KcMF: A Knowledge-compliant Framework for Schema and Entity Matching with Fine-tuning-free LLMs

Anonymous ACL submission

Abstract

Schema and entity matching tasks are crucial for data integration and management. While large language models (LLMs) have shown promising results in these tasks, they suffer from hallucinations and confusion about task instructions. In this paper, we present the Knowledge-Compliant Matching Framework (KcMF), an LLM-based approach that addresses these issues without the need for domain-specific fine-tuning. KcMF employs a pseudo-code-based task decomposition strategy to adopt task-specific natural language statements that guide LLM reasoning and reduce confusion. We also propose two mechanisms, Dataset as Knowledge (DaK) and Example as Knowledge (EaK), to build domain knowledge sets when unstructured domain knowledge is lacking. Additionally, we introduce a result-ensembling strategy to leverage multiple knowledge sources and suppress poorly formatted outputs. Comprehensive evaluations on schema and entity matching tasks demonstrate that KcMF outperforms previous non-LLM state-of-the-art (SOTA) methods by an average F1 score of 22.9% and competes effectively with SOTA fine-tuned LLMs. Moreover, KcMF generalizes well across different LLMs.

1 Introduction

Schema matching is the task of identifying correspondences between elements of two or more database schemas. This task plays an important role in data integration efforts. Another task called entity matching, also known as entity resolution or record linkage, aims to identify schema instances that refer to the same real-world entity. While these two tasks vary in their definitions and approaches, they share the common goal of matching database elements. This paper attempts to tackle these tasks under the umbrella term *data matching*, employing a unified methodology while still retaining their original problem settings independently, i.e., entity matching is solved based on the database records, whereas schema matching is solved using metadata due to privacy considerations.

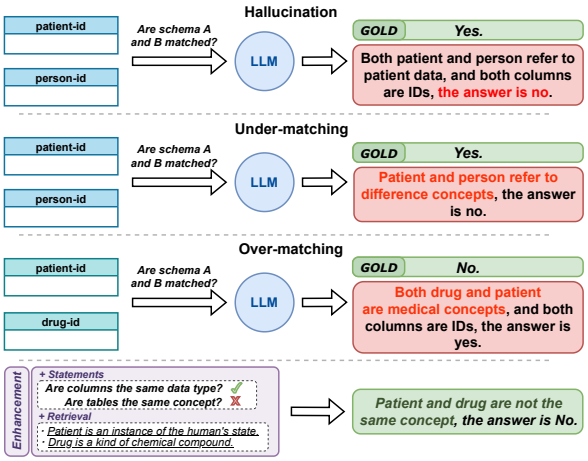

Figure 1: Three common issues in LLM-based data matching tasks and an overview of the enhancement will be discussed in this study: As demonstrated at the bottom, with pseudo-code and retrieved knowledge, by going through statements from the former, the LLM are able to reject the match between *patient-id* and *drug-id*.

Early systems for data matching tasks rely on expert systems or traditional machine learning methods, while modern data matching approaches often employ pre-trained language models (PLMs) as the backbone model (Li et al., 2020; Zhang et al., 2021; Zeakis et al., 2023), which is then fine-tuned on task-specific datasets. However, PLM-based methods may suffer from the effort collecting fine-tuning data and face performance degradation when handling unseen domain data during inference. Large Language Models (LLM) are PLMs of massive scales with billions of parameters, trained on vast and diverse datasets. With huge internal knowledge gained from pre-training and strong representation capability, LLMs show competitive performance on numerous data-wrangling benchmarks, particularly without the need for fine-tuning (Narayan et al., 2022; Peeters and Bizer,

2023; Sheetrit et al., 2024; Peeters and Bizer, 2024). This remarkable efficiency liberates users from the burden of fine-tuning efforts.

Despite promising results, LLM-based approaches to data matching face several performance issues. For clarity of presentation, we focus mainly on schema matching in the main text; further discussions of entity matching are provided in Appendix A.1. As depicted in Figure 1, these issues are categorized into three types: (1) *hallucination*, where the LLM generates incorrect deduction from correct evidence at hand; (2) *under-matching*, where the LLM rejects matching with overly strict criteria; and (3) *over-matching*, where the LLM over-generalizes the association of the input data. While hallucination is a widely recognized problem that can harm the LLM's performance in many other tasks (Zhang et al., 2023a; Rawte et al., 2023), under-matching and over-matching, jointly termed as *confusion* in this paper, are challenges specific to the data matching tasks.

In this study, we present the Knowledge-compliant Matching Framework (KCMF), a fine-tuning-free and retrieval-enhanced approach to data matching tasks. To address the problem of confusion, KCMF employs a *pseudo-code-based task decomposition strategy* for LLMs. Pseudo-code here is an ordered list of task-specific conditional statements, written in natural language, that guide the matching process. The LLM can easily evaluate the validity of each condition for the data being matched, and then follow the corresponding directives. By walking through the pseudo-code, the LLM is able to reason its way to the final matching result. Unlike Chain-of-Thought (Wei et al., 2022), which relies on LLM's internal knowledge to generate reasoning steps, our KCMF uses explicit task-specific pseudo-code, eliminating the need to write reasoning chains for different datasets.

To address hallucination, KCMF incorporates external knowledge in the form of natural-language sentences related to the input data. To compensate for the lack of such unstructured domain knowledge, KCMF builds domain knowledge sets by leveraging readily available datasets (Dataset as Knowledge, DaK) and examples (Examples as Knowledge, EaK) from various domains. Since fine-tuning-free LLMs tend to generate improperly formatted outputs that do not match the format given in the demonstration, we employ a technique called *Inconsistency-tolerant Generation Ensembling* (IntGE) to suppress such unexpected outputs and maintain an automated downstream workflow.

We evaluate KCMF on several benchmarks for knowledge-intensive matching tasks, namely MIMIC and Synthea from the OMAP project (Zhang et al., 2021), and MMM, modified from MedMentions (Mohan and Li, 2019). The results show that KCMF surpasses 1) all LLM baselines, 2) SMAT (Zhang et al., 2021), a non-LLM method, and 3) Jellyfish (Zhang et al., 2024a), a fine-tuned LLM-based state-of-the-art (SOTA). Our main contributions are as follows.

1. We introduce KCMF, an LLM-based framework that incorporates a unified methodology to solve schema matching and entity matching tasks, eliminating the need for task-dependent model design and domain-specific fine-tuning.

2. We propose 1) a pseudo-code-based task decomposition strategy to guide the LLM's reasoning by obtaining task-specific conditional statements, reducing *confusion* during matching tasks, 2) two mechanisms, namely DaK and EaK, to build knowledge sets in lack of unstructured domain knowledge, and 3) IntGE, a result-ensembling strategy designed to leverage multiple knowledge sources and suppress poorly formatted outputs.

3. We perform a comprehensive evaluation for KCMF and demonstrate its superiority against various popular data matching methods. Our analysis also shows that KCMF can generalize effectively across different LLMs.

2 Background and Task Definition

2.1 Schema Matching (SM)

Conventional SM approaches can generally be categorized into 1) constraint-based methods, which utilize attributes defined in database constraints to measure similarity among schemas (Alexe et al., 2010; Chen et al., 2018; Atzeni et al., 2019), and 2) linguistic-based methods, which leverage the semantic information contained in schema names or descriptions to construct mappings (Kettouch et al., 2017; Asif-Ur-Rahman et al., 2023). More recent studies have adopted deep neural networks to tackle SM. (Zhang et al., 2021) use an attention-based BiLSTM with pre-trained word embeddings. (Zhang et al., 2023b) leverage BERT (Kenton and Toutanova, 2019) to generate schema features, which are then used to train a linear classifier under semi-supervised learning. With the advent of

LLMs, (Narayan et al., 2022) was the first to apply LLMs to SM, using straightforward serialization with few-shot settings. Recent LLM-based SM approaches, such as (Zhang et al., 2024a), attempt to tackle this task using Supervised Fine-tuning (SFT). While SFT achieves strong results, it needs tremendous efforts to collect fine-tuning data and is hard to transfer to unseen domains. Another line of works, such as (Sheetrit et al., 2024) and (Parciak et al., 2024), utilize proprietary LLMs, which have shown promising results on several benchmarks. However, these approaches still suffer from high computational costs and produce indecisive outputs. Our approach aims to extend the scope of LLM-based SM by utilizing task-specific pseudo-code to guide LLMs' predictions within a single-round inference. The designed pseudo-code offers explicit criteria for the matching task, addressing the *confusion* problem mentioned earlier.

2.2 Entity Matching (EM)

EM is often considered a downstream task of SM by many previous works (Barlaug and Gulla, 2021). Similar to SM, traditional non-LLM solutions for EM focus on computing similarity between entities (Thirumuruganathan et al., 2018; Ebraheem et al., 2018; Kasai et al., 2019; Li et al., 2020). Following the initial application of LLM to EM (Narayan et al., 2022), LLM-based solutions have been actively explored (Peeters and Bizer, 2023; Fan et al., 2024; Peeters and Bizer, 2024; Li et al., 2024). From the outset, our approach is designed as a unified framework capable of addressing both SM and EM. By recognizing the deep semantic similarities and shared motivations between SM and EM — such as aligning and matching data elements — we develop a methodology that seamlessly integrates both tasks.

2.3 Retrieval Augmented Generation (RAG)

RAG incorporates retrieved documents into queries and has become a popular paradigm in mitigating hallucination in LLMs (Lewis et al., 2020; Yu et al., 2023; Shao et al., 2023; Jiang et al., 2023b; Asai et al., 2023; Xu et al., 2023; Shi et al., 2024). While RAG has shown reliable results in addressing hallucination, its performance is still limited by the quality of the retrieved documents and, obviously, is challenging to deploy in scenarios lacking unstructured knowledge sources. Our approach adopts the concept of introducing external knowledge to alleviate hallucination. In particular, to address the lack of unstructured domain knowledge in data matching, we present two mechanisms, DaK and EaK, for building knowledge sets by utilizing existing domain knowledge bases. Our approach showcases the potential of external knowledge enhancement strategies on LLM-based classification tasks.

2.4 Task Definition

We denote by $\mathcal{S}$ the *source schema* from database $\mathcal{D}$, and $\mathcal{S}'$ the *target schema* from database $\mathcal{D}'$. The goal of **schema matching** is to identify all pairs of **attributes** $(\mathcal{A}, \mathcal{A}')$ such that $\mathcal{A} \in \mathcal{S}$ and $\mathcal{A}' \in \mathcal{S}'$, and both attributes represent the same information in their respective schemas.

In this paper, we focus on a more straightforward scenario: we enumerate all possible mappings $\mathcal{M} \subseteq \mathcal{D} \times \mathcal{D}'$ and determine whether each mapping $\{r, r'\} \in \mathcal{M}$ is correct or not. Each candidate mapping $\{r, r'\}$ consists of two items, where each item r is composed of a schema name N and a schema description C from the corresponding database $\mathcal{D}$.

We framework our schema matching task as an LLM generation task guided by pseudo-code instructions under a knowledge-enhanced setting. This involves 1) a list $\mathcal{P}$ of designated task-specific pseudo-code and 2) a list K of knowledge items retrieved from all available knowledge sets $\mathcal{K}$.

We serialize the inputs, including the pseudo-code $\mathcal{P}$, the candidate mapping $\{r, r'\}$, and the retrieved knowledge items K, into a prompt suitable for the LLM using a serialization function ϕ. The LLM $\mathcal{L}$ then generates a response LR based on this prompt:

$$LR \leftarrow \mathcal{L}\big(\phi(\mathcal{P}, \{r, r'\}, K)\big). \quad (1)$$

Our task objective is to classify the correctness of each candidate mapping. Specifically, we aim to obtain a binary classification result $c \in \{yes, no\}$ from the LLM's response LR, indicating whether the mapping $\{r, r'\}$ is correct. For clarity of presentation, here we focus on defining SM, though it should be noticed that this setting can also be generalized to entity matching task (see A.1 for details).

3 Matching Framework

3.1 Framework Overview

Figure 2 presents an overview of our proposed KCMF, which operates in four sequential stages as follows. *S1. Pseudo-code Design*: Experts decompose the task into task-related conditional statements in natural language. A superior advantage of

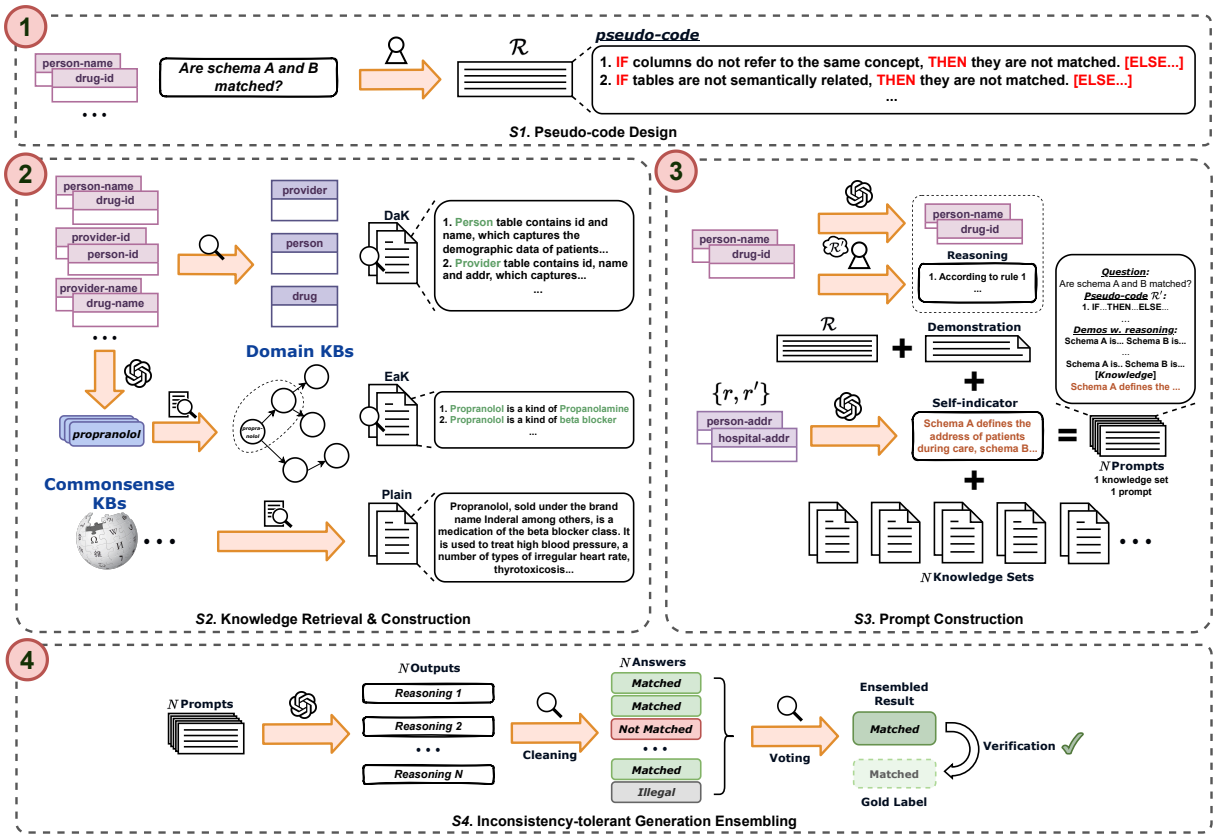

Figure 2: Overview of the Knowledge-compliant Matching Framework (KCMF). The pseudo-code we have designed (see A.4.1 for details) is reusable for SM and EM tasks, and there is no need to design from scratch when facing new datasets.

our approach is that this pseudo-code for a task is designed once and can be reused by anyone performing the matching task, without the need to write custom ones. *S2. Knowledge Retrieval & Construction*: KCMF constructs granular domain knowledge and database-structure knowledge via retrieving information from domain knowledge bases and discovering database structures. *S3. Prompt Generation*: Utilizing the knowledge from S2 and pseudo-code from S1, KCMF generates LLM prompts that include demonstrations. *S4. Inconsistency-tolerant Generation Ensembling*: KCMF uses the constructed prompts to query the LLM. The multiple outputs generated are then combined, mitigating the ill-formatting issue, to reach the final decision.

3.2 Pseudo-code Design

As Figure 1 depicts, one of the challenges that LLMs face when performing matching is the issue of under-matching and over-matching. This issue stems from the ambiguity of the task instruction "match". Without additional context, the word "match" lacks a clear definition in this setting[1]. Take schema matching as an example. The term "*match*" in this context has at least three different interpretations, depending on the focus of the task:

- The data types of the column values are the same or convertible to each other.
- The table definitions are semantically related.
- The schemas refer to the same real-world concept.

In light of this ambiguity, we propose a strategy to decompose the task into pseudo-code composed of conditional statements directly derived from the task's motivation. As shown in S1 of Figure 2, we define statements as natural language predicates organized into a set of if-then-else logical constructs. Each statement provides a sufficient condition for determining whether cases should be "matched" or "not matched". The designed pseudo-code should cover the task's full range of conditions relevant to the task. To this end, manual effort is required to understand the task's motivation and design the

[1]As shown in Table 11, there is a discrepancy between what GPT-3.5 understands the instruction "match" and the actual motivation of the schema matching task.

pseudo-code $\mathcal{R}$ accordingly, our version of implementation is shown in Algorithm A.4.1. To utilize this pseudo-code, we further propose a reasoning prompting strategy, following the idea of Chain-of-Thought (Wei et al., 2022), which will be discussed in the next section.

3.3 Building Domain Knowledge Set

To solve the matching task within a knowledge-enhanced setting, we first need to retrieve a domain knowledge list K. However, because answers for matching tasks cannot be explicitly derived from retrieved information and due to the scarcity of unstructured domain knowledge, we introduce two mechanisms, Dataset as Knowledge (DaK) and Examples as Knowledge (EaK), to build highly relevant unstructured knowledge sets specifically tailored for matching tasks and the implementation details can be found in Appendix A.4.2.

3.3.1 Dataset as Knowledge (DaK)

Due to privacy concerns, concrete records from the source databases $\mathcal{D}$ and $\mathcal{D}'$ are often inaccessible to the schema matching system (Johnson et al., 2023; Zhang et al., 2023b). This means that the matching task must be performed using only metadata. In this setting, LLMs are required to understand the structures of $\mathcal{D}$ and $\mathcal{D}'$, which, however, can only be partly seen during inference through the limited metadata representations $\{r, r'\}$ (cf. Section 2.4). Hence, we propose DaK to acquire knowledge of the structure of source databases by searching metadata from the full dataset $\mathcal{M}$, as shown in Figure 3.

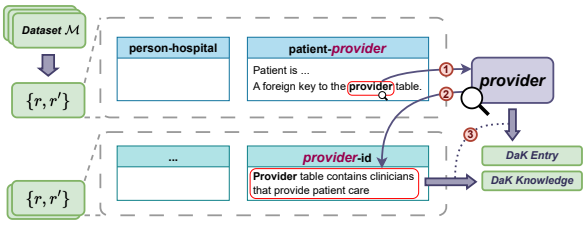

Figure 3: A toy example of DaK. Object "provider" and the description of which are identified, respectively (① & ②); then, they are integrated as an entry and a piece of DaK knowledge (③).

The procedure of DaK aims to discover database objects and their metadata from the candidate pool $\mathcal{M}$ (see Algorithm 1 for details). As shown in Figure 3, DaK operates in three steps: 1) for a given candidate pair $\{r, r'\} \in \mathcal{M}$, DaK identifies an object O; in the example, O is a table named "provider". While this step can utilize techniques like Named Entity Recognition (NER), for rapid prototyping, we extract O by traversing all schema names N and N' associated with r, r' (see ①). 2) DaK then scans the remaining pairs $\{r, r'\}$ to find metadata K_{DaK} related to the identified object O; in this example, K_{DaK} is the description of the table "provider". This is accomplished by matching O with descriptions C from the other candidate pairs (see ②). 3) After obtaining an object list $\mathcal{C}_{\text{Obj}}$ consisting of each identified object O, and a metadata list $\mathcal{C}_{\text{DaK}}$ consisting of all matched knowledge K_{DaK}, DaK proceeds to form the name of O as an entry and K_{DaK} of O as the corresponding DaK knowledge. As shown in Figure 3, the name of the identified object "provider" and the description of "provider" are constructed into a piece of DaK knowledge: {*provider*: *provider table contains clinicians that provide patient care*}.

3.3.2 Examples as Knowledge (EaK)

Using text chunks from a self-built corpus has become a *de facto* approach for retrieval enhancement to reduce LLM hallucinations (Gao et al., 2024). For common-sense QA tasks, such text chunks can be easily retrieved from existing commonsense corpora. However, the knowledge required for matching tasks is domain-specific, and existing KBs are mainly structured as entity databases or thesaurus rather than natural language text. Thus, we propose EaK, as depicted in Figure 4.

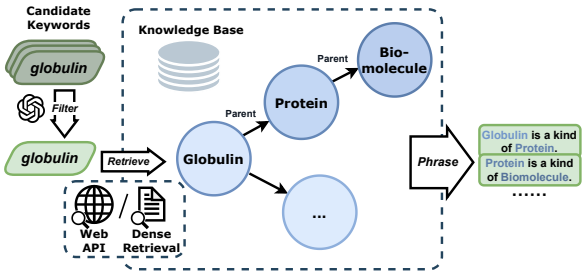

Figure 4: Building a natural language domain knowledge set using EaK.

As its name suggests, EaK aims to explain complex concepts using examples. Given a pair $\{r, r'\}$, EaK first extracts keywords from it by querying an LLM, and then again queries an LLM to filter the domain-irrelevant ones (see Appendix C for details). Obtaining all candidate keywords, EaK uses these candidates to search domain KBs for the top-k related records. For remote KBs, retrieval can be done through a search over the provided web API; while for local KBs, this procedure can be implemented using stronger dense retrieval. For each retrieved record, EaK leverages its relationships

and properties to form explanatory knowledge. For example, in Figure 4, EaK generates the knowledge by phrasing the parent relation.

3.4 Prompt Construction

Pseudo-code-based Reasoning Prompting In a paradigm of k-shot in-context learning, we sample k valid match pairs $\{d, d'\}$ as demonstrations together with the pseudo-code $\mathcal{R}$, the queried pair $\{r, r'\}$, and knowledge list K. The fundamental pattern of the prompt is illusrated in Figure 5.

Prompt

Task Instruction: Are A and B matched?

Pseudo-code $\mathcal{R}$: 1) If columns are not the same type of data, then the answer is no, else check the next statement...

Demonstrations $[\{d, d'\}]$:
Schema A is drug-code, B is drug-id $\{\mathcal{C}_{RSNG}\}\{ans\}$
Schema A is provider-id, B is drug-id $\{\mathcal{C}_{RSNG}\}\{ans\}$

Target Data $\{r, r'\}$:Schema A is patient-id, B is person-id

Knowledge: ID is a database identifier.

Figure 5: A toy prompt bringing together all parts obtained in previous sections, with two new placeholders $\{\mathcal{C}_{\text{RSNG}}\}$ and $\{ans\}$ added,

As demonstrated in Figure 5, there are two placeholders, $\{\mathcal{C}_{\text{RSNG}}\}$ for the reasoning steps and $\{ans\}$ for the answer. To generate $\mathcal{C}_{\text{RSNG}}$ and ans, we first define each single statement in pseudo-code $\mathcal{R}$ (cf. Section 3.2) as $p \rightarrow q$. Then, we check each statement sequentially: for the current statement $p \rightarrow q$, the condition p is checked if is satisfied; if p is fulfilled, terminate the process and set the answer $ans = q$; if not, proceed to the next statement; continue this process until an answer is drawn. The procedure is formalized in Algorithm 2 in Appendix. Next, the reasoning steps obtained, C_{RSNG}, and the answer, ans, will be inserted into the placeholders. This procedure is repeated for all k demonstrations to obtain the complete set of reasoning steps. Although we discuss a simple scenario here, the pseudo-code can be extended to handle more complex cases with multiple reasoning paths by adding an "ELSE" clause to specify alternative statements for the LLM to follow.

Self-Indicator Extraction Inspired by the performance gain observed when generating a high-quality summary at the beginning of reasoning in our experiments, we further introduce a Self-Indicator Extraction pre-task. This plugin module generates a filtered text segment, called self-indicator (K_{SI}), describing the key information from $\{r, r'\}$. Specifically, taking $\{r, r'\}$ and K as input, an LLM is utilized to generate K_{SI} (see A.4.3 for details), and then K_{SI} is appended to the end of the prompts created above.

Demonstration Summarization After decomposing the task instructions into pseudo-code (see Section 3.2), we leverage in-context learning to enable the LLM to learn reasoning behaviors rather than just the concept of "match". To manage the prompt length, we use an LLM to summarize all demonstration pairs $\{d, d'\}$ (see A.4.3 for details). Instead of using the original demonstrations $\{d, d'\}$ in the prompts, we may use the summarized pairs to improve efficiency without losing essential information.

3.5 Inconsistency-tolerant Generation Ensembling (IntGE)

With multiple knowledge sets available, a straightforward utilization is to combine all retrieved knowledge within one prompt, but this practice leads to information within a prompt flooded thus bringing unexpected outputs (Parciak et al., 2024). Therefore, we instead propose IntGE to integrate diverse information sources and improve output stability in a fine-tuning-free manner. We define the available knowledge sets from unique sources as $\mathcal{K} = [K_1, K_2, \ldots, K_N]$. For a given pair $\{r, r'\}$, we retrieve information from each source in $\mathcal{K}$, resulting in N knowledge lists. Each list is used to construct a prompt following the method in Section 3.4, creating N prompts. We ensemble the N binary classification results using majority voting to determine the prediction for $\{r, r'\}$.

As noted in previous studies (Cuconasu et al., 2024; Zhang et al., 2024b), LLMs tend to prefer familiar input from their pre-training phase, and distracting context can lead to poor outputs. IntGE aims to prevent interference between different knowledge sources by separating them into distinct prompts. Inspired by (Wang et al., 2023; Shi et al., 2024), we adopt a straightforward voting strategy. This approach leverages multiple knowledge sources while enhancing the model's robustness against poorly formatted outputs by keeping prompt lengths manageable.

4 Experiments

4.1 Experimental Setup

Dataset Following (Narayan et al., 2022), we choose the challenging Synthea but also the

Methods	MIMIC		Synthea		MMM	
	Acc	F1	Acc	F1	Acc	F1
SMAT	0.9865	0.2020	0.9902	0.3850	-	-
GLM-4-Flash	0.9512	0.0369	0.9895	0.1143	0.9663	0.6841
+KCMF	$0.9816_{+3.04\%}$	$0.1061_{+6.92\%}$	$0.9852_{+0.43\%}$	$0.1538_{+3.95\%}$	$0.9690_{+0.27\%}$	$0.7334_{+4.92\%}$
Llama-3-8B	0.3272	0.0060	0.8870	0.0233	0.9574	0.6738
+KCMF	$0.9774_{+65.02\%}$	$0.0765_{+7.05\%}$	$0.9572_{+7.02\%}$	$0.0863_{+6.30\%}$	$\mathbf{0.9730}_{+1.56\%}$	$\mathbf{0.7783}_{+10.45\%}$
Mistral-7B	0.9931	0.1538	0.9686	0.0792	0.9671	0.7046
+KCMF	$0.9947_{+0.16\%}$	$0.3200_{+16.62\%}$	$0.9895_{+2.09\%}$	$0.1143_{+3.51\%}$	$0.9689_{+0.18\%}$	$0.7206_{+1.60\%}$
GPT-3.5	0.9767	0.0745	0.9865	0.0909	0.9297	0.6242
+KCMF	$0.9944_{+1.77\%}$	$\mathbf{0.6207}_{+54.62\%}$	$\underline{0.9936}_{+0.71\%}$	$\underline{0.4242}_{+33.33\%}$	$\underline{0.9727}_{+4.30\%}$	$\underline{0.7758}_{+15.16\%}$
GPT-4o	**0.9964**	0.3030	0.9919	0.0769	0.9658	0.6738
+KCMF	$\underline{0.9959}_{-0.05\%}$	$\underline{0.3500}_{+4.70\%}$	$\mathbf{0.9946}_{+0.27\%}$	$\mathbf{0.4667}_{+38.98\%}$	$0.9661_{+0.03\%}$	$0.6744_{+0.06\%}$

Table 1: Main results.

MIMIC from the OMAP benchmark (Zhang et al., 2021), for the schema matching task. As for entity matching, we adapted the CC0-licensed MedMentions benchmark (Mohan and Li, 2019) to construct a new dataset called MedMentions Matching (MMM), whose details of modification is presented in Appendix A. Based on the datasets above, we give the statistics as shown in Table 2.

Dataset	Task	# (Instance)	# (Positive)	IR
MIMIC	SM	6408	13	492.9
Synthea	SM	2964	8	370.5
MMM	EM	10736	736	14.6

Table 2: Statistics of evaluation datasets. IR = # (Instance) / # (Positive) is the Imbalance Ratio. The larger the IR, the harder the matching task.

Settings Generally, we follow Section 3 to do the preparation for experiments. All LLMs used in KCMF are always gpt-3.5-turbo-1106 if there is no further notice. Detailed settings and all prompts including the pseudo-code used in the experiments can be found in Appendices A and C. Due to the fine-tuning-free feature of KCMF, we mainly focus on comparing with LLM baselines, but show results of non-LLM methods as well. On Synthea, we choose API-styled gpt-3.5-turbo-1106, gpt-4o, GLM-4-Flash and local deployed Llama-3-8B-Instruct, Mistral-7B-Instruct (Jiang et al., 2023a) as few-shot LLM baselines, and SMAT (Zhang et al., 2021), a previous non-LLM SOTA on Synthea, as the non-LLM baseline. Since SMAT is specifically designed for SM task, we only present its results on MIMIC and Synthea. To query LLMs, we follow the prompt pattern of (Narayan et al., 2022) but add more signal words to align with the instruction-tuned models.

4.2 Main Results

As shown in Table 1, generally KCMF is significantly better than all LLM baselines and SMAT. Specifically, on Synthea, GPT-3.5-version KCMF outperforms the previous SMAT by 3.92% on the F1-score, and on MIMIC the gain increases to 41.87%. In addition, we observe that the F1-score of GPT-3.5-version KCMF outperforms all baselines by an average of 33.73% and 50.59% on Synthea and MIMIC, respectively. On MMM, the strongest F1-score of KCMF surpasses all baselines by an average of 10.62%. Although we see a slight drop in accuracy on the GPT-4o version of KCMF, we consider this a strong enough result because, first, the 0.05% drop is insignificant enough for the unbalanced matching task, and both schemes already achieve accuracy better than 99%; second, the 0.05% drop is due to correct recall of three extra positive cases and incorrect recall of three negative cases, which look reasonable for a dataset with only 13 positive samples.

Pseudo-code ensures recalling while maintaining the precision. Although the specific recall and precision are not reported in the main results, the increase in F1-score shown by KCMF is mainly achieved by recalling more true-positive cases and controlling the growth of false-negative instances so that the precision does not drop drastically, which exactly cater to the matching task's preference for recall in practice, eliminating the need for additional human efforts to perform complex verification on recalled results. The performance gain of KCMF owes to the fact that the recalling ability brought by pseudo-code does not depend on the model's learning of the exposed data distribu-

tion, and thus KcMF can correctly reject negative instances while recalling positive ones.

KcMF generalizes over backbones. We also observe that after implementing KcMF, the F1-score of each backbone is improved by an average of 17.98%, 17.21%, 6.44% on MIMIC, Synthea and MMM, respectively. These experimental results demonstrate that our proposed method KcMF can generalize over different backbones and consistently improve the performance of LLMs on both SM and EM tasks. Interestingly, we observe that on MIMIC, the strongest version of KcMF achieves an F1-score of 0.6207, which is significantly better than the 0.4314 achieved by previous SFT SoTA Jellyfish (Zhang et al., 2024a). This result indicates the great potential of fine-tuning-free LLMs in matching tasks on domain-specific data.

Settings	# (Case)		
	IR	OM	PM
w/o pseudo-code	8	60	11
w. pseudo-code	5	13	1
Δ	-37.5%	-78.3%	-90.9%

Table 3: Number of erroneous cases with and without pseudo-code on Synthea.

4.3 Task-specific pseudo-code disambiguates the task instruction

The primary purpose of the pseudo-code is to avoid LLM's confusion about the task's motivation by designing explicit conditional statements. To explore its effectiveness, we examine KcMF on Synthea without the designed statements: We removed all pseudo-codes and the corresponding reasoning steps from the demonstrations to compare with the complete KcMF.

We categorize the error types of all false-positive samples into three types: the *Over-matching* (OM), which is already mentioned in Figure 1; The *Position Mismatching* (PM), which indicates LLM concludes a positive answer by mismatching the different parts of r and r', e.g., schema A's table and schema B's column; And the *Incorrect Reasoning* (IR), which is the other kind of errors can not be categorized into the former two types. As shown in Table 3, it is obvious that the pseudo-code does mitigate all the three error categories above.

4.4 IntGE ensures robustness towards badly-formatted outputs

To examine whether KcMF gains from IntGE, we test a knowledge-all-in-one (AIO) version of KcMF, which removes IntGE and concatenates the knowledge from all sources into one prompt.

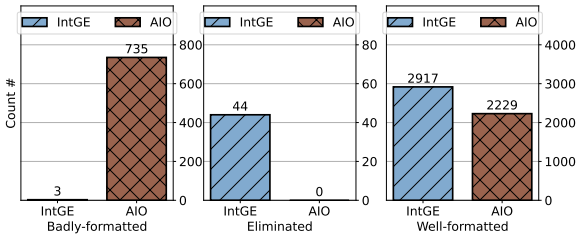

Figure 6: Ablations of IntGE on Synthea.

The results are categorized into three types as shown in Figure 6. *Badly-formatted* means the output format of the KcMF does not follow the format given in the demonstrations, and *Well-formatted* means the opposite. *Eliminated* means that the badly-formatted output appeared in IntGE's votes but was eventually eliminated by IntGE's voting strategy, thus the final output remains Well-formatted. The results clearly indicate that the AIO setting performed poorly. In contrast, IntGE effectively reduces the number of badly-formatted output via its multi-prompt voting mechanism. The results imply a paradigm for retrieving knowledge from different sources for inference-only LLMs, namely to split the retrieved knowledge by its source into multiple prompts to achieve better results while limiting the length of each prompt.

5 Conclusion

We presented KcMF, a fine-tuning-free framework for solving data matching tasks under a knowledge-enhanced setting, with a pseudo-code based task instruction strategy for addressing the problem of confusion and mechanisms DaK, EaK, and IntGE for building and utilizing knowledge from various sources in scarcity of unstructured domain knowledge. Our evaluation showed that KcMF outperformed all LLM baselines and non-LLM SoTA under a completely fine-tuning-free setting and our decomposition diagram effectively mitigated LLMs' confusion towards the matching tasks. Our future work involves extending KcMF to make it suitable for both metadata-based and instance-based scenarios, and further improving the efficiency of the matching workflow.

Limitations

One limitation of our work is the need for human labor to select domain KBs as the knowledge sources, which may pose a limitation in implementation. Also, this paper primarily focuses on the scenario where matching is performed by predicting enumerated data pairs instead of directly identifying potential mappings among the source databases, which may increase the computational burden and thus have an environmental impact, we leave this efficiency optimization potential to future work. Moreover, since matching tasks are usually studied on enterprise data, there are few open-source benchmarks, and our experiments are only conducted on MIMIC, Synthea and MMM. Due to this scarcity, it would be worth constructing more benchmarks to push this limit further. Furthermore, though KCMF performs better than all LLM baselines and the non-LLM SoTA, it only outperforms SFT-LLM SoTA on MIMIC. We deem this gap a trade-off with fine-tuning efforts in our current research stage, and it would be intriguing to go further to build a stronger fine-tuning-free system.

References

Bogdan Alexe, Mauricio Hernández, Lucian Popa, and Wang-Chiew Tan. 2010. Mapmerge: Correlating independent schema mappings. *Proceedings of the VLDB Endowment*, 3(1-2):81–92.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *arXiv preprint*. ArXiv:2310.11511 [cs].

Md Asif-Ur-Rahman, Bayzid Ashik Hossain, Michael Bewong, Md Zahidul Islam, Yanchang Zhao, Jeremy Groves, and Rory Judith. 2023. A semi-automated hybrid schema matching framework for vegetation data integration. *Expert Systems with Applications*, 229:120405.

Paolo Atzeni, Luigi Bellomarini, Paolo Papotti, and Riccardo Torlone. 2019. Meta-mappings for schema mapping reuse. *Proceedings of the VLDB Endowment*, 12(5):557–569.

Nils Barlaug and Jon Atle Gulla. 2021. Neural networks for entity matching: A survey. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 15(3):1–37.

Chen Chen, Behzad Golshan, Alon Halevy, Wang-Chiew Tan, and AnHai Doan. 2018. BigGorilla: An Open-Source Ecosystem for Data Preparation and Integration.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729.

Muhammad Ebraheem, Saravanan Thirumuruganathan, Shafiq Joty, Mourad Ouzzani, and Nan Tang. 2018. DeepER – Deep Entity Resolution. *Proceedings of the VLDB Endowment*, 11(11):1454–1467. ArXiv:1710.00597 [cs].

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

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint*. ArXiv:2312.10997 [cs].

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7B. *arXiv preprint*. ArXiv:2310.06825 [cs].

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active Retrieval Augmented Generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7969–7992, Singapore. Association for Computational Linguistics.

Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Li-wei H. Lehman, Leo A. Celi, and Roger G. Mark. 2023. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1):1.

Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource deep entity resolution with transfer and active learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5851–5861.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2. Minneapolis, Minnesota.

Mohamed Kettouch, Cristina Luca, Mike Hobbs, and Sergiu Dascalu. 2017. *Using semantic similarity for*

schema matching of semi-structured and linked data. Pages: 133.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.

Huahang Li, Shuangyin Li, Fei Hao, Chen Jason Zhang, Yuanfeng Song, and Lei Chen. 2024. Booster: Leveraging large language models for enhancing entity resolution. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 1043–1046.

Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proceedings of the VLDB Endowment*, 14(1):50–60. ArXiv:2004.00584 [cs].

Sunil Mohan and Donghui Li. 2019. MedMentions: A Large Biomedical Corpus Annotated with UMLS Concepts. *arXiv preprint*. ArXiv:1902.09476 [cs].

Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can foundation models wrangle your data? *Proceedings of the VLDB Endowment*, 16(4):738–746.

Marcel Parciak, Brecht Vandevoort, Frank Neven, Liesbet M. Peeters, and Stijn Vansummeren. 2024. Schema Matching with Large Language Models: an Experimental Study. *arXiv preprint*. ArXiv:2407.11852 [cs].

Ralph Peeters and Christian Bizer. 2023. Using chatgpt for entity matching. In *European Conference on Advances in Databases and Information Systems*, pages 221–230. Springer.

Ralph Peeters and Christian Bizer. 2024. Entity Matching using Large Language Models. *arXiv preprint*. ArXiv:2310.11244 [cs].

Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A Survey of Hallucination in Large Foundation Models. *arXiv preprint*. ArXiv:2309.05922 [cs].

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Eitam Sheetrit, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. ReMatch: Retrieval Enhanced Schema Matching with LLMs. *arXiv preprint*. ArXiv:2403.01567 [cs].

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8364–8377.

Saravanan Thirumuruganathan, Shameem A. Puthiya Parambath, Mourad Ouzzani, Nan Tang, and Shafiq Joty. 2018. Reuse and Adaptation for Entity Resolution through Transfer Learning. *arXiv preprint*. ArXiv:1809.11084 [cs, stat].

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-Consistency Improves Chain of Thought Reasoning in Language Models. *arXiv preprint*. ArXiv:2203.11171 [cs].

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. RECOMP: Improving Retrieval-Augmented LMs with Compression and Selective Augmentation. *arXiv preprint*. ArXiv:2310.04408 [cs].

Wenhao Yu, Hongming Zhang, Xiaoman Pan, Kaixin Ma, Hongwei Wang, and Dong Yu. 2023. Chain-of-Note: Enhancing Robustness in Retrieval-Augmented Language Models. *arXiv preprint*. ArXiv:2311.09210 [cs].

Alexandros Zeakis, George Papadakis, Dimitrios Skoutas, and Manolis Koubarakis. 2023. Pre-Trained Embeddings for Entity Resolution: An Experimental Analysis. *Proceedings of the VLDB Endowment*, 16(9):2225–2238.

Haochen Zhang, Yuyang Dong, Chuan Xiao, and Masafumi Oyamada. 2024a. Jellyfish: A Large Language Model for Data Preprocessing. *arXiv preprint*. ArXiv:2312.01678 [cs].

Jing Zhang, Bonggun Shin, Jinho D. Choi, and Joyce C. Ho. 2021. SMAT: An Attention-Based Deep Learning Solution to the Automation of Schema Matching. In Ladjel Bellatreche, Marlon Dumas, Panagiotis Karras, and Raimundas Matulevičius, editors, *Advances in Databases and Information Systems*, volume 12843, pages 260–274. Springer International Publishing, Cham. Series Title: Lecture Notes in Computer Science.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. 2023a. Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models. *arXiv preprint*. ArXiv:2309.01219 [cs].

Yuji Zhang, Sha Li, Jiateng Liu, Pengfei Yu, Yi R. Fung, Jing Li, Manling Li, and Heng Ji. 2024b. Knowledge Overshadowing Causes Amalgamated Hallucination in Large Language Models. *arXiv preprint*. ArXiv:2407.08039 [cs].

Yunjia Zhang, Avrilia Floratou, Joyce Cahoon, Subru Krishnan, Andreas C. Müller, Dalitso Banda, Fotis Psallidas, and Jignesh M. Patel. 2023b. Schema Matching using Pre-Trained Language Models. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 1558–1571. ISSN: 2375-026X.

A Methodology

A.1 Definition of Entity Matching

Because entity matching is performed on the level of records, we can define the task of entity matching by slightly changing the definition of r described in Section 2.4:

$$r = \{N, attr_1, attr_2, ...attr_m\} \quad (2)$$

Here N similarly represents the name of the entity, but we replace the schema description C with a sequence of $attr$, which represents all the other column values of the entity r. Specifically, for the MMM benchmark where only entity name and entity context are available, r can be directly derived from the serialization of the name and context.

A.2 Algorithms

Algorithm 1 Metadata Discovery

Input: A matching task dataset D
Output: A metadata list $\mathcal{C}_{\text{DaK}}$ and an object list $\mathcal{C}_{\text{Obj}}$

$\mathcal{C}_{\text{DaK}} \leftarrow \emptyset, \mathcal{C}_{\text{Obj}} \leftarrow \emptyset$
for $\{r_i, r'_i\} \in D$ **do**
  $O \leftarrow ObjectDiscovery(r_i, r'_i)$
  for $\{r_j, r'_j\} \in D, j \neq i$ **do**
    $K_{\text{DaK}} \leftarrow FindMetadata(O, r_j, r'_j)$
    if $K_{\text{DaK}} \neq$ NULL **then**
      $\mathcal{C}_{\text{DaK}}.insert(K_{\text{DaK}})$
      $\mathcal{C}_{\text{Obj}}.insert(O)$
    end if
  end for
end for
return $\mathcal{C}_{\text{DaK}}, \mathcal{C}_{\text{Obj}}$

Algorithm 2 Constructing reasoning steps

Input: current demonstration $\{d, d'\}$ and the pseudo-code $\mathcal{R}$
Output: a reasoning steps C_{RSNG} and a corresponding final answer ans

$C_{\text{RSNG}} \leftarrow \emptyset$
for $R \in \mathcal{R}'$ **do**
  $C_{\text{RSNG}}.append(genReasoningStep(R, d, d'))$
  if $isConditionMet(R, d, d')$ or $isLastStatement(R, \mathcal{R})$ **then**
    $ans \leftarrow getQ(R)$
    return C_{RSNG}, ans
  end if
end for

A.3 MMM Modification

We present **M**ed**M**entions **M**atching, or MMM, a clinical entity matching dataset modified from MedMentions (Mohan and Li, 2019). MedMentions is originally an entity linking dataset whose raw data is recognized entity mentions from PubMed documents labeled with Unified Medical Language System (UMLS) code. Inspired by the fact that entities referred to the same real-world concept are matched from the view of entity matching, we re-organized the entities in MedMentions like Figure 7: For entities belonging to the same UMLS concept, we keep the pairs with lowest similarity as positive. For entities belonging to different UMLS concepts, we sample the pairs with the highest similarity as negative. Then we use sentences where sampled entities are located as context attributes. MMM contains 7,359 positive and 100,000 negative examples in total and an example can be found in Figure 8.

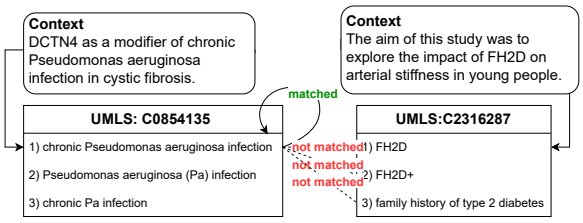

Figure 7: The process of constructing MedMentions Matching (MMM) dataset.

Entity A	Context A	Entity B	Context B	Label
chronic Pseudomonas aeruginosa	DCTN4 as a modifier of chronic Pseudomonas aeruginosa..	FH2D	The aim of this study was to explore the impact of FH2D...	**0** (not matched)
...				

Figure 8: An example from MMM.

A.4 Implementation

A.4.1 Pseudo-code Design

The pseudo-code we used in our experiments is shown in Tables 20 and Table 21, and the given pseudo-code is directly written by the author according to the motivation of the matching task. As emphasized in the main text, the pseudo-code statements are defined as "IF-THEN-ELSE". In order to better utilize the large language model's natural language comprehension capability, we have corresponded "IF-THEN-ELSE" to "IF, THEN, otherwise" in the actual design of the pseudo-code. To ensure that the given pseudo-code is checked in the correct order, a statement is added at the beginning of the designed pseudo-code to declare that the rest of the statements should be checked in order, which can be removed when using stronger LLMs (e.g., GPT-4o). Also, to cover all the task motivations of the matching task, we include overlapped conditions in the pseudo-code statement design, and for the last statement of all the pseudo-codes, we set a

general condition to ensure full coverage.

A.4.2 Knowledge Retrieval & Construction

The retrieval pipeline in this paper is based on keywords extracted from $\{r, r'\}$, then the quality-managed keywords are used to retrieve and construct the knowledge sets following the techniques demonstrated in Section 3. Specifically, we adopt a gpt-3.5-1106 to extract domain-specific, difficult-to-understand keywords from $\{r, r'\}$ (see Table 12 for details of the prompt). After that, we use a gpt-3.5-1106 to filter the extracted keywords (see Table 13 and Table 14 for details of the prompt), and we empirically design a blacklist for further rejecting low-quality keywords.

To construct the EaK knowledge set, we select Snomed-CT as the knowledge base and use the Snomed-CT API[2] based on the above keywords to search for the associated entities in the KB, specifically here we keep the top-1 search results, and for the searched entities, we query their children entities again using the Snomed-CT API and randomly sample up to 3 children entities. The parent and children are then serialized into EaK knowledge in the form of "One of parent is children".

For out-of-domain knowledge sources, we based on the English Wikidata API[3] to search for entity codes using extracted keywords. We use the searched top-1 entity code to construct SPARQL statements for knowledge query from Wikidata[4] while using this entity code to query the page extraction from English Wikipedia[5]. Then, for the retrieved results, we use gpt-3.5-turbo-1106 here to summarize the retrieved extractions to limit their length to no more than 1000 words under a zero-shot setting (see Table 17).

A.4.3 Prompt Construction

We use a gpt-3.5-1106 to extract the self-indicator of schema pairs (or entity pairs in MMM), used prompts can be found in 15 . To implement Demonstration-summarization-only, we also use a gpt-3.5-1106 to compress the input $\{r, r'\}$. Specifically, to control the length of the prompt, we split $\{r, r'\}$ into 2 queries (see Table 16 for detailed prompt).

[2]https://browser.ihtsdotools.org/snowstorm/snomed-ct
[3]https://www.wikidata.org/w/api.php
[4]https://query.wikidata.org/
[5]https://en.wikipedia.org/w/api.php

A.4.4 Inference Settings

For MIMIC and Synthea, we take Wikidata with DaK, Wikipedia with EaK, and EaK as the knowledge sources to evaluate KCMF under 4-shot. For MMM, we take Wikidata, Wikipedia, and EaK as the knowledge sources to evaluate the framework under 2-shot. We empirically set temperature to 0 and top_p to 0.1 for stable outputs if these two parameters are available. Also, we directly took poorly formatted outputs as negative predictions in our experiments for metrics calculation and workflow automation. The pseudo-code we designed for these two tasks can be found in Appendix C. We derived the main results on MIMIC and Synthea from the best result of running five times, and for MMM, due to its larger size of data, we ran each setting once as the final results.

A.4.5 Dataset & Code

All datasets can be accessed through original papers cited in this study, specifically for modifying MMM, the script is given in our supplementary codes.

B Ablation Study

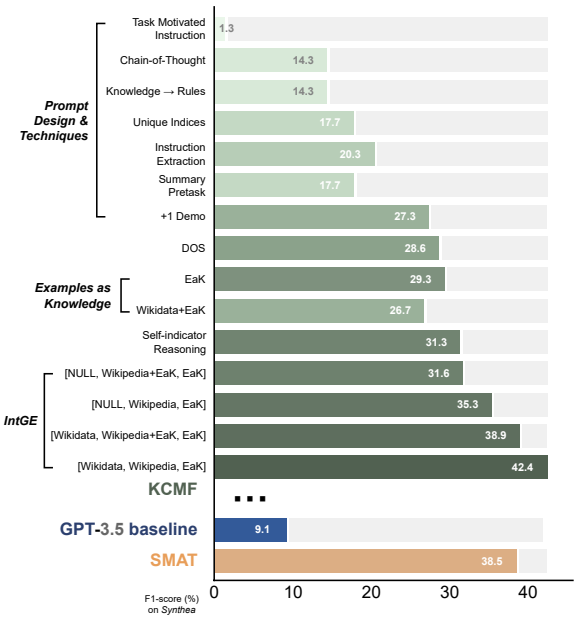

Figure 9: KCMF evolved from a plain zero-shot LLM query method. The horizontal axis depicts the F1-score for each version of KCMF on Synthea.

B.1 Prompt Design & Techniques

We take the Wikidata+DaK with a 3-shot on Synthea as the initial setting and conduct incre-

Increment	Shot	Accuracy	F1-score
TO Inst.	3	0.5735	0.0125
CoT	3	0.9798	0.1429
Know2Rule	3	0.9798	0.1429
U-indices	3	0.9811	0.1765
Inst. Extraction	3	0.9841	0.2034
Summary Pretask	3	0.9781	0.1772
+1 demo	4	**0.9892**	**0.2727**

Table 4: Results of prompt design and techniques' ablation experiment.

mental experiments. The results are presented in Table 4.

TO Inst. denotes Task-Oriented Instruction. Based on the instruction used by (Narayan et al., 2022), we changed it to "Can records in schema B be transformed and stored into schema A?", which is more compliant with the motivation of the schema matching task.

Know2Rule denotes Knowledge to Rule. We renamed the "knowledge for the task" used in the prompt "rules for task".

U-indices denotes Unique indices. We apply unique indices for the different elements of the prompt that need to be numbered. We use Roman numerals, lowercase letter numbers, and numeric numbers separately for the row number of pseudo-code, the sequence number of retrieved knowledge, and sequence number of reasoning steps.

For Inst. Extraction, we extract the instructions and rules for each demonstration and place them at the beginning of the prompt.

All the optimizations we added are motivated by achieving higher precision while ensuring promising recalls. As shown in Table 4, CoT, U-indices and Inst. Extraction brought 13.04%, 3.36%, 2.69% improvement in F1-score on Synthea dataset. Although Summary Pretask caused a temporary drop in performance, the combination of Summary Pretask and +1 Demo still shows a strong result; the F1-score is improved by 6.93% compared to the Inst. Extraction step. We examine the results of the Summary Pretask and the +1 Demo as a whole here because the motivation for the Summary Pretask was to shorten the length of the prompt.

B.2 Summary Pretask

In order to analyze the effectiveness of each module in KCMF, we compare the KCMF with its variants under different settings.

For the summary pretask, we apply two strategies: All and *Demo Only*, where *Demo Only* is the strategy of summarizing only the demonstration used in the prompt, as shown in Section 3.4. All is the strategy of adding the summarizing of the target samples on top of Demo Only (in this case we summarize the descriptions of given schema pairs). As shown in Table 5, the compression strategy of Demo Only outperforms All strategy by 1.3% on F1-socre under the 4-shot setting.

Strategy	Accuracy	F1-score
All	0.9892	0.2727
Demo only	**0.9899**	**0.2857**

Table 5: Ablation results of summary pretask's strategy on Synthea.

B.3 Examples as Knowledge

Knowledge Source	Accuracy	F1-score
Wikidata	0.9899	0.2857
EaK	**0.9902**	**0.2927**
Wikidata+EaK	0.9889	0.2667

Table 6: Ablation results of EaK on Synthea.

Following the *Demo Only* setting in B.2, we further verify the validity of the knowledge constructed through EaK by introducing EaK as a knowledge source and form three available knowledge sets of Wikidata, EaK, and Wikidata+EaK for constructing the prompt. As shown in Table 6, due to the short length and high quality of EaK knowledge, single EaK as the knowledge set achieves the best results, outperforms single Wikidata and Wikidata+EaK 0.7% and 2.6% on F1-score.

B.4 Self-indicator Reasoning (SIR)

In order to verify the effectiveness and generalizability of self-indicator reasoning, for MMM, we introduce SIR into demonstrations under cases with different knowledge sources.

Knowledge Source	w. SI	Accuracy	F1-score
Wikidata	No	0.9498	0.6534
Wikidata	Yes	0.9667	0.7387
Wikipedia	No	0.9603	0.6979
Wikipedia	Yes	0.9719	0.7684
EaK	No	0.9593	0.6997
EaK	Yes	**0.9729**	**0.7767**

Table 7: 2-shot ablation results of self-indicator reasoning on MMM: SI denotes self-indicator.

Table 7 presents the results with and without self-indicator reasoning on the MMM dataset with

Wikidata, Wikipedia, and EaK as the knowledge sources, respectively. We observe that all cases with SIR outperform cases without SIR, with their F1-scores improving by an average of 7.73%. More specifically, F1-scores of SIR with Wikidata, Wikipedia, and EaK as knowledge sources improved by 8.43%, 7.05%, and 7.70%, respectively.

Knowledge Source	w. SI	Accuracy	F1-score
Wikidata	No	0.9899	0.2857
Wikidata	Yes	**0.9926**	**0.3125**

Table 8: 4-shot ablation results of self-indicator reasoning on Synthea.

Meanwhile, for Synthea, samely based on the *Demo Only* setting, we also introduce SIR with Wikidata as the knowledge source. As shown in Table 8, SIR improved the F1-score on Synthea by 2.68%.

B.5 Inconsistency-tolerant Generation Ensembling

The ablation experiments for IntGE are mainly further experiments based on SIR, here we combine the available knowledge sources and check the performance of IntGE under different combinations. Since the effect of SI on Synthea is less significant compared to MMM, we also task SI as an experimental variable,

As shown in Table 9, we tried different combinations of knowledge sources for IntGE on Synthea, and all the ensembled results outperform the result from single knowledge source Wikidata. The F1-scores of IntGE under combinations [NULL, Wikipedia+EaK, EaK], [NULL, Wikipedia, EaK], [Wikidata, Wikipedia+EaK, EaK], [Wikidata, Wikipedia, EaK] are improved by 7.74%, 4.04%, 11.04%, 0.33%, respectively.

We conducted similar experiments on the MMM dataset, as shown in Table 10. The F1-scores of IntGEs based on the combinations [Wikidata, Wikipedia, EaK], [Wikidata, Wikipedia, EaK*], [Wikidata*, Wikipedia, EaK], [Wikipedia, EaK, NULL] increased by 3.61%, 2.88%, 3.71%, and 3.71%, respectively, as compared to the results based on the single Wikidata data source.

Knowledge Source	Voting	Accuracy	F1-score
Wikidata	No	0.9926	0.3125
[NULL, Wikipedia+EaK, EaK]	Yes	0.9926	0.3889
[NULL, Wikipedia, EaK]	Yes	0.9926	0.3529
[Wikidata, Wikipedia+EaK, EaK]	Yes	**0.9936**	**0.4242**
[Wikidata, Wikipedia, EaK]	Yes	0.9912	0.3158

Table 9: 4-shot ablation results of IntGE on Synthea: Here and subsequently NULL denotes No-knowledge.

Knowledge Source	Voting	Accuracy	F1-score
Wikidata	No	0.9667	0.7387
[Wikidata, Wikipedia, EaK]	Yes	**0.9727**	0.7748
[Wikidata, Wikipedia, EaK*]	Yes	0.9715	0.7675
[Wikidata*, Wikipedia, EaK]	Yes	**0.9727**	**0.7758**
[Wikipedia, EaK, NULL]	Yes	**0.9727**	**0.7758**

Table 10: 2-shot ablation results of IntGE on MMM: Asterisk means the prompt of corresponding knowledge source contains no self-indicator in reasoning steps.

C Prompts

Are schema A and B matched? Let's think step by step. **schema A**: person-month_of_birth **description of schema A**: the person domain contains records that uniquely identify each patient in the source data who is time at-risk to have clinical observations recorded within the source systems.;the month of birth of the person. for data sources that provide the precise date of birth, the month is extracted and stored in this field. **schema B**: patients-birthdate **description of schema B**: patient demographic data.;the date the patient was born.
Based on the descriptions provided: **Schema A:** - **Name:** person-month_of_birth - **Description:** This schema appears to be related to a person domain where records uniquely identify each patient. It specifically captures the month of birth of the person. If the source data provides the precise date of birth, only the month is extracted and stored in this field. **Schema B:** - **Name:** patients-birthdate - **Description:** This schema relates to patient demographic data and specifically captures the date when the patient was born. Now, to determine if schema A and schema B are a match, we need to consider the following points: 1. **Data Captured:** - Schema A captures only the month of birth if the precise date is available. - Schema B captures the full birthdate. 2. **Granularity:** - Schema A has a lower granularity by capturing only the month, whereas Schema B captures the full date. 3. **Context and Usage:** - Schema A seems more focused on situations where the precise date is not necessary, and only the month suffices (possibly for anonymization or simplification purposes). - Schema B is more comprehensive as it captures the exact birthdate, which is typically more useful for demographic analysis and age calculation. **Conclusion:** Based on the descriptions: - Schema A (‘person-month_of_birth‘) and Schema B (‘patients-birthdate‘) do not appear to be a direct match because they capture different levels of detail regarding the birthdate of patients. Schema A focuses on the month of birth, while Schema B captures the full birthdate. Therefore, they serve different purposes and would likely be used in different contexts within a database or system that manages patient or person-related data.

Table 11: GPT-3.5's response on Synthea.

You need to extract all the keywords in the schema that require special domain knowledge to understand, keywords should be separated by commas.
Example 1:
Schema: provider-npi
Schema description: the provider table contains a list of uniquely identified healthcare providers. these are individuals providing hands-on healthcare to patients, such as physicians, nurses, midwives, physical therapists etc.;the national provider identifier (npi) of the provider.
Answer: national provider identifier, npi

Example 2:
Schema: imaging_studies-sop description
Schema description: patient imaging metadata.;description of the sop code.
Answer: sop

Example 3:
Schema: procedure_occurrence-modifier_concept_id
Schema description: the procedure_occurrence table contains records of activities or processes ordered by, or carried out by, a healthcare provider on the patient to have a diagnostic or therapeutic purpose. procedures are present in various data sources in different forms with varying levels of standardization.;a foreign key to a standard concept identifier for a modifier to the procedure (e.g. bilateral). these concepts are typically distinguished by 'modifier' concept classes (e.g., 'cpt4 modifier' as part of the 'cpt4' vocabulary).
Answer: foreign key, identifier, cpt4

Your turn:

Table 12: Prompt for keyword extraction.

You need to find out which of the given keywords are relevant to the database or medical field and return them, keywords should be separated by commas.

Example 1:
birthday, home, dcm, location, primary key
Answer: dcm, primary key

Example 2:
endtime, id, recognition, observation, data model, algorithm, artificial
Answer: id, data model

Your turn:

Table 13: Prompt for keyword quality management on MIMIC and Synthea.

Tell if the given word is a hard-to-understand biomedical domain term, only yes or no.

Example 1:
leaf
Answer: no

Example 2:
deoxynivalenol
Answer: yes

Your turn:

Table 14: Prompt for keywords quality management on MMM.

Instruction: Given two schemas, you need to summarize the column and table for each considering the given knowledge.
[*Examples*]
Your turn:
$\{r, r'\}$
$[K_i]$
Answer:

Table 15: Prompt for Self-indicator (SI) extraction.

Instruction: You need to summarize the given schema based on its schema name and description. The summary should be focused on retaining and explaining concepts in the database domain. Schema name is the table and column names of the schema separated by a dash. Schema description is the table and column descriptions of the schema separated by a semicolon.

[*Examples*]

Your turn:
$\{r, r'\}$
Answer:

Table 16: Prompt for Demonstration Summarization.

Instruction: You need to summarize the given text into a paragraph less than 1000 words.
{CONTENT}
Your answer:

Table 17: Prompt for retrieval summarization.

Are schema A and B the same? ONLY yes or no.
[*Examples*]
Your turn:
$\{r, r'\}$
{*RESPONSE*}

Table 18: Prompt template of baseline on MIMIC and Synthea.

Question: Do entity A and entity B refer to the same real-world concept? Only yes or no.
[*Examples*]
Your turn:
$\{r, r'\}$
Answer:
{*RESPONSE*}

Table 19: Prompt template of baseline on MMM.

I: Rules II, III, and IV MUST be checked SEQUENTIALLY until you conclude an answer.
II: If the columns of the two schemas can not be the same type of data in the database, the answer is no, otherwise, check rule III.
III: If the tables of the two schemas are not semantically the same, the answer is no, otherwise, check rule IV.
IV: If the columns of the two schemas do not refer to the same concept, the answer is no, otherwise, the answer is yes.

Table 20: Pseudo-code for MIMIC and Synthea.

I: Rules II, III, and IV MUST be checked SEQUENTIALLY until you conclude an answer.
II: If Entity A is an abbreviation of Entity B or vice versa, the answer is yes, otherwise check rule III.
III: If Entity A is an alias of Entity B or vice versa, the answer is yes, otherwise check rule IV.
IV: If Entity A and Entity B refer to the same real-world concept the answer is yes, otherwise the answer is no.

Table 21: Pseudo-code for MMM.

Question:
Can records in schema B be transformed and stored in schema A? The task should be solved by completing the reasoning steps and concluding a final answer ONLY yes or no. Do not stop until you draw a final answer. Schema name is the table and column names of the schema separated by a dash.
Rules for the task:
[$\mathcal{R}$]
[*Demonstrations*]
Your turn:
$\{r, r'\}$
Knowledge for the task:
[K_i]
Reasoning:
1. {*Self-indicator*}
Please continue the reasoning until you draw a final answer ONLY yes or no:
{*RESPONSE*}

Table 22: Prompt template of KCMF on MIMIC and Synthea.

Question: Do entity A and entity B refer to the same real-world concept? You must think step by step, and finally draw an answer only yes or no. **Rules for the task**: [$\mathcal{R}$] [*Demonstrations*] **Your turn**: $\{r, r'\}$ **Knowledge for the task**: [K_i] **Reasoning**: 1. {*Self-indicator*} **Please continue the reasoning until you draw a final answer ONLY yes or no**: {*RESPONSE*}

Table 23: Prompt template of KCMF on MMM.