front. Comput. Sci., 2025, 19(11): 1911376
https://doi.org/10.1007/s11704-025-41143-4

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, 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 Large Language Models (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 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.

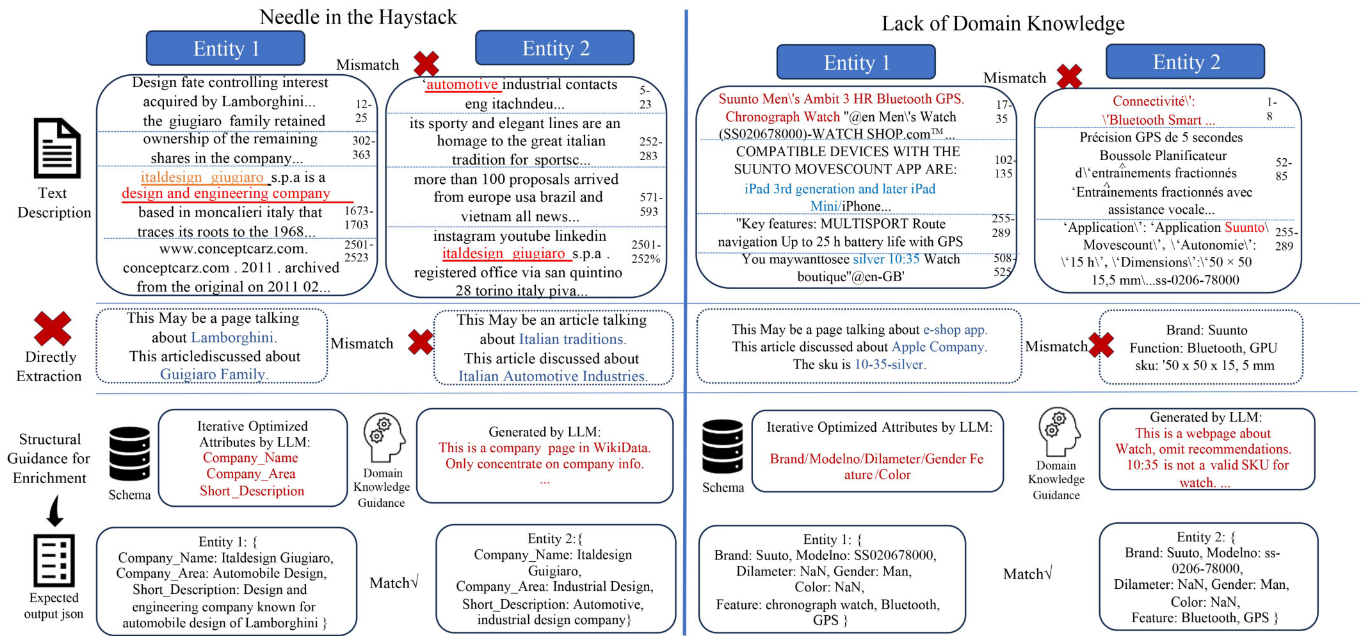
However, these methods often cost substantial human effort to label data as matches or mismatches, alleviating overfitting issue by expanding the training set. They are mostly designed to handle relational data with well-organized schemas, so that the feature alignment relations are learned, e.g., for 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 Fig. 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 Fig. 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, solving the Entity Resolution (ER) problem in real-world scenarios often involves **Unstructured** and **Long-text Entities** (ULE in short). Existing ER methods face significant challenges in addressing ULE due to the following reasons: (1) **Input Length Constraint:** Current PLM-based ER approaches limit the input length to 512 tokens, causing a decline in performance

Received October 24, 2024; accepted March 12, 2025

E-mail: lijx@buaa.edu.cn



Needle in the Haystack
Entity 1
Entity 2
Mismatch
Lack of Domain Knowledge
Text Description
Directly Extraction
Structural Guidance for Enrichment
Schema
Domain Knowledge Guidance
Expected output json
Match√
This May be a page talking about Lamborghini. This articlediscussed about Guigiaro Family.
This May be an article talking about Italian traditions. This article discussed about Italian Automotive Industries.
Iterative Optimized Attributes by LLM: Company_Name Company_Area Short_Description
Generated by LLM: This is a company page in WikiData. Only concentrate on company info. ...
Entity 1: { Company_Name: Italdesign Giugiaro, Company_Area: Automobile Design, Short_Description: Design and engineering company known for automobile design of Lamborghini }
Entity 2:{ Company_Name: Italdesign Guigiaro, Company_Area: Industrial Design, Short_Description: Automotive, industrial design company}
This May be a page talking about e-shop app. This article discussed about Apple Company. The sku is 10-35-silver.
Brand: Suunto Function: Bluetooth, GPU sku: '50 x 50 x 15, 5 mm
Iterative Optimized Attributes by LLM: Brand/Modelno/Dilameter/Gender Feature/Color
Generated by LLM: This is a webpage about Watch, omit recommendations. 10:35 is not a valid SKU for watch. ...
Entity 1: { Brand: Suuto, Modelno: SS020678000, Dilameter: NaN, Gender: Man, Color: NaN, Feature: chronograph watch, Bluetooth, GPS }
Entity 2: { Brand: Suuto, Modelno: ss-0206-78000, Dilameter: NaN, Gender: Man, Color: NaN, Feature: Bluetooth, GPS }

**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

as input text length increases. These methods often rely on truncation or summarization techniques, e.g., TF-IDF, which inevitably result in information loss and introduce errors. (2) **Needle in the Haystack [8]:** This indicates the difficulty of extracting small, valuable information (the "Needle") from extensive irrelevant text (the "Haystack"). Most ER techniques rely on schema information to align key features, but in ULE-ER scenarios, the absence of schema makes it challenging to extract and align relevant details from long, unstructured texts. (3) **Insufficient Domain Knowledge:** Domain-specific abbreviations and acronyms exacerbate the difficulty of ER tasks. A lack of domain understanding leads to models focusing on noise, while missing critical details.

These challenges make existing ER solutions highly dependent on large amounts of annotated data (e.g., over 20K labeled samples [2]) or large parameter-size Large Language Models (LLMs; e.g., 70B LLM [9]) to achieve competitive performance for ULE-ER in full-data training, incurring significant annotation and computational costs. To address these issues, it is crucial to minimize manual annotation through high-quality few-shot learning approaches for ER, which focus on improving the quality of unlabeled data. Numerous works [2,4,10] explore few-shot ER methods with minimal annotated tuple pairs.

The above challenges are not exclusive to ER but are also observed in related areas such as question answering [11], generative information Retrieval [12,13], and Retrieval-Augmented Generation (RAG) systems [14]. Managing long-text scenarios accurately remains a significant barrier for LLM in small parameter size.

It is worth noting that although several related works [15,16] have introduced LLMs into ER tasks, directly replacing 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 address the aforementioned challenges, a straightforward approach is to extract and enrich structured information from ULE, a process commonly referred to as information extraction [17]. As shown in Fig. 1, the objective of **Structural Data Enrichment** (SDE) is to extract and organize a set of structured attributes with a clear schema from ULE. This improves data quality and effectively reduces entity text length without significant information loss, facilitating ER tasks. Given the lack of annotated data for structured extraction, LLMs are employed to generate extraction results, as they have demonstrated effectiveness and robustness in similar tasks [18,19] through in-context learning ability [20]. However, LLMs are known to suffer from reliability issues, such as generating factual inaccuracies, commonly referred to as hallucinations [21]. As shown in the middle of Fig. 1 (Directly Extraction), directly querying LLMs for SDE is often infeasible. Without schema constraints and domain knowledge guidance, LLMs tend to generate incorrect attributes and values, leading to factual errors that degrade data quality. Moreover, LLMs are not efficient enough to handle large collections of tuples effectively.

To overcome these limitations, we propose a novel Few-shot Uncertainty-calibrated Structural Data Enrichment method for ER, abbreviated as FUSER. This method enhances the quality of SDE by employing unsupervised pairwise entity enrichment. FUSER leverages LLMs to extract well-organized structural attributes from unstructured entities, while simultaneously generating additional schema and domain knowledge guidance to act as constraints. To mitigate the risk of factual errors and hallucinations, FUSER incorporates a two-tier uncertainty-calibration module that evaluates and selects reliable enrichment solutions from multiple candidate outputs generated by LLMs, considering both attribute-level and entity-level uncertainty. Additionally, FUSER introduces a lightweight ER pipeline capable of efficiently performing both blocking and matching tasks with as few as 50 labeled positive samples. Experimental results demonstrate the superior performance of FUSER in both tasks, highlighting its effectiveness and efficiency in solving the ULE-ER problem.

Our contributions are summarized as follows:

- 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 entity-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 provides the definition of the ULE-ER problem, then discusses the challenge of directly applying LLMs for ULE-ER. Sections 4–6 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 [22]. We classify entity blocking into the following categories.

(1) Rule-based methods. There are methods that design handcrafted rules, e.g., Matching Dependencies (MD) [23], DNF [24], and meta blocking rules [25]; Besides, some works learns entity blocking rules with a few labeled training instances [26,27]. 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 [28,29] 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 [5,30–34] 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 [35–37] design logical rules to predict whether pairs of tuples are matched or not;

(2) Traditional ML models [10,38] 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 [2,6,39–42] 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] under low-resource setting [43], unsupervised learning [44], transfer learning [45–48], semi-supervised learning [4], and multi-task learning [49]. There are also works that integrate entity blocking into EM, e.g., [31,33,50] and adopt LLMs [9,15,16,51–53].

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 under few-shot labeled samples and small parameter-size open-source LLMs (e.g., $\leqslant$7B).

### 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 [54].

Although UQ has been well studied in the area with distinct labels, such as classification tasks in computer vision [55], NLP [56], graph learning [57], and knowledge graph [58] areas, the effective UQ for open-form LLMs, e.g., GPT series [59], LLaMA [60], Mistral [61] 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 [62] 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 [56,63] target at quantifying uncertainty by directly prompting the language model to generate uncertainty scores regarding to their own generations. SelfCheckGPT [64] 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. [65] estimates the free-form LLMs uncertainty level by calculating the accumulative predictive entropies over multiple generations. E$^2$CNN [66] incorporates entity-type information and cascaded neural networks to effectively handle triple overlapping and improve the precision of relation extraction in long sentences.

Among these methods, token-level UQ has proven efficient in hallucination detecting on various NLP tasks [67], 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) [68] 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 uncertainty measurement with DeBERTa-based classification model. Recent studies [69,70] 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

LLMs [59,60], which typically contain billions (or more) of parameters and pre-trained on massive text data [71], have demonstrated surprising emergent behaviors and good zero-shot generalization to new tasks. Most decoder-only LLMs, e.g., LLaMA [60] and Mistral [61], 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 output query with $N$ tokens, respectively; here, LLM has to generate output $\mathbf{s}$ regarding prompt $x$, where $\mathbf{s}$ is an output sentence $\mathbf{s}=\{z_1,\ldots,z_N\}$ containing $N$ tokens $z_i$ $(1\leqslant i\leqslant N)$. 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 $\{z_1,\ldots,z_{i-1}\}$ 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 (PE) [56], defined as the entropy over the whole sentence $\mathbf{s}$:

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

where $u(\mathbf{s})$ 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 $R_l=\{A^l_1,\ldots,A^l_k\}$ and $R_r=\{A^r_1,\ldots,A^r_k\}$, 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 attribute-value pairs $\{A^l_i,v^l_i\}$ (resp. $\{A^r_i,v^r_i\}$), where $v^l_i$ (resp. $v^r_i$) is the value of the $i$th attribute $A^l_i$ (resp. $A^r_i$) 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.

**Entity Blocking**. Most DL-based blocking methods follow the dense retrieval paradigm [30,31,34] 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 $T_r$.

**Entity 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). 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 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 Figs. 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 [12]. 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.

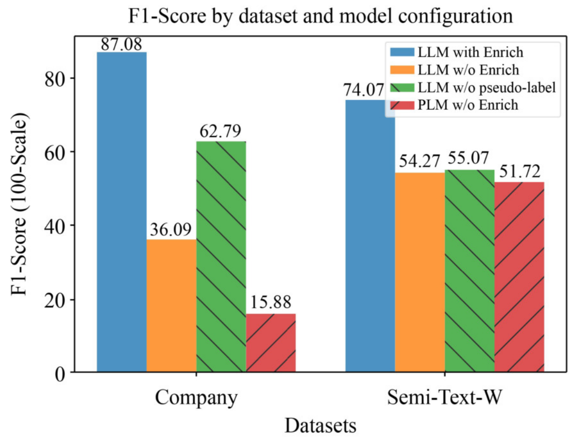


**Fig. 2** Entity matching performance of LLMs and PLM-based ER solutions

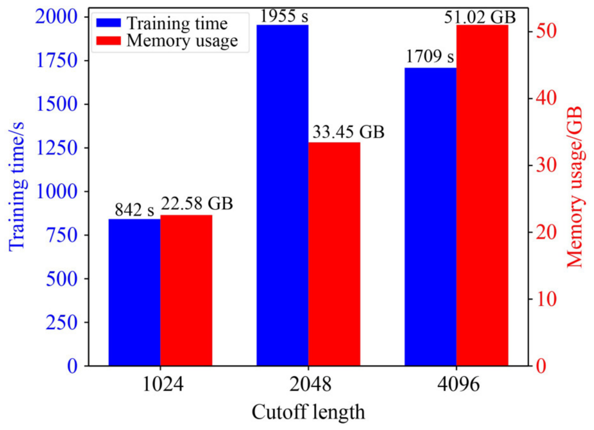


**Fig. 3** Training cost of LLMs in different maximum input token lengths. We apply the contamination-free packing [72] method to merge/pack short inputs into a single input at cutoff length, so the sample number in 4096 is fewer than that in 2048, leading to less training time

(c) High Training cost. When input length in-creases, it leads to a rapidly increased training cost 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 2789 and 345, up to 8685 and 3926, 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 Figs. 2 and 3; see more results in Section 7.5.

**Settings.** In Fig. 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 Fig. 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, caused by 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, leading to significant resource wastage.

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). Table 1 provides the main abbreviations, variables, and their corresponding descriptions used in this paper.

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)).

**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 |
| $R_{\text{extend}}$ | 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 $\mathbf{s}$ |

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 diversity of training set with pseudo-labeled positive/negative pairs, such to alleviate the overfitting issue (in response to challenge (b)).

## 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 Fig. 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 generates and selects additional enriched schema $R$ for $\mathbf{T}$, as well as domain-related prompt guidance $g$ 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) FUSER takes tuple pairs set $\mathbf{T}$, selected extra schema $R_{\text{extend}}$ and optimized domain knowledge set $g_{\mathbf{T}}$ as input, and generates structured enrichment output results $\mathbf{T_s}$ in schema $R_{\text{extend}}$ (Sections 4.3 and 4.4).

### 4.1 Tuple pairs selection with RAG

To address the aforementioned challenges in information retrieval task for unstructured long texts (ULE), we propose a series of optimization methods based on LLM generation

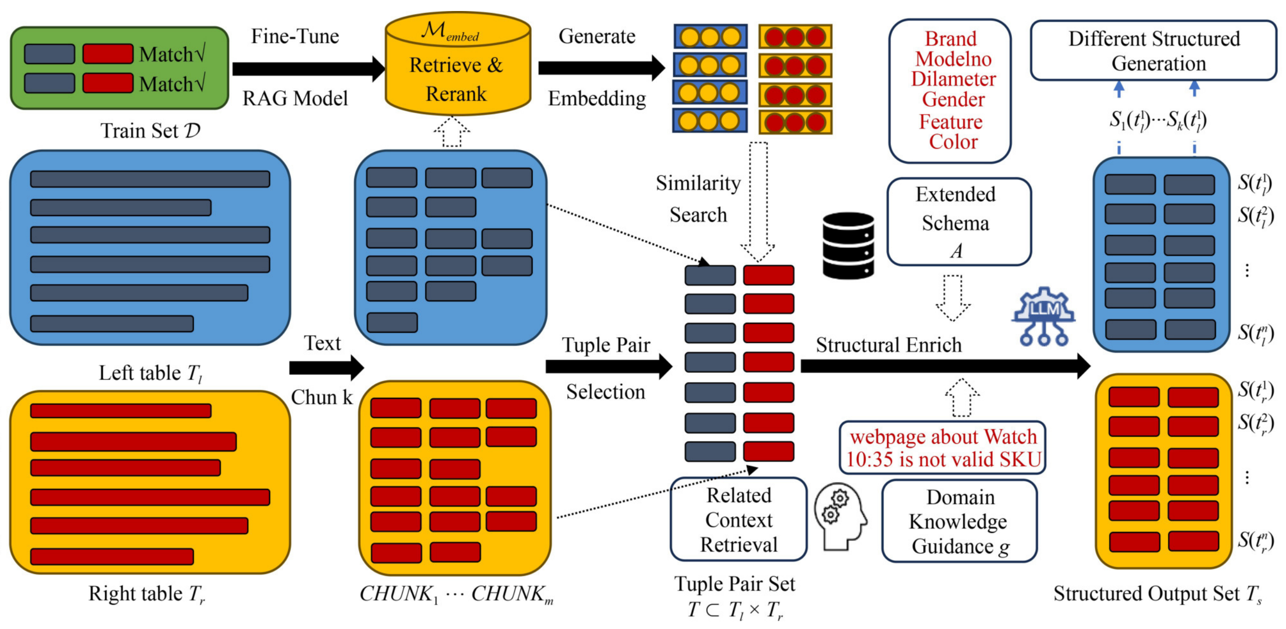

Match√
Match√
Train Set $\mathcal{D}$
Fine-Tune
RAG Model
$\mathcal{M}_{embed}$
Retrieve & Rerank
Generate
Embedding
Brand Modelno Dilameter Gender Feature Color
Different Structured Generation
$S_1(t_l^1)\cdots S_k(t_l^1)$
Similarity Search
Extended Schema $A$
$t_l^1$ $t_l^2$ ⋮ $t_l^n$
Left table $T_l$
Text Chun k
Tuple Pair Selection
Structural Enrich
LLM
$S(t_l^1)$ $S(t_l^2)$ ⋮ $S(t_l^n)$
$t_r^1$ $t_r^2$ ⋮ $t_r^n$
webpage about Watch 10:35 is not valid SKU
Related Context Retrieval
Domain Knowledge Guidance $g$
$S(t_r^1)$ $S(t_r^2)$ ⋮ $S(t_r^n)$
Right table $T_r$
$CHUNK_1 \cdots CHUNK_m$
Tuple Pair Set $T \subset T_l \times T_r$
Structured Output Set $T_s$

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

process, 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 Fig. 4 for a brief illustration.

**Text chunk.** Firstly, to tackle the "*Needle in the Haystack*" problem, we apply the recursive structure-aware chunking strategy [73,74], 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 $\text{CHUNK}_i$ for $i$th sliding window, covering several consistent structural segments within the input window size for embedding model $\mathcal{M}_{\text{embed}}$. Please check Fig. 5 for illustration.

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

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

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

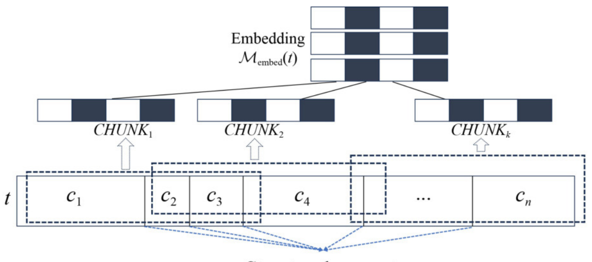
$$\text{CHUNK}_2 = \{c_{m+1}, \dots, c_{m+l_2}\}, \tag{3}$$

where the discard segments $\{c_1, \dots, c_m\}$ is approximately the step length, and the covered segments $\{c_{m+1}, \dots, c_{m+l_2}\}$ 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



Structural separator

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

negative samples. Subsequently, we fine-tune $\mathcal{M}_{\text{embed}}$ using contrastive loss [75] 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 $\{t_2^1, \dots, t_2^k\} \in T_r$, 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 list the prompt guidance and generated domain knowledge example in Figs. A1 and A2 at Appendix. Next 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 $\mathcal{V} = \{v_1, \dots, v_n\}$ 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 $R_{\text{all}}$, enhancing the ER tasks with labels $y$. Using LLM, we query tuple pairs from $\mathcal{D}_{\text{update}}$ to generate enriched values based on $R_{\text{all}}$ and measure the correlation between these values and $\mathcal{Y}$. We select the top 5 attributes with the highest correlation scores from $R_{\text{all}}$ to form the final extended schema $R_{\text{extend}}$, as these are most relevant to the EM task performance.

In detail, to evaluate the quality of extended attribute $R_{\text{extend}}$, for each tuple pair $(t_l, t_r, y) \in \mathcal{D}_{\text{update}}$ and attribute $v \in \mathcal{V}_A$ for attribute $R_{\text{extend}}$ 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 [76]:

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

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 $\{\mathrm{Sim}(v,v')|v' \in \mathcal{V}_A, v' \neq v\}$ within $\mathcal{D}_{\mathrm{update}}$, $n_0$ and $n_1$ are the counts of mismatched and matched instances, and $n = |\mathcal{D}_{\mathrm{update}}|$. The top 5 attributes in $R_{\mathrm{all}}$ with the highest PBC values are selected for the extended schema $R_{\mathrm{extend}}$, $R$ 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 [77], 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 xgrammar [78] 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-based [79] 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 s, 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.

In summary, in Section 5.1, we analyze the necessity of applying UQ to measure reliable outputs from 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 text chunking, pairwise context selection 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 Fig. 6 (follows the left example in Fig. 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 [68], SAR [69] 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 compared to other outputs will be

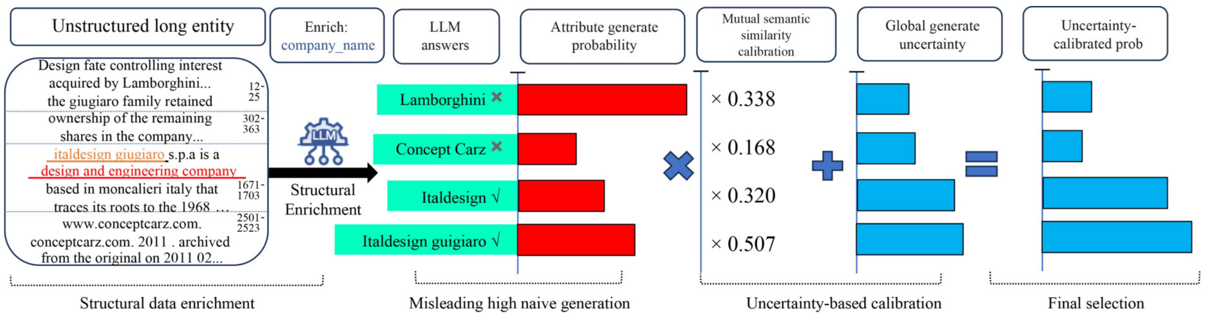

Unstructured long entity
Enrich: company_name
LLM answers
Attribute generate probability
Mutual semantic similarity calibration
Global generate uncertainty
Uncertainty-calibrated prob
Design fate controlling interest acquired by Lamborghini... the giugiaro family retained
12-25
ownership of the remaining shares in the company...
302-363
italdesign giugiaro s.p.a is a design and engineering company based in moncalieri italy that traces its roots to the 1968 ...
1671-1703
www.conceptcarz.com. conceptcarz.com. 2011 . archived from the original on 2011 02...
2501-2523
LLM
Structural Enrichment
Lamborghini ×
Concept Carz ×
Italdesign √
Italdesign guigiaro √
× 0.338
× 0.168
× 0.320
× 0.507
Structural data enrichment
Misleading high naive generation
Uncertainty-based calibration
Final selection

**Fig. 6** Illustration of uncertainty calibration component in FUSER, where the example at left corner is extracter from Fig. 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

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: (a) token-level selections are inappropriate for constrained structural outputs as they may disrupt the inherent structure of the data; (b) 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.

## 5.2 Attribute-level uncertainty qualification

As discussed in [80], 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 $\mathcal{V}_A(t) = \{v_A^1, \ldots, v_A^k\}$, 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:

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

where $p(v_A^i|g)$ is the generative probability of $v_A^i$, and $t$ is the temperature parameter used to control the relevance shifting scale. And the $\text{sim}(v_A^i, v_A^j)$ 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 noting that Eq. (5) shares a similar form with SE [68] and SAR [69], 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, however FUSER calculates $E_t$ 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 Fig. 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 $R = \{A_1, \ldots, A_n\}$. We mark $v_{A_j}^i$ as single enriched value for attribute $A_j$ in $S_i(t)$, while $\mathcal{V}_{A_j}(t)$ 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:

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

As presented in Fig. 7, if we expand all LLM-generated candidates $\mathcal{S}(t)$ 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 $S_i(t) \in \mathcal{S}(t)$:

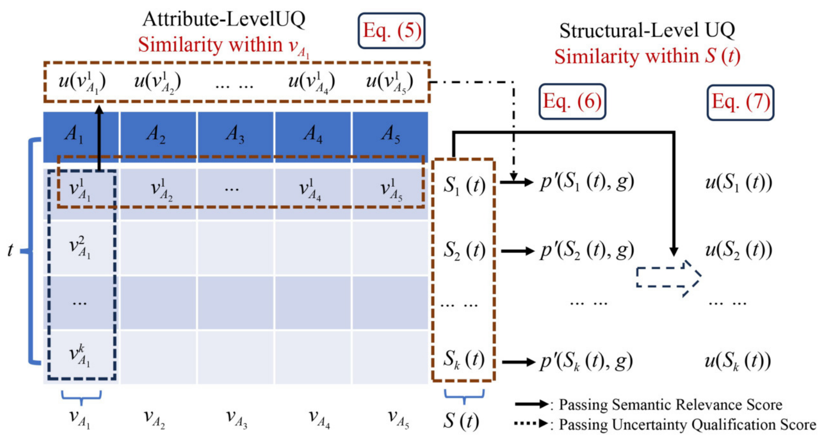

Attribute-LevelUQ
Similarity within $v_{A_1}$
Eq. (5)
Structural-Level UQ
Similarity within $S(t)$
Eq. (6)
Eq. (7)
$u(v_{A_1}^1)$ $u(v_{A_2}^1)$ … … $u(v_{A_4}^1)$ $u(v_{A_5}^1)$
$A_1$ $A_2$ $A_3$ $A_4$ $A_5$
$v_{A_1}^1$ $v_{A_2}^1$ … $v_{A_4}^1$ $v_{A_5}^1$
$v_{A_1}^2$
$v_{A_1}^k$
$t$
$S_1(t)$ $p'(S_1(t), g)$ $u(S_1(t))$
$S_2(t)$ $p'(S_2(t), g)$ $u(S_2(t))$
$S_k(t)$ $p'(S_k(t), g)$ $u(S_k(t))$
$v_{A_1}$ $v_{A_2}$ $v_{A_3}$ $v_{A_4}$ $v_{A_5}$ $S(t)$
: Passing Semantic Relevance Score
: Passing Uncertainty Qualification Score

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

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

Equation (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 Fig. 6 illustrate the entity-level uncertainty calibration. We highlight that calculating Eq. (7) only requires additional comparison across $S_i(t)\in\mathcal{S}_t$, 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 $\lambda$ 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 $\{U(v_A^1),\dots,U(v_A^k)\}$ for $k$ values $\{v_A^1,\dots,v_A^k\}\in\mathcal{V}_A$, we have two different sampling strategy to generate the final selection for $t[A]$.

One is greedy-based selection with certainty, marked as $\mathsf{Certain}(\mathcal{S}(t))$, 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 $\mathsf{Prob}(\mathcal{S}(t))$,where for $t[A]$, $\{v_A^1,\dots,v_A^k\}$ is selected by probability $\mathrm{Softmax}(-U'(v_A^1),\dots,-U'(v_A^k))$, 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 [81] 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 [69,82], 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 [82]. 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 group query 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 by vLLM with KV cache, 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 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 $i\neq 1,2$.

This upsampling strategy mitigates the risk of overfitting to a biased distribution from limited annotated samples, leveraging a weak label prior method [83]. Upon completing the training of $\mathcal{M}_{\text{block}}$ with contrastive learning loss, during inference, $\mathsf{Certain}(\mathcal{S}(t))$ is used to sample and generate the embedding vector for the tuple $t$, ensuring stable and reliable 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:

$$\mathrm{Serial}(t_1,t_2) = \{g,[S(t_1),S(t_2)],ICL[S(t_1),S(t_2)],y\}. \tag{9}$$

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 [84]. Similarly, during inference with trained $\mathcal{M}_{\text{match}}$, $\text{Certain}(\mathcal{S}(t))$ is applied to sample enriched 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 retrieval quality of data enrichment in FUSER? Can it outperform existing approaches based on text augmentation and summarization? Can it effectively tackle the *Needle in the Haystack* problem in ULE-ER?

*(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 compares the effectiveness of data enrichment module in FUSER with 4 data augmentation baselines for ER, while Section 7.6 evaluates FUSER's retrieval quality and computational efficiency, 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 7.7 provide ablation study and hyperparameter sensitivity analysis.

### 7.1 Experimental settings

We start with our settings.

**Datasets**. We conducted experiments on the following 6 benchmark datasets, as shown in Tables 2 and A1 of Appendix.

Company [38]. A benchmark dataset applied for Ditto [2]. The left and right tables are Wikipedia page and website homepage for certain company 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 [85]. A benchmark dataset from WDC dataset [85]. Both left and right tables in it are unstructured text description for certain product from different data sources.

Walmart-Amazon-Dirty [38]. A benchmark dataset applied processed from Ditto [2]. Both left and right table are unstructured description of product from different sources.

Abt-Buy-Dirty [38]. 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 and Appendix Table A1.

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

- DeepBlocker [30]: a transformer-based entity blocking model in the self-supervised strategy;
- Sudowoodo [31]: 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 method based on PLM RoBERTa [1], and uses various data perturbation,

**Table 2** Datasets used in our experiments, # means Number of, $T_l, T_r$ represents left/right table in ER; $|\mathcal{D}|$ means the number of few-shot labeled samples; 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. The train/validation/test split and dataset characteristics are kept the same with existing ER works [2,4]

| 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) [38] | 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) [85] | Product | 13,436 | 3,516 | Unstructured | Unstructured | 7,437-8,091 | 0.77% |
| Abt-Buy(AB) [38] | Product | 9,575 | 1,028 | Unstructured | Unstructured | 1,081-1,092 | 0.87% |
| Walmart-Amazon(WA) [38] | Product | 10,242 | 962 | Unstructured | Unstructured | 2,554-22,074 | 0.81% |

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 [49]: a PLM-based multi-task data matching framework with mixture-of-experts architecture, and pre-trained on various task-relafed corpus.
- JellyFish [15]: 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 [82]:

- PE [56]: predictive entropy method, which is defined as the entropy over the whole sentence, detailed in Eq. (1).
- SE [68]: 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 [86] model, predicting entailment relations.
- SAR [69]: 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.

For a fair comparison, all baselines are provided with the same set of 50 positive tuple pairs (denoted as $\mathcal{D}$), all labeled negative pairs from the training set of the benchmark, and the same relational left and right tables. In contrast, FUSER is provided with $\mathcal{D}$ and relational left and right tables but without the labeled negative pairs, representing few-shot setting. Notably, Unicorn and JellyFish were also pretrained on additional labeled entity matching corpus.

**Evaluation metrics** For entity blocking, following existing work [30], 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 $T_{\text{gt}}$ as 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 $T_{\text{predict}}$. 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 [61] as the backbone model of $\mathcal{M}_{\text{match}}$ and $G$, bge-rerank-large [87] as the pairwise similarity calculation model for UQ in Eqs. (5) and (7), bge-large-en [87] 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 [87] framework, and the LLM-based $\mathcal{M}_{\text{match}}$ with llama-factory [84].

Following existing UQ works [69], we set the top-$k$ sampling parameter for data enrichment as 5, relevance parameter $t$ in Eqs. (5) and (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 [73,88], we set the sliding window size as 512, and step size limit as 256 for text chunk for all datasets exclude CO. We apply 1024 sliding window size and 256 step size for CO. Following the findings in eliminating RAG redundancies [14,88], for certain extreme long entity, we calculate the cosine similarity among its text chunks using $\mathcal{M}_{\text{embed}}$, and filter the redundant chunks. We conduct our experiment on a single machine powered by 1.5TB RAM and 128 processors with Intel® Xeon® 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. As mentioned above, all methods are evaluated under the same few-shot setting.

**Entity blocking** First we report the result of entity blocking. Table 3 reports 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

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

| Datasets | DeepBlocker [30] | | Sudowoodo [31] | | 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** |

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 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 evaluates whether the blocking model 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.

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.

We observe that the performance gap between FUSER and the LLM baseline JellyFish widens as the text length and complexity increase. For example, in the CO dataset, due to the higher complexity of long texts, which means relevant information is typically hidden in the middle (illustrated in Fig. 8), directly using long texts for EM remains a challenging task for LLM. This results in a performance drop of over 50% for JellyFish compared to FUSER. However, for relatively shorter entities (e.g., WA/AB dataset), basic LLM demonstrates a strong understanding ability over unstructured data, with fewer instances of the *lost in the middle* issue that is typical of ULE. As a result, the performance of FUSER and JellyFish is comparable.

**Table 4** Performance of entity blocking with precision, recall and top-$K$. Following [34], 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 [30] | | Sudowoodo [31] | | 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** |

**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 [49] | | | PromptEM [4] | | | JellyFish [15] | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 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 |

However, regarding efficiency, as demonstrated in Fig. 3, for ULE-ER process, JellyFish requires to extend the input window size to 4096 tokens for optimal performance during training, 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 (DK) injection method applied by Ditto. Such method applies Named-Entity Recognition (NER) model Spacy [89] to identify and insert external 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 [90] 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 4,320 s to generate 1,000 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) [56], Semantic Entropy (SE) [68], and Shifting Attention Relevance (SAR) [69]. 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

**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 |

fair comparison, all baseline UQ methods are conducted with framework lm-polygraph [82], 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 [79] is used for LLM generation and FlagEmbedding [87] for calculating semantic relevance in batch. 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 be deployed on multiple GPUs to accelerate both the generation and the similarity relevance calculation procedure.

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 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 $\mathcal{D}_{\mathrm{ER}}$ 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 PLM-based Unicorn, while maintain $\mathcal{D}_{\mathrm{ER}}$ generated with FUSER unchanged. FUSER w/o pseudo-label means removing the hard negative sampling process in Section 6.1, only train the model with the initial few-shot positive samples in $\mathcal{D}_{\mathrm{ER}}$. We conduct the ablation study on 4 datasets, CO/SW/AB/WS.

First of all, we can see SDE 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" [12] 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.

FUSER w/o pseudo-label also has a siginficant

**Table 7** Uncertainty Qualification (UQ) performance in comparison to baseline UQ methods in EM task. We apply Unicorn as the EM method $\mathcal{M}_{\mathrm{match}}$ 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/s | Logits/s | Semantic/s | Sum/s |
|---|---|---|---|---|---|
| PE [56] | 5 | 1.36 | 0.08 | 0 | 1.42 |
| SE [68] | 5 | 1.36 | 0.08 | 0.22 | 1.66 |
| SAR [69] | 5 | 1.36 | 0.08 | 1.88 | 3.32 |
| FUSER(1-GPU) | 3.36 | 0.114 | 0.0027 | 0.013 | 0.1297 |
| FUSER(2-GPU) | 3.36 | 0.058 | 0.0027 | 0.007 | 0.0684 |

**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 |
| FUSER w/o pseudo-label | 99.61 | 45.85 | 62.79 | 42.65 | 77.74 | 55.07 | 43.16 | 96.60 | 59.67 | 26.80 | 8.34 | 12.73 |

performance drop. Such result demonstrate that LLM requires diversified training data to achieve comparable results. Even with correctly enriched data, LLM also tends to overfitting to biased distributions, and such observation is consistent with our findings in Section 3.3.

## 7.6 Analysis for retrieval quality

In this section, we primarily analyze the retrieval quality and computational efficiency of FUSER, i.e., tacking *Needle in the Haystack* issue with limited computational resources. Specifically, we evaluate whether the method can accurately locate and extract the correct information from unstructured long text for use in real-world ER scenarios. Figure 8 and Table 10 present the results of these experiments.

To effectively measure the aforementioned metrics, a critical step is to determine whether the attributes values extracted by the LLM correspond to the ground truth. However, such information is often missing in existing ER benchmark datasets. By querying metadata (Wikidata [91] for the CO dataset and WDC Product Data [85] for the SW and SC datasets), and combining manual annotation and verification, we identified the intersection between the metadata and the extended schema set $R_{\text{extend}}$, denoted as $R_{\text{gt}}$, to serve as the ground truth for verifying retrieval quality. Clearly, measuring the coverage of the LLM-generated structural enrichment results against the ground truth $R_{\text{gt}}$ (i.e., Retrieval Accuracy in Table 10) provides an effective evaluation of the model's performance. In this section we filter 5,000/1,374/3,292 entities for CO/SW/SC dataset correspondingly, which can be correctly mapped to metadata.

After obtaining the annotated data, we briefly present the characteristics of the aforementioned datasets. Figure 8 illustrates the relative position distribution of the ground truth attributes in long text at their first occurrence, i.e., the position of the *Needle* in the *Haystack*, across three typical ULE datasets: CO, SW, and SC. For each dataset, the critical information for more than half of the entities appears at 30%–70% of the whole context. This demonstrates that addressing the "lost-in-the-middle" issue of LLMs is a non-trivial and widespread challenge. Therefore, adopting a text-chunk-based RAG framework is necessary.

Next, we demonstrate through a set of ablation experiments how FUSER effectively improves information extraction accuracy via the RAG framework. In Table 10, FUSER w/o RAG represents a baseline where LLM directly generates extended schema and value from long text $t_l$, without the tuple pair selection, text chunking in Section 4.1, and the uncertainty qualification in Section 5. This baseline method significantly reduces the number of queries to the LLM (as shown by the Query Number row), but notably increases the query context length (see the Token Length row), making the extraction accuracy heavily reliant on the LLM's intrinsic capabilities. In contrast, FUSER, leveraging the RAG framework and uncertainty calibration, increases the query number to some extent but significantly reduces the presence of extremely long query contexts. All methods are conducted with the same backbone LLM model Mistral-7B with 2 A800 GPUs.

In Table 10, from the perspective of effectiveness, FUSER has a significantly higher retrieval accuracy than baseline, over 12% higher on average, and leads to over 4% higher F1 score in downstream ER tasks. Such improvement indicates FUSER has a higher retrieval accuracy for both unstructured and semi-structured entities.

From the perspective of efficiency, FUSER incorporates a typical RAG framework with a sliding window text chunk mechanism, resulting in more queries to handle long texts. However, since FUSER balances the text length distribution across queries, it effectively avoids the occurrence of extremely long texts compared to the baseline method, thereby

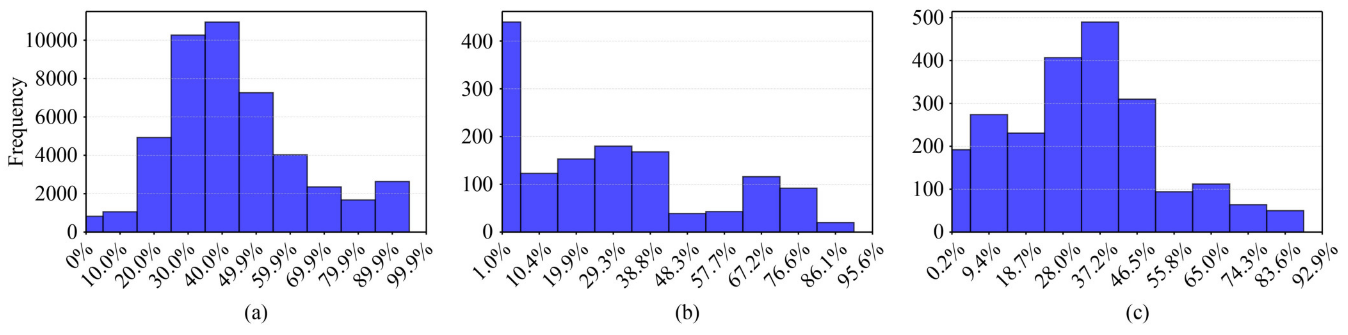


**Fig. 8** Illustration figure for the relative position distribution of the ground truth attributes in long text across three typical ULE datasets: CO, SW, and SC, i.e., the position of the *Needle* in the *Haystack*. The percentage on the $x$-axis represents the position of the *Needle* in their first occurrence. (a) Dataset CO, $R_{\text{gt}}$ = *Company-Name*; (b) dataset SW, $R_{\text{gt}}$ = *SKU/Brand*; (c) dataset SC, $R_{\text{gt}}$ = *SKU*

**Table 10** Retrieval quality evaluation for EM task. Retrieval accuracy, runtime, query number and token length are evaluated for SDE and UQ process, while F1-score is evaluated under downstream EM task with the same setting

| **Methods** | CO | | SW | | SC | |
|---|---|---|---|---|---|---|
| | FUSER w/o RAG | FUSER | FUSER w/o RAG | FUSER | FUSER w/o RAG | FUSER |
| Retrieval accuracy/% | 69.09 | **77.32** | 39.31 | **63.42** | 52.02 | **58.06** |
| RunTime/s | 541 | 389 (0.71×) | 58 | 85 (1.46×) | 83 | 117 (1.41×) |
| Query number | 4681 | 14039 (3×) | 1971 | 2712 (1.4×) | 3287 | 3660 (1.12×) |
| Token length (avg/max) | 2789/8685 | 595/1283 | 345/3926 | 360/1178 | 244/22504 | 412/3530 |
| F1-Score | 82.76 | **87.08** | 65.83 | **74.07** | 76.08 | **76.42** |

reducing computational complexity. Additionally, the similarity-based tuple pair selection enables FUSER to extensively reuse the query results' KV Cache, grouping similar queries for computation to accelerate inference speed.

A typical example is the CO dataset, where FUSER reduces the average text length by nearly 5 times and the maximum text length by 7 times compared to the original text. Although this introduces 2 times more queries, it reversely reduces the inference time by 30%. On the SC and SW datasets, where text lengths are relatively short, FUSER effectively handles extreme cases by chunking the few extremely long texts, resulting in a retrieval accuracy improvement of 24% and 6%, respectively, with affordable external computational cost.

We also claim that FUSER has the scalability to apply weak LLMs with fewer parameters without significant performance degradation. In Table 11, we compare the retrieval accuracy and EM performance of FUSER using smaller LLMs, with parameters ranging from 0.5B to 7B. We observe that, although smaller LLMs, such as Qwen2.5-0.5B, lead to worse retrieval quality, they can still extract distinguishable $R_{\text{extend}}$ from chunked text, leading to a moderate EM performance drop. As a result, smaller model does not significantly degrade the overall performance of the framework, demonstrating the potential for FUSER to be applied under low-resource conditions.

Regarding the model parameter size of LLMs, we also observe that for relative simple dataset, e.g., SC, small LLMs can retrieve high-quality structural enrichment for downstream ER task. However, in more complex scenarios, e.g., multilingual content in dataset SW and excluding irrelevant webpage content in dataset CO, LLM with larger parameter size performs consistently better.

## 7.7 Hyperparameter sensitivity

In this section, we primarily analyze the impact of two key hyperparameters: $\lambda$ and the number of extended attributes $|R_{\text{extend}}|$, as illustrated in Figs. 9 and 10.

Figure 9 demonstrates the effect in EM performance on the SC dataset as $\lambda$ varies from 0 to 1. According to Eq. (8), a smaller $\lambda$ tends to favor attribute-level selection, whereas a larger $\lambda$ leans towards entity-level. Firstly, FUSER is not particularly sensitive to $\lambda$, 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 $\lambda$ value, would lead to better performance, although this disturbance is insignificant. Therefore, for robustness, in our experiments, we chose $\lambda = 0.8$.

**Table 11** Retrieval Accuracy and Entity Matching performance of different LLM parameter size for schema enrichment. We report retrieval accuracy (Acc) and the F1-score (F1) for EM task

| Models | CO | | SW | | SC | |
|---|---|---|---|---|---|---|
| | Acc/% | F1/% | Acc/% | F1/% | Acc/% | F1/% |
| Qwen2.5-0.5B | 53.69 | 71.59 | 58.85 | 57.67 | 38.51 | 74.10 |
| Qwen2.5-1.5B | 67.90 | 82.40 | 60.27 | 71.79 | 51.29 | 75.55 |
| Qwen2.5-3B | **78.98** | 81.96 | 54.95 | 67.36 | 48.91 | **78.37** |
| Mistral-7B | 77.32 | **87.08** | **63.42** | **74.07** | **58.06** | 76.42 |

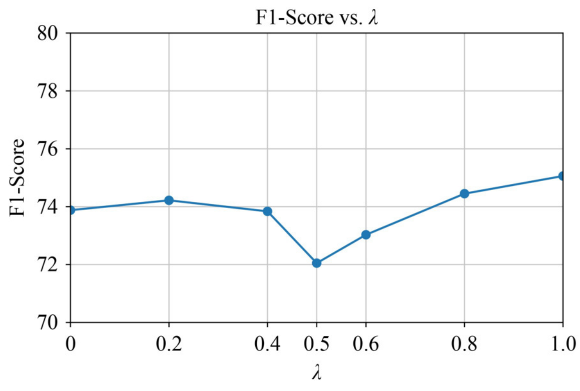


**Fig. 9** Hyperparameter analysis varying $\lambda$ in Eq. (8) for dataset SC

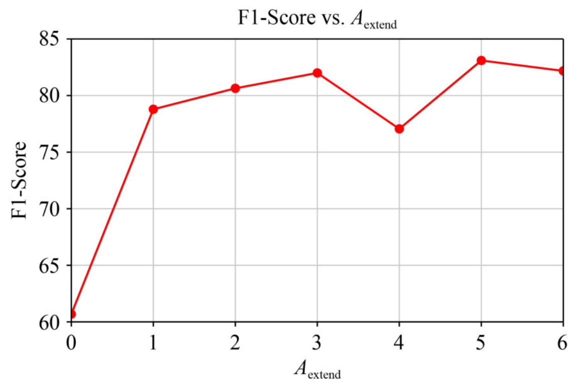


**Fig. 10** Hyperparameter analysis varying the number of enriched attribute number $|R_{\text{extend}}|$ for dataset WS

Figure 10 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 [92].

## 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 National Natural Science Foundation of China (Grant No. 62225202), and Longhua Science and Technology Innovation Bureau (Grant No. 10162A20220720B12AB12).

**Competing interests** The authors declare that they have no competing interests or financial conflicts to disclose.

## Appendixes

### Prompt template for domain knowledge guidance

In Figs. A1 and A2, we provide the prompt guiding LLM to generate domain knowledge $g_T$. As illistrated, for each query, we provide per-dataset **task description** from metadata, **instruction** to ask LLM generating multiple guidance as

**Instruction for generating domain knowledge $g_T$ for dataset CO**

**(system message)** You are an AI assistant that follows instruction extremely well. User will give you a question. Your task is to answer as faithfully as you can.
**(task description)** You are an expert in extracting information from dataset Company, which comes from webpage for a certain company. You are required to extract company name, company type, and short description from unstructured webpage content.
**(instruction)** Please write 3 guidance as prompts, which aims to help retrieving the above attributes, avoiding to rely on specific examples. The following is an example of the input webpage text:
**(input) Company 1**: {'content': 'and the outdoors type private founded 1977 headquarters columbia missouri key people larry potterfield founder $\cdots$ the company is headquartered in columbia missouri and sells in the continental united states . the company markets online . contents 1 history 2 modern management practices 3 nra support 4 midwayusa foundation 5 references$\cdots$'
**(Output)** Here are three guidance prompts for extracting values properly:
**Guidance 1: Identifying the Company Name 1. Identifying the Company Name**
Look for phrases or sections in the text that explicitly reference the company name. Indicators might include:

- The company's website URL .
- Phrases like "The company is known as...", "The name of the company is...", or "The business name was changed to...".
- The first occurrence of a proper noun or acronym consistently used throughout the text.

**Guidance 2. Determining the Company Type**
Scan for terms describing the business's ownership or organizational structure. These often follow keywords like:

- **"Type"** or phrases such as **"Privately/Publicly held"**.
- Descriptions including **"corporation," "LLC," "non-profit," "retailer,"** or other business entity terms.

**Guidance 3. Extracting the Short Description**
Look for concise summaries or introductions that encapsulate what the company does. These are typically found:

- Near the beginning of the content, often within the first few sentences of a paragraph.
- Following phrases like "is a..." or "is known as...".
- Summarizing its main activities, products, or services.

**Fig. A1** Prompts for LLM generating domain knowledge for dataset CO and examples of generation result $g_T$

**Instruction for generating domain knowledge $g_T$ for dataset SW**

**(task description)** You are an expert in extracting information from dataset semi-text-watch, which comes from webpage for a certain watch in different languages. You are required to extract title, brand, gender, sku, diameter from unstructured webpage content.
**(instruction)** Please write 3 guidance as prompts, which aims to help retrieving the above attributes, avoiding to rely on specific examples. The following is an example of the input webpage text:
**(input) Watch 1**: {'content': 'Seiko Scuba Divers Automatik Herrenuhrde Seiko Modell SKX007K2 Kollektion: Seiko Divers watch Scuba Diver´s Automatik Präzises Automatik Uhrwerk 7S26 mit 21 Lagersteinen'}
**Watch 2**: {'content': ' A finely crafted timepiece makes a bold statement of style $\cdots$ At Dejaun Jewelers, we are delighted to offer vast collection of some of the world's finest luxury watch brands, including Bvlgari. $\cdots$ We are also honored to be an authorized Bvlgari retailer in Los Angeles. Bvlgari's unique timepieces are crafted to last. This Ladies watch with model number BBLP33WGG 12 is part of the unique Bvlgari collection'}
**(Output)** Here are three guidance prompts for extracting values properly:
**Guidance 1. Multilingual Strategy**: Focus on proper nouns, common keywords for watch descriptors (e.g., "Automatik," "Watch"), and specific product identifiers.

- **Title:** Identify the main descriptive phrase combining brand, model, and key features. Look for prominent headers or bold text.
- **Brand:** Locate the manufacturer name, often the first word or frequently repeated in the text.

**Guidance 2. Extracting Gender, and SKU**

- **Gender:** Look for keywords such as "Men's," "Ladies," or equivalent gender indicators (e.g., "Herrenuhr").
- **SKU:** Extract unique alphanumeric product identifiers, often labeled "Model number" or "Modell."

**Guidance 3. Extracting Diameter**

- **Diameter:** Find case size measurements (e.g., "42 mm"), often near keywords like "diameter" or "Gehäusedurchmesser."

**Fig. A2** Prompts for LLM generating domain knowledge for dataset SW and examples of generation result $g_T$

**Table A1** Additional datasets characteristics in our experiments, as supplementary to Table 2

| Dataset | Domain | # All | # Match | # Query | # Token (w/o FUSER) | # Token (FUSER) | $\lvert R\rvert$ | $\lvert R_{\text{extend}}\rvert$ |
|---|---|---|---|---|---|---|---|---|
| Company(CO) [38] | Company | 112,632 | 28,200 | 66,550 | 2789/8685 | 595/1283 | 1/1 | 5/5 |
| Semi-Text-W(SW) [7] | Watch | 9,234 | 1,089 | 6,144 | 345/3926 | 360/1178 | 1.65/1 | 5/5 |
| Semi-Text-C(SC) [7] | Electronic | 20,897 | 2,940 | 7,936 | 244/22504 | 412/3530 | 1.62/1 | 5/5 |
| WDC-All-Small(WS) [85] | Product | 13,436 | 3,516 | 16,686 | 239/789 | 239/789 | 1/1 | 5/5 |
| Abt-Buy(AB) [38] | Product | 9,575 | 1,028 | 11,269 | 302/493 | 302/493 | 1/1 | 5/5 |
| Walmart-Amazon(WA) [38] | Product | 10,242 | 962 | 20,534 | 288/467 | 288/467 | 1/1 | 5/5 |

injected domain knowledge for the next structural data enrichment stage, and **input** for randomly-selected entity examples in the dataset. Due to space limitations, we only present the prompts for representative datasets. In the input, we display only a portion of the entity example content, with $\cdots$ indicating omitted content. **Output** shows the example outputs for $g_{\mathbf{T}}$, denoted as **guidance**.

After collecting multiple guidance from LLM, we select the top-3 guidance, acting as the global injected domain knowledge to conduct structural data enrichment process in Section 4.3. The selection criteria is the uncertainty score corresponding to each generated guidance, and the calculation method is presented at the end of Section 5.3.

Briefly speaking, the selection pipeline is highly similar to the proposed two-tier uncertainty qualification method in Section 5, in which we treat each individual generated guidance as *attribute*-level, and the whole generation output for each query as *entity*-level.

## Dataset characteristics

In Table A1 below, we provide additional datasets characteristics in our experiments, as supplementary to Table 2. # means Number of. # Query means query number for LLM over all dataset for FUSER after text chunk. We provide average(avg) and maximum(max) of token length for each query after text chunk. We also report existing attribute number $|R|$ for $T_l, T_r$ of each dataset, and enriched schema number $|R_{\text{extend}}|$ by FUSER. It is worth noting that $|R| = 1$ for unstructured data, and we report the average number of attributes for semi-structured data, as different entities may have different attributes.

## References

1. Liu Y. RoBERTa: a robustly optimized BERT pretraining approach. 2019, arXiv preprint arXiv: 1907.11692
2. Li Y, Li J, Suhara Y, Doan A, Tan W C. Deep entity matching with pre-trained language models. Proceedings of the VLDB Endowment, 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: Proceedings of 2021 International Conference on Management of Data. 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. Proceedings of the VLDB Endowment, 2022, 16(2): 369−378
5. Reimers N, Gurevych I. Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: Proceedings of 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing. 2019, 3982−3992
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: Proceedings of 2018 International Conference on Management of Data. 2018, 19−34
7. Wang J, Li Y, Hirota W. Machamp: a generalized entity matching benchmark. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. 2021, 4633−4642
8. Chaudhury S, Dan S, Das P, Kollias G, Nelson E. Needle in the haystack for memory based large language models. 2024, arXiv preprint arXiv: 2407.01437
9. Tang J, Dou W, Shen D, Nie T, Kou Y. Towards long-text entity resolution with chain-of-thought knowledge augmentation from large language models. In: Proceedings of the 29th International Conference on Database Systems for Advanced Applications. 2024, 322−336
10. Wu R, Chaba S, Sawlani S, Chu X, Thirumuruganathan S. ZeroER: entity resolution using zero labeled examples. In: Proceedings of 2020 ACM SIGMOD International Conference on Management of Data. 2020, 1149−1164
11. He J, Pan K, Dong X, Song Z, LiuYiBo L, Qianguosun Q, Liang Y, Wang H, Zhang E, Zhang J. Never lost in the middle: mastering long-context question answering with position-agnostic decompositional training. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics. 2024, 13628−13642
12. Liu N F, Lin K, Hewitt J, Paranjape A, Bevilacqua M, Petroni F, Liang P. Lost in the middle: how language models use long contexts. Transactions of the Association for Computational Linguistics, 2024, 12: 157−173
13. Xu D, Chen W, Peng W, Zhang C, Xu T, Zhao X, Wu X, Zheng Y, Wang Y, Chen E. Large language models for generative information extraction: a survey. Frontiers of Computer Science, 2024, 18(6): 186357
14. Singh I S, Aggarwal R, Allahverdiyev I, Taha M, Akalin A, Zhu K, O'Brien S. ChunkRAG: novel LLM-chunk filtering method for rag systems. 2024, arXiv preprint arXiv: 2410.19572
15. Zhang H, Dong Y, Xiao C, Oyamada M. Jellyfish: instruction-tuning local large language models for data preprocessing. In: Proceedings of 2024 Conference on Empirical Methods in Natural Language Processing. 2024, 8754−8782
16. Narayan A, Chami I, Orr L, Ré C. Can foundation models wrangle your data? Proceedings of the VLDB Endowment, 2022, 16(4): 738−746
17. Cardie C. Empirical methods in information extraction. AI Magazine, 1997, 18(4): 65−79
18. Wu H, Yuan Y, Mikaelyan L, Meulemans A, Liu X, Hensman J, Mitra B. Structured entity extraction using large language models. 2024, arXiv preprint arXiv: 2402.04437
19. Yang Y, Huang P, Cao J, Li J, Lin Y, Ma F. A prompt-based approach to adversarial example generation and robustness enhancement. Frontiers of Computer Science, 2024, 18(4): 184318
20. Wu Y, Yang X. A glance at in-context learning. Frontiers of Computer Science, 2024, 18(5): 185347
21. Huang L, Yu W, Ma W, Zhong W, Feng Z, Wang H, Chen Q, Peng W, Feng X, Qin B, Liu T. A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions. ACM Transactions on Information Systems, 2025, 43(2): 42
22. Papadakis G, Skoutas D, Thanos E, Palpanas T. Blocking and filtering

techniques for entity resolution: a survey. ACM Computing Surveys, 2021, 53(2): 31
23. Fan W, Jia X, Li J, Ma S. Reasoning about record matching rules. Proceedings of the VLDB Endowment, 2009, 2(1): 407−418
24. Kejriwal M, Miranker D P. A DNF blocking scheme learner for heterogeneous datasets. 2015, arXiv preprint arXiv: 1501.01694
25. Papadakis G, Koutrika G, Palpanas T, Nejdl W. Meta-blocking: taking entity resolutionto the next level. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(8): 1946−1960
26. Singh R, Meduri V V, Elmagarmid A, Madden S, Papotti P, Quiané-Ruiz J A, Solar-Lezama A, Tang N. Synthesizing entity matching rules by examples. Proceedings of the VLDB Endowment, 2017, 11(2): 189−202
27. Paulsen D, Govind Y, Doan A. Sparkly: a simple yet surprisingly strong TF/IDF blocker for entity matching. Proceedings of the VLDB Endowment, 2023, 16(6): 1507−1519
28. Paul Suganthan G C, Ardalan A, Doan A, Akella A. Smurf: self-service string matching using random forests. Proceedings of the VLDB Endowment, 2018, 12(3): 278−291
29. Efthymiou V, Papadakis G, Papastefanatos G, Stefanidis K, Palpanas T. Parallel meta-blocking: realizing scalable entity resolution over large, heterogeneous data. In: Proceedings of 2015 IEEE International Conference on Big Data (Big Data). 2015, 411−420
30. 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. Proceedings of the VLDB Endowment, 2021, 14(11): 2459−2472
31. Wang R, Li Y, Wang J. Sudowoodo: contrastive self-supervised learning for multi-purpose data integration and preparation. In: Proceedings of the 39th IEEE International Conference on Data Engineering. 2023, 1502−1515
32. Brinkmann A, Shraga R, Bizer C. SC-block: supervised contrastive blocking within entity resolution pipelines. In: Proceedings of the 21st International Conference on the Semantic Web. 2024, 121−142
33. 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. Proceedings of the VLDB Endowment, 2023, 17(3): 292−304
34. Wang T, Lin H, Han X, Chen X, Cao B, Sun L. Towards universal dense blocking for entity resolution. 2024, arXiv preprint arXiv: 2404.14831
35. Guo S, Dong X L, Srivastava D, Zajac R. Record linkage with uniqueness constraints and erroneous values. Proceedings of the VLDB Endowment, 2010, 3(1−2): 417−428
36. Fan W, Gao H, Jia X, Li J, Ma S. Dynamic constraints for record matching. The VLDB Journal, 2011, 20(4): 495−520
37. Whang S E, Garcia-Molina H. Joint entity resolution on multiple datasets. The VLDB Journal, 2013, 22(6): 773−795
38. Konda P, Das S, Paul Suganthan G C, Doan A, Ardalan A, Ballard J R, Li H, Panahi F, Zhang H, Naughton J, Prasad S, Krishnan G, Deep R, Raghavendra V. Magellan: toward building entity matching management systems. Proceedings of the VLDB Endowment, 2016, 9(12): 1197−1208
39. Ebraheem M, Thirumuruganathan S, Joty S, Ouzzani M, Tang N. Distributed representations of tuples for entity resolution. Proceedings of the VLDB Endowment, 2018, 11(11): 1454−1467
40. Zhao C, He Y. Auto-EM: end-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In: Proceedings of the World Wide Web Conference. 2019, 2413−2424
41. Li B, Wang W, Sun Y, Zhang L, Ali M A, Wang Y. GraphER: token-centric entity resolution with graph convolutional neural networks. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence. 2020, 8172−8179
42. Fu C, Han X, Sun L, Chen B, Zhang W, Wu S, Kong H. End-to-end multi-perspective matching for entity resolution. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence. 2019, 4961−4967
43. Tang J, Song R, Huang Y, Gao S, Yu Z. Semantic-aware entity alignment for low resource language knowledge graph. Frontiers of Computer Science, 2024, 18(4): 184319
44. Zeng X, Wang P, Mao Y, Chen L, Liu X, Gao Y. MultiEM: efficient and effective unsupervised multi-table entity matching. In: Proceedings of the 40th IEEE International Conference on Data Engineering. 2024, 3421−3434
45. Kirielle N, Christen P, Ranbaduge T. TransER: homogeneous transfer learning for entity resolution. In: Proceedings of the 25th International Conference on Extending Database Technology. 2022, 118−130
46. Tu J, Han X, Fan J, Tang N, Chai C, Li G, Du X. DADER: hands-off entity resolution with domain adaptation. Proceedings of the VLDB Endowment, 2022, 15(12): 3666−3669
47. Sun C, Xu Y, Shen D, Nie T. Matching feature separation network for domain adaptation in entity matching. In: Proceedings of the ACM Web Conference 2024. 2024, 1975−1985
48. Loster M, Koumarelas I, Naumann F. Knowledge transfer for entity resolution with Siamese neural networks. Journal of Data and Information Quality (JDIQ), 2021, 13(1): 2
49. Fan J, Tu J, Li G, Wang P, Du X, Jia X, Gao S, Tang N. Unicorn: a unified multi-tasking matching model. ACM SIGMOD Record, 2024, 53(1): 44−53
50. Li B, Miao Y, Wang Y, Sun Y, Wang W. Improving the efficiency and effectiveness for BERT-based entity resolution. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence. 2021, 13226−13233
51. Li P, He Y, Yashar D, Cui W, Ge S, Zhang H, Rifinski Fainman D, Zhang D, Chaudhuri S. Table-GPT: table fine-tuned GPT for diverse table tasks. Proceedings of the ACM on Management of Data, 2024, 2(3): 176
52. Wang T, Chen X, Lin H, Chen X, Han X, Sun L, Wang H, Zeng Z. Match, compare, or select? An investigation of large language models for entity matching. In: Proceedings of the 31st International Conference on Computational Linguistics. 2025, 96−109
53. Li H, Feng L, Li S, Hao F, Zhang C J, Song Y, Chen L. On leveraging large language models for enhancing entity resolution: a cost-efficient approach. 2024, arXiv preprint arXiv: 2401.03426
54. Gawlikowski J, Tassi C R N, Ali M, Lee J, Humt M, Feng J, Kruspe A, Triebel R, Jung P, Roscher R, Shahzad M, Yang W, Bamler R, Zhu X X. A survey of uncertainty in deep neural networks. Artificial Intelligence Review, 2023, 56(S1): 1513−1589
55. Kendall A, Gal Y. What uncertainties do we need in Bayesian deep learning for computer vision? In: Proceedings of the 31st International Conference on Neural Information Processing Systems. 2017, 5580−5590
56. Kadavath S, Conerly T, Askell A, Henighan T, Drain D, et al. Language models (mostly) know what they know. 2022, arXiv preprint arXiv: 2207.05221
57. Zhao X, Chen F, Hu S, Cho J H. Uncertainty aware semi-supervised learning on graph data. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 1076
58. Liu Q, Zhang Q, Zhao F, Wang G. Uncertain knowledge graph embedding: an effective method combining multi-relation and multi-path. Frontiers of Computer Science, 2024, 18(3): 183311
59. Brown T B, Mann B, Ryder N, Subbiah M, Kaplan J D, et al. Language models are few-shot learners. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 159
60. Touvron H, Lavril T, Izacard G, Martinet X, Lachaux M A, Lacroix T, Rozière B, Goyal N, Hambro E, Azhar F, Rodriguez A, Joulin A, Grave E, Lample G. LLaMA: open and efficient foundation language models. 2023, arXiv preprint arXiv: 2302.13971
61. Jiang A Q, Sablayrolles A, Mensch A, Bamford C, Chaplot D S, de las Casas D, Bressand F, Lengyel G, Lample G, Saulnier L, Lavaud L R, Lachaux M A, Stock P, Scao T L, Lavril T, Wang T, Lacroix T, Sayed W E. Mistral 7B. 2023, arXiv preprint arXiv: 2310.06825
62. 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. In: Proceedings of Findings of the Association for Computational Linguistics: EMNLP 2022. 2022, 7273−7284
63. Lin S, Hilton J, Evans O. Teaching models to express their uncertainty in words. Transactions on Machine Learning Research, 2022
64. Manakul P, Liusie A, Gales M. SelfCheckGPT: zero-resource black-box hallucination detection for generative large language models. In: Proceedings of 2023 Conference on Empirical Methods in Natural Language Processing. 2023, 9004−9017
65. Malinin A, Gales M. Uncertainty estimation in autoregressive structured prediction. In: Proceedings of the 9th International Conference on Learning Representations. 2021
66. Li M, Shi X, Qiao C, Huang X, Wang W, Wan Y, Zhang T, Jin H. E$^2$CNN: entity-type-enriched cascaded neural network for Chinese financial relation extraction. Frontiers of Computer Science, 2025, 19(10): 1910352
67. Huang Y, Song J, Wang Z, Zhao S, Chen H, Juefei-Xu F, Ma L. Look before you leap: an exploratory study of uncertainty analysis for large language models. IEEE Transactions on Software Engineering, 2025, 51(2): 413−429
68. Kuhn L, Gal Y, Farquhar S. Semantic uncertainty: linguistic invariances for uncertainty estimation in natural language generation. In: Proceedings of the 11th International Conference on Learning Representations. 2023
69. 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: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics. 2024, 5050−5063
70. Yin Z, Sun Q, Guo Q, Wu J, Qiu X, Huang X J. Do large language models know what they don't know? In: Proceedings of Findings of the Association for Computational Linguistics: ACL 2023. 2023, 8653−8665
71. 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
72. Krell M M, Kosec M, Perez S P, Fitzgibbon A. Efficient sequence packing without cross-contamination: accelerating large language models without impacting performance. 2022, arXiv preprint arXiv: 2107.02027
73. Luo K, Liu Z, Xiao S, Liu K. BGE landmark embedding: a chunking-free embedding method for retrieval augmented long-context large language models. 2024, arXiv preprint arXiv: 2402.11573
74. Liu J. Llamaindex. See GitHub repository (LlamaIndex) website, 2023
75. Chen T, Kornblith S, Norouzi M, Hinton G. A simple framework for contrastive learning of visual representations. In: Proceedings of the 37th International Conference on Machine Learning. 2020, 1597−1607
76. Kornbrot D. Point biserial correlation. In: Wiley StatsRef: Statistics Reference Online. New York: John Wiley & Sons, 2014
77. Brinkmann A, Shraga R, Bizer C. ExtractGPT: exploring the potential of large language models for product attribute value extraction. In: Proceedings of the 26th International Conference on Information Integration and Web Intelligence. 2025, 38−52
78. Dong Y, Ruan C F, Cai Y, Lai R, Xu Z, Zhao Y, Chen T. XGrammar: flexible and efficient structured generation engine for large language models. 2024, arXiv preprint arXiv: 2411.15100
79. Kwon W, Li Z, Zhuang S, Sheng Y, Zheng L, Yu C H, Gonzalez J, Zhang H, Stoica I. Efficient memory management for large language model serving with PagedAttention. In: Proceedings of the 29th Symposium on Operating Systems Principles. 2023, 611−626
80. Farquhar S, Kossen J, Kuhn L, Gal Y. Detecting hallucinations in large language models using semantic entropy. Nature, 2024, 630(8017): 625−630
81. Leviathan Y, Kalman M, Matias Y. Fast inference from transformers via speculative decoding. In: Proceedings of the 40th International Conference on Machine Learning. 2023, 19274−19286
82. Fadeeva E, Vashurin R, Tsvigun A, Vazhentsev A, Petrakov S, Fedyanin K, Vasilev D, Goncharova E, Panchenko A, Panov M, Baldwin T, Shelmanov A. LM-polygraph: uncertainty estimation for language models. In: Proceedings of 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. 2023, 446−461
83. Zhang J, Fan R, Tao H, Jiang J, Hou C. Constrained clustering with weak label prior. Frontiers of Computer Science, 2024, 18(3): 183338
84. Zheng Y, Zhang R, Zhang J, Ye Y, Luo Z, Feng Z, Ma Y. Llamafactory: unified efficient fine-tuning of 100+ language models. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations). 2024, 400−410
85. Primpeli A, Peeters R, Bizer C. The WDC training dataset and gold standard for large-scale product matching. In: Proceedings of the 2019 World Wide Web Conference. 2019, 381−386
86. He P, Liu X, Gao J, Chen W. DeBERTa: decoding-enhanced BERT with disentangled attention. In: Proceedings of the 9th International Conference on Learning Representations. 2021
87. Zhang P, Xiao S, Liu Z, Dou Z, Nie J Y. Retrieve anything to augment large language models. 2023, arXiv preprint arXiv: 2310.07554
88. 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. In: Proceedings of 2024 Conference on Empirical Methods in Natural Language Processing. 2024, 17716−17736
89. Honnibal M, Montani I, Van Landeghem S, Boyd A. spaCy: industrial-strength natural language processing in python. See github.com/explosion/spaCy website, 2020
90. 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. The Journal of Machine Learning Research, 2020, 21(1): 140
91. Vrandečić D, Krötzsch M. Wikidata: a free collaborative knowledgebase. Communications of the ACM, 2014, 57(10): 78−85
92. 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, Australia in 2018. He is currently a senior researcher in Shenzhen Institute of Computing Sciences, China. 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, China. 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, 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.