ReMatch: Retrieval Enhanced Schema Matching with LLMs

Eitam Sheetrit*
Microsoft
Israel
eitams@microsoft.com

Menachem Brief*
Microsoft
Israel
menibrief@microsoft.com

Moshik Mishaeli*
Microsoft
Israel
mmishaeli@microsoft.com

Oren Elisha
Microsoft
Israel
oren.elisha@microsoft.com

ABSTRACT

Schema matching is a crucial task in data integration, involving the alignment of a source database schema with a target schema to establish correspondence between their elements. This task is challenging due to textual and semantic heterogeneity, as well as differences in schema sizes. Although machine-learning-based solutions have been explored in numerous studies, they often suffer from low accuracy, require manual mapping of the schemas for model training, or need access to source schema data which might be unavailable due to privacy concerns. In this paper we present a novel method, named ReMatch, for matching schemas using retrieval-enhanced Large Language Models (LLMs). Our method avoids the need for predefined mapping, any model training, or access to data in the source database. In the ReMatch method the tables of the target schema and the attributes of the source schema are first represented as structured passage-based documents. For each source attribute document, we retrieve J documents, representing target schema tables, according to their semantic relevance. Subsequently, we create a prompt for every source table, comprising all its attributes and their descriptions, alongside all attributes from the set of top J target tables retrieved previously. We employ LLMs using this prompt for the matching task, yielding a ranked list of K potential matches for each source attribute. Our experimental results on large real-world schemas demonstrate that ReMatch significantly improves matching capabilities and outperforms other machine learning approaches. By eliminating the requirement for training data, ReMatch becomes a viable solution for real-world scenarios.

1 INTRODUCTION

Schema matching is a fundamental task in data management and integration, involving the identification of semantic correspondences between elements of two or more database schemas [15]. This process is essential as it lays the groundwork for various data manipulation and integration tasks, including data warehousing, database federation, and the merging of information systems. At its core, schema matching aims to establish mappings between schema elements that are semantically related, regardless of differences in naming, structure, or data type.

The necessity for schema matching arises from the ever-growing volume of data generated by diverse applications and organizations, where data is often trapped in siloed repositories, each with its unique schema. In domains ranging from healthcare to retail, efficient schema matching can lead to more informed decision-making, seamless integration of heterogeneous systems, and ultimately, a competitive advantage in data-driven insights.

However, the schema matching task is challenging due to several inherent complications. Firstly, schemas are designed with different perspectives and terminologies, namely textual heterogeneity, reflecting the conceptualization of domain experts from disparate fields. This semantic heterogeneity can lead to ambiguous mappings where schema elements have the same name but different meanings, or different names but the same meaning. Secondly, structural heterogeneity compounds this complexity, with schemas exhibiting varied architectures, hierarchies, constraints, and model granularity differences.

Human schema matching, a manual and time-consuming process, requires significant effort from skilled individuals. This can be expensive and impractical, particularly in large-scale projects. Furthermore, human matchers are prone to errors and inconsistencies due to cognitive biases and fatigue [27]. Their performance can also be influenced by the complexity and ambiguity of the schema elements, leading to potential inaccuracies in the matched results. As a result, the automation of schema matching has become a major focus within the AI and database-oriented research community over the years.

In recent years large language models (LLMs) have achieved significant advancements across many challenging tasks that require a deep understanding of semantics. This includes tasks that until recently required significant human efforts. These models have shown an impressive ability to generalize to new tasks without any task-specific fine-tuning, even in areas significantly divergent from the ones they were originally trained on, including various data related tasks [19, 32].

In this paper, we present a new approach that unlike previously proposed machine learning (ML) methods, circumvents the need for predefined mapping, model training, or access to data in the source database. Our approach, which we call ReMatch, improves the task of schema matching by using retrieval-enhanced LLMs. Fig. 1 illustrates our proposed approach. First, we represent schema tables and attributes (columns) as structured documents (1); then, for each source attribute, we retrieve semantically similar target tables (2); Lastly, we create a prompt containing source attributes and target candidate attributes, and employ LLM for the task of selecting top K most relevant target attributes, yielding a list of potential K matches for each attribute in the source schema (3).

*Equal contribution.

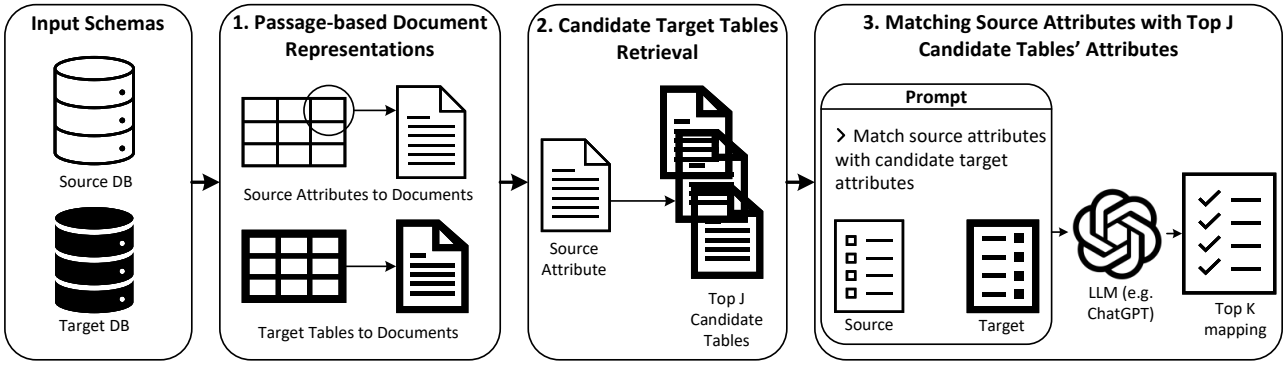


Figure 1: Overview of the ReMatch method.

Our contributions in this study are as follows: (1) Introducing a new method for the task of schema matching, which allows for scalable and accurate matching results, without any model training or access to labeled data. (2) Proposing a mechanism to reduce the search space of the target schema for efficient candidate generation, inspired by techniques used in information retrieval (IR), and in particular, by retrieval augmented generation (RAG) methods. (3) Exploiting the generative abilities of LLMs, and their text comprehension to perform semantic ranking between two schemas, instead of Cartesian-product-based classification, in alignment with human matchers. (4) Demonstrating the effectiveness of our proposed method, in the challenging healthcare domain, and provide a complete mapping between two important healthcare schemas. To the best of our knowledge, this is currently the largest publicly available schema matching dataset.

The rest of the paper is structured as follows. The next section provides background on LLMs and on schema matching. Section 3 presents our proposed method. Section 4 describes the experimental evaluation and results. Section 5 reviews related work, Section 6 discusses insights from the obtained results and limitations, and finally, Section 7 concludes the paper.

2 BACKGROUND

2.1 Large Language Models (LLMs)

A paradigm shift in language models (LMs) was initiated with the introduction of Transformer models by Vaswani et al.[30]. This architecture enables models to capture long-range dependencies between input and output sequences, overcoming limitations in previous models. The power of transformer models is reflected in generative and embedding models such as the GPT model family [2, 24, 25] and BERT [4]. Following the introduction of ChatGPT [21], the terms *Large Language Model* and *LLMs* have become synonymous with general purpose pretrained language models.

Generative LMs like GPT are trained to generate human-like text and can be fine-tuned for a variety of tasks, but also can be reused for a variety of tasks, with no additional training [2, 14, 17, 24]. Embedding models like BERT or Ada [20] provide contextual embeddings that have significantly improved performance on a wide range of NLP tasks [16, 29]. Moreover, the representation of text via embeddings allows for efficient and accurate passage retrieval, using semantic similarity [12, 20].

2.2 Schema Matching

Schema matching is a crucial process in various domains such as data warehousing, E-commerce, and semantic web services. It involves identifying correspondences between different data schemas and creating a mapping between them. A schema, in its simplest definition, is a representation of the structure of a database. Schema matching, therefore, plays a vital role in data integration, enabling the consolidation of various data sources into a unified, coherent form.

Traditionally, schema matching was executed manually, requiring significant human effort. However, this approach strongly relies on the expertise of human matchers [6]. This has led to development of automated schema matching, primarily using ML algorithms [1, 5, 15]. Although automatic, previous work on ML-based schema matching was limited to very basic schemas, with most works dealing only with toy datasets [5, 8, 18]. This stemmed from a combination of lack of high-quality datasets [11], as well as relatively poor results of past methods.

Recently, following the advancements in NLP, mechanisms with enhanced understanding of language semantics have been able to achieve better results. Leveraging LLMs for schema matching has been shown to provide initial promising results, principally due to their ability to understand and interpret the semantics and context of the data schemas [13, 33].

3 ReMATCH

In this section, we present our new approach, called ReMatch, inspired by techniques from IR in general, and by LLMs in particular, for the task of schema matching. The basic steps of our approach are illustrated in Fig. 1. Below, we provide further details regarding the concepts underlying our overall methodology, as well as pseudo code of relevant algorithms.

3.1 Problem Statement

Given two database schemas, a source schema $\mathcal{S}_1$ and a target schema $\mathcal{S}_2$, with sets of tables $\mathcal{T}_1$ and $\mathcal{T}_2$ and sets of attributes $\mathcal{A}_1$

and $\mathcal{A}_2$, respectively, the schema matching task involves finding a mapping between $(\mathcal{A}_1, \mathcal{T}_1) \in \mathcal{S}_1$ to $(\mathcal{A}_2, \mathcal{T}_2) \in \mathcal{S}_2$.

The matches between their attributes are captured by a relation match: $\mathcal{P}(\mathcal{A}_1) \times \mathcal{P}(\mathcal{A}_2)$. An element $(A_1, A_2) \in match$ defines a matching pair possibly representing the same information in the schemas. If $|A_1| = 1$ and $|A_2| = 1$, we call the match an *elementary match* or 1:1 match. Otherwise, we refer to it as a *complex match* or m:n match.

The goal of automatic schema matching is to find all those matches that are meaningful. Usually, there are conditions that a collection of meaningful matches has to obey, for instance, that the matches do not overlap. That is, for every pair of matches (a_1, a_2) and (a_3, a_4) in the mapping it both holds that $a_1 \cap a_3 = \emptyset$ and $a_2 \cap a_4 = \emptyset$. We call this collection of matches a *mapping*. From a structural point of view, a mapping is an element of the power set of matches.

Formally, we would like to find a function $\Psi : \mathcal{A}_1 \rightarrow \mathcal{P}(\mathcal{A}_2)$, such that $\forall a \in \mathcal{A}_1,\ \forall a' \in \Psi(a),\ (a, a') \in match$. In other words, $\Psi(a) = A' \subseteq \mathcal{A}_2$, should contain all relevant matches for a.

A possible simplification of this problem can be made by limiting Ψ to be a $N \times K$, $N = |\mathcal{A}_1|, K \in \mathbb{N}$, matrix of elements from $\mathcal{A}_2$, denoted as Ψ_K. In the case of elementary matches, or m:1 complex matches, the goal from above now becomes to maximize the *accuracy@K* metric, i.e., to maximize:

$$\frac{1}{N} \sum^{N} \mathbf{1}_{\{\exists a',\ (a,a') \in match,\ a' \in \Psi_K(a)\}} \tag{1}$$

.

Algorithm 1: ReMatch

Inputs : a source schema $\mathcal{S}_1$, $\mathcal{T}_1$, $\mathcal{A}_1$, with all its textual descriptions; a target schema $\mathcal{S}_2$,$\mathcal{T}_2$, $\mathcal{A}_2$ with all its textual descriptions; K,$J \in \mathbb{N}$; an embedding model Φ; a generative LLM $\mathcal{F}$.

Outputs : a $N \times K$ matrix of attribute candidates from $\mathcal{A}_2$, denoted as Ψ_K.

1: $C_s \leftarrow \emptyset, C_t \leftarrow \emptyset, \Psi_K \leftarrow \{\}$

/* lines 2-4 are performed once, as a pre-processing step. */

2: $C_t \leftarrow TableToDoc(\mathcal{S}_2, \mathcal{T}_2, \mathcal{A}_2)$

3: $C_s \leftarrow AttributeToDoc(\mathcal{S}_1, \mathcal{T}_1, \mathcal{A}_1)$

4: $\tilde{C}_t \leftarrow \Phi(C_t)$

5: **foreach** $t_i \in \mathcal{T}_1$ **do**

6: $\quad \mathcal{T}_c \leftarrow \{\}$

7: $\quad$ **foreach** $a_k \in \mathcal{A}_1[t_i]$ **do**

8: $\quad\quad \mathcal{T}_c \leftarrow RankTopjTables(C_s[t_i; a_k], C_t, \tilde{C}_t, \Phi, J)$

9: $\quad$ **end**

10: $\quad \Psi_K \leftarrow CreateTopkMapping(t_i, C_s[t_i], \mathcal{T}_c, C_t[\mathcal{T}_c], \mathcal{F})$

11: **end**

3.2 Method Description

The proposed methodology leverages textual descriptions, tables and attributes constraints, and data types, for creating an alignment between source and target schemas. To this end, we look at the problem from an IR perspective. We adapt passage-based modeling such that each schema element is represented as a structured document, containing all its textual descriptions and information. We utilize this representation to judge the similarity and retrieve relevant documents. We then derive correspondences based on LLM generation, by producing prompts containing source schema attributes (columns) with relevant target schema candidates.

Algorithm 2: RankTopjTables

Inputs : a source attribute as a document $C_s[t_i; a_k]$; the corpus of target tables as documents and its embedding C_t and $\tilde{C}_t$; an embedding model Φ; $J \in \mathbb{N}$.

Outputs : a list of J retrieved target tables.

1: $\tilde{a}_{emb} \leftarrow \Phi(C_s[t_i; a_k])$

2: $scores \leftarrow Similarity(\tilde{a}_{emb}, \tilde{C}_t)$

3: $top_j \leftarrow \text{argmaxes}(\{scores_1, \ldots, scores_n\})[1 : j]$

4: **return:** $C_t[top_j]$

The methodology is composed of three stages: Given a source schema $\mathcal{S}_1$ with a set of tables $\mathcal{T}_1$ and a set of attributes $\mathcal{A}_1$, and a target schema $\mathcal{S}_2$ with a set of tables $\mathcal{T}_2$ and a set of attributes $\mathcal{A}_2$:

(1) Target schema tables and source schema attributes are first transformed into two corpora of structured documents, C_t and C_s, respectively. Each target schema table and source schema attribute is represented as a structured document consisting of four descriptive paragraphs. The title of the document is the table's name, and the opening paragraph provides an overview of the table's purpose and characteristics. The subsequent paragraphs detail the set of attributes serving as the table's primary key, the set of attributes referring to other tables (foreign keys), and the rest of the attributes belonging to this table, respectively. Each attribute is followed by its data type and a textual description. For attribute documents, the specific attribute is highlighted above the title.

(2) Given constraints on prompt size, we employ a retrieval strategy to allow for scalability to large schemas. For each attribute in the source schema, we search for the top J documents that represent candidate tables from the target schema. These tables should contain the most promising candidate attributes for deriving correspondences. This step ensures that only the most relevant documents are considered for matching, thereby coping with the limitations imposed by the maximal prompt size. To facilitate the retrieval of candidate tables, we utilize a text embedding model to encode both the candidate source attribute, accompanied by its table description, and the corpus of target table documents. These embeddings serve as a basis for measuring semantic similarity, enabling the efficient retrieval of candidate tables. Finally, for every source table $t_i \in \mathcal{T}_1$, we create a set of all top J candidate tables $\mathcal{T}_c$ retrieved.

(3) In the last step, the LLM is tasked with selecting the top K most similar target attributes from the set of retrieved tables $\mathcal{T}_c$ identified earlier. The model assesses the similarity

based on the context provided by the document representations, yielding a ranked list of K potential matches for each attribute in the source schema.

An example of a full prompt, including the outputs of stage (1) and (2) is given in Appendix A. Algorithm 1 presents the entire ReMatch method, where the functions *TableToDoc* and *AttributeToDoc* refer to the process from stage (1), and Algorithm 2 focuses on the document ranking process from stage (2).

4 EVALUATION

4.1 Dataset Creation

To evaluate our method we used two primary datasets, both involving mappings between healthcare database schemas.

MIMIC-III to OMOP For the first dataset, we created a mapping between the schema of MIMIC-III [10] and The Observational Medical Outcomes Partnership Common Data Model (OMOP)[1]. MIMIC-III is a public database containing deidentified records from patients who were admitted to the critical care units of the Beth Israel Deaconess Medical Center. OMOP is an open-source data standard, created to standardize the structure and content of healthcare related data. This scenario closely resembles real-world use cases, where a proprietary database created for a specific purpose (a source schema) is mapped to a given industry standard (a target schema) for further uses.

The mapping was created manually by a domain expert, aided by the mapping created in [23]. The full MIMIC-III schema was mapped. If no matching attribute in OMOP could be found, the attribute was assigned *NA*. For convenience we will refer to this dataset as MIMIC.

OMAP Benchmark Synthea Dataset Following [19] we used the *Synthea* [31] *to OMOP* dataset from the OMAP benchmark [33]. The dataset contains a partial mapping of the schema for Synthea (the source), a synthetic healthcare dataset and data generator, to a partial subset of relevant OMOP attributes (the target). For convenience we will refer to this dataset as Synthea.

It is important to note that although both datasets are similar in their goals, the first maps *the entire* MIMIC-III schema to OMOP, while Synthea contains only *partial* mappings. To the best of our knowledge, the MIMIC dataset is the largest single-schema to single-schema dataset published. Table 1 contains statistics detailing both datasets. The full MIMIC dataset is publicly available at https://github.com/meniData1/MIMIC_2_OMOP.

4.2 Experiments

We established the performance of ReMatch and previous state-of-the-art (SOTA) methods on MIMIC and Synthea. In this process we aimed to both assess the abilities of ReMatch, as well as compare it to methods that use labeled data.

It's important to note that while ReMatch was evaluated on the full mapping, including nulls, the evaluation of the other models focused solely on attributes with existing matches. This means that input attributes with no match in the target were not scored, which is a significant relaxation of the problem.

[1]https://www.ohdsi.org/data-standardization/

Table 1: Dataset statistics for MIMIC and Synthea. #Columns and #Tables refer to the number of columns and tables used in the dataset. #Mapped Columns refers to the total number of mappings for the source schemas, and in the case of target schemas (OMOP), the number of unique columns mapped. #Null Mappings refers to columns without a mapping (mapped to *NA*).

Dataset	#Columns	#Tables	#Mapped Columns	#Null Mappings
MIMIC	268	25	156	112
OMOP (MIMIC)	425	38	95	-
Synthea	38	8	105	-
OMOP (Synthea)	67	8	67	-

Prior methods, as explained in Section 5, approached schema matching as a binary classification problem, with the usual F1-score. Since our method treats it as a retrieval problem, the most appropriate metric is accuracy@K, as defined in Section 3.1. Unlike the F1-score, standard accuracy@K is not well defined for m:n matches. To deal with this, we limited ourselves to evaluating 1:1 or m:1 relations. Nonetheless, it is worth noting that for a dataset with only 1:1 and m:1 mappings, accuracy@1 is equivalent to F1-score when using *argmax*, instead of a threshold, for prediction (since it trivially reflects both the precision and the recall simultaneously).

It is also important to note that there are multiple dimensions we would like to optimize simultaneously. The first is maximizing accuracy@K, and is the primary goal. We also seek to achieve better results with lower values of K, as our aim is to assist human schema matching, and minimize cognitive overload [1, 3]. Additionally, we would also like to minimize the number of target tables, to reduce prompt size and therefore costs. Although choosing J does not guarantee the number of target tables in the final prompt, minimizing J does serve as a proxy for smaller prompts.

Evaluating ReMatch For all experiments we used GPT-4 [22] as our generative LLM and Ada2 [9, 20] as our embedding model. The full prompts and hyperparameters can be found in Appendix A.

We performed a grid search over various values of J and K, to establish the performance of our method. This enabled us to select hyperparameters for further evaluations.

We performed an ablation study regarding the usefulness of the first and second stage in ReMatch. We evaluated the performance on MIMIC when no descriptions are used, meaning that instead of using stage (1) in ReMatch, only the *names* of the tables and attributes are used for retrieval and generation. We also evaluated a version with no retrieval stage, i.e., for each table in the source schema we used the entire target schema.

Finally, we ran ReMatch multiple times on each of the chosen setups in the ablation study. This was done to deal with the stochastic nature of LLM text generation and to ensure the results are consistent.

Evaluating ReMatch with Guidance In real-world mapping scenarios a human matcher may receive ReMatch's output and determine that the suggestions are not good enough. Alternatively, a human matcher may decide to provide information they already have about matches. In either of these cases, ReMatch should allow the matcher to influence the output of the mechanism. This includes

Table 2: Grid search results on MIMIC.

Retrieved Documents	Acc@1	Acc@2	Acc@3	Acc@5	Acc@7	**Avg #T**
$J = 1$	0.424	0.589	0.644	0.709	0.697	2.44
$J = 2$	0.425	0.541	0.638	0.729	0.758	4.68
$J = 3$	0.321	0.477	0.533	0.657	0.733	6.64
$J = 5$	0.336	0.425	0.504	0.657	0.754	9.88
$J = 7$	0.317	0.399	0.414	0.616	0.71	12.84

highlighting the relevancy of a certain table, or explicitly providing the mapping of an attribute.

In our evaluation we investigated a mechanism, inspired by the relevance feedback technique [26], of the impact this may have on ReMatch's performance. To do so, we selected a mapping of the form $(T_1, a_1) \rightarrow (T_2, a_2)$ for a single attribute from each table. Specifically, we used the *SUBJECT_ID* attribute where available (19/25 tables), and an attribute containing unique identifiers otherwise. We then automatically included T_2 in the set of retrieved tables, $\mathcal{T}_c$, and provided the LLM with this mapping in the prompt.

Evaluating Other Models Our method's performance was evaluated against Ditto [13] and SMAT [33], two previously SOTA non-LLM models. For Ditto training we used the default hyperparameters: {`Learning Rate: 3e-5, Batch Size: 64, Epochs: 20`}, with early stopping. For SMAT training we also used the default hyperparameters: {`Learning Rate: 0.8, Batch Size: 64, Epochs: 30`}, with early stopping. In all setups, 10% of the training data was used as a validation set to avoid over-fitting. Models were trained on a single Nvidia A-100 GPU.

Both models demand a preprocessing step to generate a Cartesian product of all source and target attributes, and each pair is then labeled as positive for a match and negative otherwise. To compare ReMatch to these models we created a similar representation for our dataset (MIMIC) with a Cartesian product representation, which resulted in a positive to negative class ratio of approximately 1:400. While we utilized both MIMIC and Synthea to directly compare to ReMatch, we also used Synthea as a sanity check for verifying our ability to reproduce models at least as good as Ditto and SMAT's reported results.

We used different split ratios of training and test subsets to understand how the models perform under varying conditions and prerequisites. Namely, we tried two setups, 80% training and 20% testing, and vice versa. The splits were stratified across table names. This approach also allows for an understanding of how much preliminary work is necessary to achieve similar scores to ReMatch on the MIMIC dataset.

We also extended the evaluation to include accuracy@K. To do so we adapted the models' output to calculate rankings rather than binary predictions. We calculated the accuracy@K by determining whether the correct match received one of the K highest classification scores. Binary predictions for F1-score calculation were given by using the optimal threshold for the validation set.

4.3 Results

ReMatch Results The results of the grid search for MIMIC can be found in Table 2. As explained in Section 4.2, the highest value in a given column or row is not necessarily the optimal choice. Rather, we found $\{(J = 1,\ K = 1),\ (J = 2,\ K = 5)\}$ to be the best balance points between the three optimization dimensions for the MIMIC dataset. Note the strong, albeit non-linear, correlation between J and the average number of tables (Avg #T) retrieved in practice. This is in contrast to accuracy@K, which seems to perform best for $J \leq 2$. This may imply that the retrieval is efficient, capturing all relevant tables at low values of J, and additional tables only add 'noise' to the matching prompt.

Table 3: Grid search results on Synthea. $J = \infty$ means skipping the retrieval step.

Retrieved Documents	Acc@1	Acc@3	Acc@5	**Avg #T**
$J = 1$	0.562	0.6	0.6	1.125
$J = 2$	0.505	0.581	0.743	2.625
$J = \infty$	0.438	-	0.924	8

Table 4: Results on MIMIC for ablation study with standard deviations across runs. Baseline is the full ReMatch method, names-only refers to skipping the document creation step, and $J = \infty$ means skipping the retrieval step.

Variation	Acc@1	Acc@5	**Avg #T**
$J = 1$ *baseline*	0.424 ± 0.017	-	2.44
$J = 2$ *baseline*	-	0.729 ± 0.023	4.68
$J = 1$ *names-only*	0.396 ± 0.005	-	1.48
$J = 2$ *names-only*	-	0.503 ± 0.008	3.28
$J = \infty$	0.311 ± 0.0157	0.518 ± 0.03	38

Table 5: Results for MIMIC with guidance provided. Relative improvement over baseline (in percentage) in bold.

Variation	Acc@1	Acc@5	**Avg #T**
$J = 1$	0.539/**20.38%**	0.783/**10.41%**	2.84
$J = 2$	0.459/**7.89%**	0.765/**4.86%**	4.92

The results for the grid search for Synthea are presented inTable 3. While $\{(J = 1,\ K = 1),\ (J = 2,\ K = 5)\}$ still appear to work well, for Synthea we found that not using retrieval yielded optimal results for accuracy@5. This may be explained by the small size of the dataset, with only 8 tables in the target. It is still interesting to note that ReMatch achieved almost 75% accuracy@5 also with $J = 2$. While the results are a bit higher for Synthea, the gap in performance between datasets is relatively small.

The results of the ablation study are shown in Table 4. We found that skipping the retrieval step and adding all of the tables yields inferior results, in alignment with the results from the grid search. We also found that using only the names of the tables and attributes, with no descriptions, results in a 6.5% decrease in accuracy@1 and a 31% decrease in accuracy@5. A possible explanation for this may be that 'easy matches' can be done based on name semantics alone, but more complex, less likely, matches require additional information. Additionally, Table 4 demonstrates that although ReMatch uses generative modeling, the method's outputs are consistent.

Table 6: Results for Ditto and SMAT on both datasets.

(a) MIMIC

Variation	Acc@1	Acc@5	F1
Ditto 80-20	0.0	0.03	0.011
Ditto 20-80	0.0	0.061	0.012
SMAT 80-20	0.454	1.0	0.102
SMAT 20-80	0.077	0.361	0.084

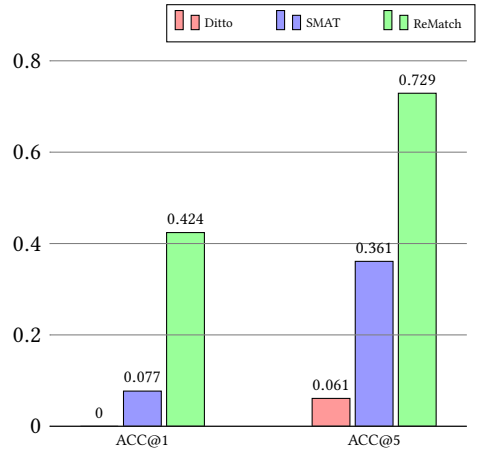
(b) Synthea

Variation	Acc@1	Acc@5	F1
Ditto 80-20	0.05	0.08	0.29
Ditto 20-80	0.02	0.12	0.06
SMAT 80-20	0.875	1.0	0.483
SMAT 20-80	0.271	0.694	0.241

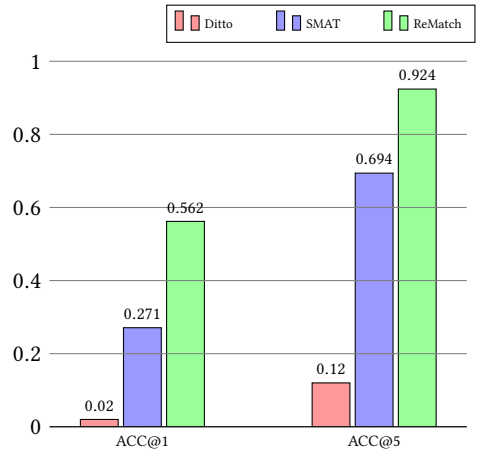
Adding guidance to ReMatch indeed improves performance, as can be seen in Table 5. The largest improvements are for $J = 1$, achieving the highest overall scores for MIMIC. This is consistent with the rest of the results, with more efficient retrieval being correlated with better matching accuracy.

SMAT and Ditto Results SMAT's performance was successfully validated using the F1-score, yielding an F1-score of 0.48276 for the 80-20 train-test split on the Synthea dataset, indicating successful model training when compared to the original values reported by Zhang et al. The SMAT model outperformed the Ditto model across all metrics and setups. Findings from the Ditto model revealed poor performance in all schema matching setups. Notably, in all scenarios, accuracy@1 was ≤ 0.05.

When using 80% of the data for training, SMAT achieved a perfect accuracy@5 score (accuracy@5 = 100%) on the remaining 20% of the attributes. Unfortunately, in real-world scenarios only a small portion of mappings can be available, since mapping 80% manually removes the need for automatic mapping. In the much more realistic split, with only 20% of the data, SMAT achieved significantly lower results than ReMatch on both datasets, as is emphasized in Fig. 2. Unlike ReMatch, the results were significantly higher on Synthea, when compared to MIMIC dataset, for both splits. This finding is aligned with our expectations about the difficulty of MIMIC, and demonstrates the robustness of ReMatch.



(a) Performance on MIMIC.



(b) Performance on Synthea.

Figure 2: Comparison of the different models' performance. Ditto and SMAT were trained and evaluated on 20%, 80% of the data, respectively, after removing all null mappings. ReMatch was evaluated on the entire dataset, with no guidance, and with nulls. Optimal setup from grid search is shown for ReMatch.

5 RELATED WORK

Prior work has dealt with various ML based matchers to tackle the schema matching task, but these have often resulted in low-quality matches or necessitated extensive manual annotation [1].

More recent mechanisms often employed similarity matrices between schemas, with various ML or deep learning (DL) components [13, 18, 28], but these were limited to small datasets, like Purchase Orders [5], or Web Forms [7], and focused primarily on entity matching rather than schema matching. More recent work in the field such as SMAT [33] and LSM [34] has been proposed, leveraging recent improvements in natural language processing techniques to obtain semantic mappings between source and target schemas. These approaches improved previous results, but still demanded that a large percentage of the data be tagged, making real-world usage impractical.

LLMs have been applied to various data preprocessing tasks, including error detection, data imputation, entity matching, as well as schema matching [17, 19]. These models have shown promising results, with LLMs achieving good results on various datasets, with no additional training involved. However, these approaches still faced challenges related to computational expense and inefficiency, as well as lacking evaluation on data that represents real-world schema matching challenges.

Finally, it is important to note that the majority of previous methods dealt with schema-matching as a binary classification task over the full Cartesian-product $\mathcal{A}_1 \times \mathcal{A}_2$. While allowing for simplified training objectives and evaluations, in normalized schemas positive labels scale by $O(n)$, whereas negative labels scale by $O(n^2)$. This is especially problematic when inference is expensive, as in the case of LLMs.

6 DISCUSSION

Schema matching is a complex task challenged by heterogeneity across schemas, semantic ambiguity of schema elements, and the

scale and complexity of large enterprise systems. Accurate matching is further complicated by the need for domain-specific knowledge, limited context for interpreting schema elements, variability in database design practices, and data privacy concerns that may restrict access to the actual data.

Our evaluation has shown that ReMatch achieves good accuracy results in practice. This was done with no labels being provided, on a large schema matching dataset, while maintaining efficiency in prompt size. We found that ReMatch can be further improved by incorporating additional information in the form of guidance. Additionally, we have shown that other methods indeed demand significant amounts of annotated data to achieve good performance, demonstrating the impracticality of direct usage in real-world scenarios.

Limitations While the ReMatch method presents an innovative approach to schema matching, and has demonstrated significant improvements in matching capabilities and outperforms other machine learning approaches, it is important to acknowledge its limitations.

Similar to human matchers, the effectiveness of ReMatch is contingent upon the quality of schema documentation. The approach relies on the presence of meaningful labels and textual descriptions for tables and attributes within the schema. In cases where schema documentation is poor or lacks clarity, ReMatch might struggle to perform optimally.

While we have successfully demonstrated high accuracy using the ReMatch method, we acknowledge that we have not yet delved into the influence of varying document structures and prompt designs on the matching results. The current implementation utilizes a specific document format and prompt configuration, which has proven to be effective; however, it is conceivable that alternative configurations may further optimize performance. This aspect represents an area for potential refinement rather than a significant drawback. Future investigations into these variables may provide incremental improvements and contribute to the fine-tuning of our approach, ensuring that ReMatch remains at the forefront of schema matching solutions.

7 CONCLUSIONS

In this work we introduced ReMatch, a scalable and efficient method for matching schemas using retrieval-enhanced LLMs. It starts by representing source schema attributes and target schema tables as structured documents. For each source attribute, the method retrieves the most semantically similar target table documents, constructing a candidate set for potential mapping. Subsequently, a prompt comprising these candidates is formulated, and an LLM is employed to identify the top matches for each source attribute.

ReMatch avoids the need for predefined mapping, any model training, or access to data in the source database. Instead, it exploits the generative abilities of LLMs, and their text comprehension to perform semantic ranking between two schemas, in alignment with a human matching process.

ReMatch is not designed to completely replace human matchers. It aims to complement their efforts by acting as a schema matching Copilot, minimizing human error and aiding matchers throughout their work.

Future Work Adjusting the structure of documents and the prompts used with LLMs to better suit specific domains could significantly improve the accuracy and relevance of the matches produced by ReMatch. This could involve tailoring the language and context within the prompts to align with the specific jargon and data relationships found in each industry. By fine-tuning these aspects, the method can become more sensitive to the nuances of each industry's data representation needs.

Moreover, in situations where the source schema data is available and can be accessed without privacy or security constraints, this data can be helpful for enrichment of labels and the potential for generating more informative labels in cases where existing documentation is sparse or ambiguous.

Finally, further research is needed regarding combining ReMatch with additional algorithmic improvements. These may include explicit usage of type constraints, foreign keys, and primary keys, both directly and as post-processing steps, as well as additional usage of guidance mechanisms. New methods combining ReMatch, for initial labeling of the majority of the data, with other methods for downstream inference, may be of interest. Such an approach could leverage the advantages of each method to achieve even better results.

REFERENCES

[1] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. 2011. Generic schema matching, ten years later. *Proceedings of the VLDB Endowment* 4 (2011), 695 – 701. https://api.semanticscholar.org/CorpusID:6302654

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *ArXiv* abs/2005.14165 (2020). https://api.semanticscholar.org/CorpusID:218971783

[3] Alexander Chernev, Ulf Böckenholt, and Joseph K. Goodman. 2015. Choice overload: A conceptual review and meta-analysis. *Journal of Consumer Psychology* 25 (2015), 333–358. https://api.semanticscholar.org/CorpusID:46655935

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv* abs/1810.04805 (2019).

[5] Hong Hai Do and Erhard Rahm. 2002. COMA - A System for Flexible Combination of Schema Matching Approaches. In *Very Large Data Bases Conference.* https://api.semanticscholar.org/CorpusID:9318211

[6] Zlatan Dragisic, Valentina Ivanova, Patrick Lambrix, Daniel Faria, Ernesto Jiménez-Ruiz, and Catia Pesquita. 2016. User Validation in Ontology Alignment. In *International Workshop on the Semantic Web.* https://api.semanticscholar.org/CorpusID:1578751

[7] Avigdor Gal. 2011. Uncertain schema matching: the power of not knowing. In *International Conference on Information and Knowledge Management.* https://api.semanticscholar.org/CorpusID:43482147

[8] Avigdor Gal, Haggai Roitman, and Roee Shraga. 2021. Learning to Rerank Schema Matches. *IEEE Transactions on Knowledge and Data Engineering* 33 (2021), 3104–3116. https://api.semanticscholar.org/CorpusID:143427155

[9] Ryan Greene, Ted Sanders, Lilian Weng, and Arvind Neelakantan. 2022. New and improved embedding model. https://openai.com/blog/new-and-improved-embedding-model

[10] Alistair E W Johnson, Tom J Pollard, Lu Shen, Li-Wei H Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3, 1 (2016), 1–9.

[11] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2020. Valentine: Evaluating Matching Techniques for Dataset Discovery. *2021 IEEE 37th International Conference on Data Engineering (ICDE)* (2020), 468–479. https://api.semanticscholar.org/CorpusID:222378204

[12] 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 (2020), 9459–9474.

[13] 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 (2020), 50 – 60. https://api.semanticscholar.org/CorpusID:214743579

[14] Yi-Hsien Liu, Tianle Han, Siyuan Ma, Jia-Yu Zhang, Yuanyu Yang, Jiaming Tian, Haoyang He, Antong Li, Mengshen He, Zheng Liu, Zihao Wu, Dajiang Zhu, Xiang Li, Ning Qiang, Dingang Shen, Tianming Liu, and Bao Ge. 2023. Summary of ChatGPT-Related Research and Perspective Towards the Future of Large Language Models. *Meta-Radiology* (2023). https://api.semanticscholar.org/CorpusID:257921533

[15] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. 2001. Generic Schema Matching with Cupid. In *Very Large Data Bases Conference*. https://api.semanticscholar.org/CorpusID:1456533

[16] Shervin Minaee, E. Cambria, and Jianfeng Gao. 2021. Deep Learning Based Text Classification: A Comprehensive Review. https://api.semanticscholar.org/CorpusID:235386502

[17] Suvir Mirchandani, F. Xia, Peter R. Florence, Brian Ichter, Danny Driess, Montse Gonzalez Arenas, Kanishka Rao, Dorsa Sadigh, and Andy Zeng. 2023. Large Language Models as General Pattern Machines. *ArXiv* abs/2307.04721 (2023). https://api.semanticscholar.org/CorpusID:259501163

[18] Sidharth Mudgal, Han Li, Theodoros Rekatsinas, AnHai Doan, Youngchoon Park, Ganesh Krishnan, Rohit Deep, Esteban Arcaute, and Vijay Raghavendra. 2018. Deep Learning for Entity Matching: A Design Space Exploration. *Proceedings of the 2018 International Conference on Management of Data* (2018). https://api.semanticscholar.org/CorpusID:44063437

[19] Avanika Narayan, Ines Chami, Laurel J. Orr, and Christopher R'e. 2022. Can Foundation Models Wrangle Your Data? *Proc. VLDB Endow.* 16 (2022), 738–746. https://api.semanticscholar.org/CorpusID:248965029

[20] Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas A. Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David P. Schnurr, Felipe Petroski Such, Kenny Sai-Kin Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and Code Embeddings by Contrastive Pre-Training. *ArXiv* abs/2201.10005 (2022). https://api.semanticscholar.org/CorpusID:246275593

[21] OpenAI. 2022. Introducing chatgpt. https://openai.com/blog/chatgpt/

[22] OpenAI. 2023. GPT-4 Technical Report. *ArXiv* abs/2303.08774 (2023). https://api.semanticscholar.org/CorpusID:257532815

[23] Nicolas Paris, Antoine Lamer, and Adrien Parrot. 2021. Transformation and Evaluation of the MIMIC Database in the OMOP Common Data Model: Development and Usability Study. *JMIR Medical Informatics* 9 (2021). https://api.semanticscholar.org/CorpusID:244194789

[24] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving Language Understanding by Generative Pre-Training. https://api.semanticscholar.org/CorpusID:49313245

[25] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. https://api.semanticscholar.org/CorpusID:160025533

[26] J. J. Rocchio. 1971. Relevance feedback in information retrieval. https://api.semanticscholar.org/CorpusID:61859400

[27] Roee Shraga, Ofra Amir, and Avigdor Gal. 2021. Learning to Characterize Matching Experts. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. 1236–1247. https://doi.org/10.1109/ICDE51399.2021.00111

[28] Roee Shraga, Avigdor Gal, and Haggai Roitman. 2020. ADnEV: Cross-Domain Schema Matching using Deep Similarity Matrix Adjustment and Evaluation. *Proc. VLDB Endow.* 13 (2020), 1401–1415. https://api.semanticscholar.org/CorpusID:214588544

[29] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to Fine-Tune BERT for Text Classification?. In *China National Conference on Chinese Computational Linguistics*. https://api.semanticscholar.org/CorpusID:153312532

[30] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Neural Information Processing Systems*. https://api.semanticscholar.org/CorpusID:13756489

[31] Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. 2018. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association* 25, 3 (2018), 230–238.

[32] Haochen Zhang, Yuyang Dong, Chuan Xiao, and M. Oyamada. 2023. Large Language Models as Data Preprocessors. *ArXiv* abs/2308.16361 (2023). https://api.semanticscholar.org/CorpusID:261397017

[33] Jing Zhang, Bonggun Shin, Jinho D. Choi, and Joyce Ho. 2021. SMAT: An Attention-Based Deep Learning Solution to the Automation of Schema Matching. *Advances in databases and information systems. ADBIS* 12843 (2021), 260–274. https://api.semanticscholar.org/CorpusID:237207055

[34] Yunjia Zhang, Avrilia Floratou, Joyce Cahoon, Subru Krishnan, Andreas C. Müller, Dalitso Banda, Fotis Psallidas, and Jignesh M. Patel. 2023. Schema Matching using Pre-Trained Language Models. *2023 IEEE 39th International Conference on Data Engineering (ICDE)* (2023), 1558–1571. https://api.semanticscholar.org/CorpusID:255188911

A FULL MATCHING PROMPT

A.1 Prompt

You are an expert in databases, and schema matching at top k specifically. Your task is to create matches between source and target tables and attributes. For each attribute from the source you always suggest the top 2 most relevant tables and columns from the target. You are excellent at this task.
If none of the columns are relevant, the last table and column should be "NA", "NA". This value may appear only once per mapping!
Your job is to match the schemas. You never provide explanations, code or anything else, only results. Below are the two schemas.
Create top k matches between source and target tables and columns.
Make sure to match the entire input. Make sure to return the results in the following json format with top 2 target results foreach input in source.

Expected output format:
{'1': {'SRC_ENT': 'SOURCE_TABLE_NAME', 'SRC_ATT': 'SOURCE_COLUMN_NAME',
'TGT_ENT1': 'TARGET_TABLE_NAME1', 'TGT_ATT1': 'TARGET_COLUMN_NAME1', 'TGT_ENT2': 'TARGET_TABLE_NAME2', 'TGT_ATT2': 'TARGET_COLUMN_NAME2'},
'2': {'SRC_ENT': 'SOURCE_TABLE_NAME', 'SRC_ATT': 'SOURCE_COLUMN_NAME',
'TGT_ENT1': 'TARGET_TABLE_NAME1', 'TGT_ATT1': 'TARGET_COLUMN_NAME1', 'TGT_ENT2': 'TARGET_TABLE_NAME2', 'TGT_ATT2': 'TARGET_COLUMN_NAME2'}} ...

Source Schema:
,SRC_ENT, SRC_ATT
0,ADMISSIONS, SUBJECT_ID
1,ADMISSIONS, HADM_ID

###16 more rows###

ADMISSIONS
The ADMISSIONS table gives information regarding a patient's admission to the hospital.
Primary Keys:
HADM_ID (INTEGER): Each row of this table contains a unique HADM_ID, which represents a single patient's admission to the hospital. HADM_ID ranges from 1000000 - 1999999.
Foreign Keys:
SUBJECT_ID (INTEGER): The ADMISSIONS table can be linked to the PATIENTS table using SUBJECT_ID. References to [PATIENTS, SUBJECT_ID]
Other Columns:
ADMITTIME (TIMESTAMP): provides the date and time the patient was admitted to the hospital

###15 more rows###

Target Schema:
,TGT_ENT,TGT_ATT
0,PERSON,person_id
1,PERSON,gender_concept_id

###17 more rows###

19,VISIT_OCCURRENCE,visit_occurrence_id
20,VISIT_OCCURRENCE,person_id

###38 more rows listing all retrieved target tables###

PERSON
This table serves as the central identity management for all Persons in the database. It contains records that uniquely identify each person or patient, and some demographic information.
Primary Keys:
person_id (bigint): It is assumed that every person with a different unique identifier is in fact a different person and should be treated independently.
Foreign Keys:
gender_concept_id (integer): This field is meant to capture the biological sex at birth of the Person. This field should not be used to study

gender identity issues. References to [CONCEPT, concept_id]

```
###76 more rows describing all target tables###
```

Remember to match the entire input. Make sure to return only the results!

A.2 GPT-4 Hyperparameters

The model version used for generation was GPT-4-1106, with the following settings:

```
{seed=42 temperature=0.5, max_tokens=4096, top_p=0.9, frequency_penalty=0, presence_penalty=0}
```