Schema Matching with Large Language Models: an Experimental Study

Marcel Parciak
UHasselt, BIOMED & DSI
Diepenbeek, Belgium
marcel.parciak@uhasselt.be

Brecht Vandevoort
UHasselt, DSI
Diepenbeek, Belgium
brecht.vandevoort@uhasselt.be

Frank Neven
UHasselt, DSI
Diepenbeek, Belgium
frank.neven@uhasselt.be

Liesbet M. Peeters
UHasselt, BIOMED & DSI
Diepenbeek, Belgium
liesbet.peeters@uhasselt.be

Stijn Vansummeren
UHasselt, DSI
Diepenbeek, Belgium
stijn.vansummeren@uhasselt.be

ABSTRACT

Large Language Models (LLMs) have shown useful applications in a variety of tasks, including data wrangling. In this paper, we investigate the use of an off-the-shelf LLM for schema matching. Our objective is to identify semantic correspondences between elements of two relational schemas using only names and descriptions. Using a newly created benchmark from the health domain, we propose different so-called task scopes. These are methods for prompting the LLM to do schema matching, which vary in the amount of context information contained in the prompt. Using these task scopes we compare LLM-based schema matching against a string similarity baseline, investigating matching quality, verification effort, decisiveness, and complementarity of the approaches. We find that matching quality suffers from a lack of context information, but also from providing too much context information. In general, using newer LLM versions increases decisiveness. We identify task scopes that have acceptable verification effort and succeed in identifying a significant number of true semantic matches. Our study shows that LLMs have potential in bootstrapping the schema matching process and are able to assist data engineers in speeding up this task solely based on schema element names and descriptions without the need for data instances.

VLDB Workshop Reference Format:
Marcel Parciak, Brecht Vandevoort, Frank Neven, Liesbet M. Peeters, and Stijn Vansummeren. Schema Matching with Large Language Models: an Experimental Study. VLDB 2024 Workshop: Tabular Data Analysis Workshop (TaDA).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/UHasselt-DSI-Data-Systems-Lab/code-schema-matching-LLMs-artefacs.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

1 INTRODUCTION

Schema matching [17] constitutes a core task in data integration [6]. It refers to the problem of identifying semantic correspondences between elements of two relational schemas that represent the same real-world concept. For example, a schema matching system may conclude that an attribute `admittime` from one table in a medical information system semantically corresponds to an attribute `visit_start_date` in another table. Once correspondences are identified, they can be used to translate data from the source schema into data conforming to the target schema [6], a process known as *schema mapping*. In this paper, we focus on schema matching.

Schema matching systems are software systems that help data engineers perform schema matching. They generate a set of *match candidates* (i.e., candidate correspondences) which the data engineer can accept, reject or edit in order to obtain a final set of matches [17]. To generate match candidates, a wide variety of signals that hint at element correspondence have been considered in the research literature. These include syntactic similarity of attribute names; consulting thesauri; looking at data values and their distributions in concrete database instances; and exploiting database constraints [1, 2, 5, 17]. Unfortunately, many such signals remain unavailable in real-world schemas [12]: attribute names are often cryptic and involve domain-specific abbreviations not occurring in thesauri. Use of actual data values and concrete database instances may be restricted for legal reasons; e.g., this is the case in the health domain where real database instances are problematic to obtain due to privacy constraints. In the absence of available real instances, one may consider leveraging synthetically-generated instances to aid in matching. However, accurately replicating the complexity and subtle patterns of real medical data is highly challenging and time-consuming, and rigorous validation of generated schema matches is hindered by the lack of a true ground truth. In the health domain setting, it is hence vital to be able to generate match candidates with as little information as possible.

In the healthcare data integration context, we found that, despite its restrictions, we often have schema documentation in the form of data dictionaries available, as well as natural-language descriptions of some schema elements. In particular, target schemas are often *common data models*: data schemas designed by community consensus that harmonize healthcare data [14]. These data models

are well documented, explaining the semantics of schema elements in detail.

In this paper, we aim to exploit this information and present an experimental study on schema matching using an off-the-shelf generative *Large Language Model* (LLM). We investigate how LLMs can be prompted to generate a set of match candidates. We focus on the use of schema documentation as the sole signal and evaluate the performance against a newly defined real-world benchmark.

We define different *task scopes* for doing LLM-based schema matching. Task scopes are prompting methods they vary primarily in the amount of context information contained in the prompt. Using these task scopes, we aim to answer the following research questions:

(1) How does the quality of LLM-based schema matching vary among different task scopes and LLM models, and how does it compare to a string-similarity-based baseline?
(2) How decisive are LLMs in expressing their opinions on attribute pairs, and how does this affect their reliability and consistency?
(3) What is the extent of the complementarity between the match results for different task scopes and the baseline?
(4) Is it useful and practical to combine different LLM-based and/or string-similarity-based matchings?

To answer these questions, we introduce the schema matching task and experimental setup in Section 2. There, we also introduce the different task scopes. We then present and discuss our findings w.r.t. the first two research questions in Section 3.1 and investigate the last two questions in Section 3.2. We conclude in Section 4.

Related Work. We traditionally perform schema matching by exploiting signals such as syntactic similarity of attribute names; thesauri; data values and distributions; and database constraints [1, 2, 5, 17]. In this work, we are interested in schema matching using LLMs in more restricted settings where, except for schema documentation (i.e., the attribute names and their natural-language descriptions) these signals remain unavailable.

Dataset discovery is the process of navigating a collection of data sources in order to find datasets that are relevant for a task at hand, as well as the relationships among those datasets. It has been observed that schema matching is a critical component in dataset discovery, and that many dataset discovery systems implement their own schema matcher [10, 11]. Indeed, conceptually, one can also see dataset discovery as generalizing schema matching. Like traditional schema matching algorithms, however, dataset discovery algorithms will aim to exploit a rich variety of signals to do the discovery, including access to the actual data instances. By contrast, in this work, we are interested in schema matching using LLMs in the setting where, except for schema documentation, such signals are unavailable.

LLMs are general machine learning models trained on large and generic natural text data, such as the web. They are able to solve a variety of tasks with no or minimal fine-tuning effort [3]. In the field of data management, LLMs have shown promising results for data wrangling tasks such as error detection and data imputation [13]. However, except for [22], they have not been widely applied to schema matching, yet.

Zhang et al [22] also use language models for instance-free schema matching, but employ and fine-tune an encoder-only model (BERT). By contrast, we use an off-the-shelf generative decoder-only model (GPT) without any need for fine-tuning.

Also related is SMAT [21] which uses an attention-based neural network to match GLoVe embeddings [16] of schema elements, but requires a majority of the data to be labelled: 80% of the data that needs to be matched is used for training, and subsequently an additional 10% is used for tuning weights, leaving only 10% to evaluate the model. For practical applications, this presents a significant limitation, as requiring 90% of the input schemas to be labeled, amounts to almost completely matching the schemas by hand. Our approach, however, does not require any labelled data, allowing an off-the-shelf usage.

AdnEV [18] proposes a methodology based on deep learning and weak supervision to adjust and combine different schema matching algorithms. In this work, we observe that it makes sense to combine different task scopes to achieve the greatest effectiveness. It is an interesting direction for future work whether approaches such as AdnEV can be used to make this combination even more effective.

2 METHODS

Schema Matching. For the purpose of this paper, a *schema* refers to a relational schema, i.e., a finite set of *attributes*. A *1:1 match* between two schemas S_1 and S_2 is a pair $(a, b) \in S_1 \times S_2$ that is meant to indicate that there is a semantic correspondence between attribute $a \in S_1$ and $b \in S_2$. Because in the schema mapping phase we should be able to unambiguously map data values of attribute a into data values of attribute b (and vice versa) we say that (a, b) is a *(semantically) valid 1:1 match* if there exists an invertible function mapping values of a into values of b. We define *schema matching* to be the problem of deriving a set of valid 1:1 matches between two given schemas.[1] We note that in the literature, also matches of kind $1 : m$, $n : 1$ and $n : m$ exist. For example, in a match of kind $1 : m$ we may relate a single attribute a in S_1 to a *set* of attributes $B \subseteq S_2$, meaning that the information of a-values in S_1 will be "distributed" among all the attributes in B and that we need all attributes in B to recover the a-value. A typical example is relating Name in S_1 to $B = \{\text{First name}, \text{Last name}\}$. In this paper, we restrict ourselves to 1:1 matches for two reasons. First, this shrinks the search space significantly for possible matches, making our experimental approach feasible even for larger schemas. Second, it allows us to compare our results to a baseline using string similarity measures, which are difficult to extend to $1 : m$, $n : 1$ or $n : m$ matches.

Benchmark. In order to gauge the suitability of LLMs for schema matching we have created a new benchmark, situated in the healthcare domain. We draw source schemas from the MIMIC-IV dataset [7] and target schemas from OHDSI OMOP Common Data Model [14]. Both are public, well-known data models in the medical informatics community. The OHDSI community maintains an ETL process to transform data from MIMIC-IV to OMOP [8]. We use this ETL specification to manually identify all semantically valid 1:1 matches that will serve as the ground truth. That is, we manually inspect all

[1] In this paper, we assume that the source and target table are already provided, the table matching step, i.e., identifying corresponding tables, has thus already been completed.

applied ETL transformations and derive each attribute combination (a, b) where a single value from the source attribute a is sufficient to determine the value from the target attribute b and vice versa. For example, the attribute `gender` from MIMICs `Patients` table is mapped to both `gender_concept_id` and `gender_source_value` of OMOPs `Person` table. Both mappings are valid 1:1 matches, as the value in `gender` can be mapped to a valid value fit for either attribute and vice versa. In contrast, the attribute `admittime` of MIMICs `Admissions` table is not a valid match for `visit_start_datetime` of OMOPs `Visit_Occurrence` table, as the ETL specification needs to combine it with another attribute to determine the value of `visit_start_datetime`.

We have extracted a total of 49 valid 1:1 matches between 7 relations from MIMIC-IV and 6 relations from OMOP. In total, there are 9 relation pairs that contain at least one semantic match. We will refer to each of these relation pairs as a *dataset* in our benchmark. Our 9 datasets create a search space of 1839 attribute pairs that contain 49 true semantic 1:1 matches as summarized in Table 1. We consider all other attribute pairs as non-matches. The schema matching problem is hence highly imbalanced. For each (source or target) table and each attribute we extract the name as well as a natural language description from respective documentations. Our benchmark is publicly available in our artefacts repository [15].

We acknowledge that a benchmark consisting of public datasets is probably contained in the training data of an LLM trained on the web. As an example, when asking ChatGPT to give a description of the attribute `dischtime` from the `admissions` table in MIMIC-IV, the answer returned from the model fits the description given in the official documentation of MIMIC-IV well. This represents a limitation of our experimental setup. We argue that although the datasets are known to the LLM, the true semantic matches to transform data from MIMIC-IV to OMOP are not readily explicitly available: significant effort is required to extract them from the ETL scripts.

Prompt Engineering. Generative LLMs are trained to answer questions in natural language. As such, we need to interface with the LLM via prompts that describe the task to be performed by the LLM as well as the table and attribute names and descriptions. Previous research into prompt engineering has proposed a number of *prompt engineering patterns* that positively influence answer quality [13]. We next discuss how we have applied these common practices in our prompt design by means of the visual representation in Figure 1. Each prompt is always applied to a single source schema and a single target schema (plus their descriptions), and consists of four sections referred to as *Introduction*, *Source Information*, *Target Information*, and *Task Description*.

First, we introduce the schema matching problem to the LLM by utilizing the *Persona Pattern* to let the LLM act as a schema matcher [20]. We then introduce our definition of a valid 1:1 match using the *Meta Language Creation* pattern [20]. Both patterns are illustrated in the *Introduction* section in Figure 1.

Subsequently, we serialize the schema information, including table and attribute descriptions, using a serialization inspired by [13]. Concretely, we first serialize the source information, followed by the target information. An example of this can be viewed in Figure 1 in the *Source Information* and *Target Information* sections.

We finalize our prompts with the task description that utilizes the phrase "Lets think step by step" which has been shown to increase performance by instructing the LLM to build up a step-by-step argument in the output [9] and to which we refer as the *Chain of Thought Pattern* in Figure 1. We end the task description with the *Output Automater* pattern to instruct the model to output structured data (in particular: JSON) for further processing [20]. Here, we ask the LLM to generate a structured output such that we can extract (a, b, out) triples, where a and b are attributes from the source and target schema, respectively, and *out* (discussed further below) is the LLM's opinion of whether (a, b) is a semantically meaningful 1:1 match. Both patterns are illustrated in the example prompt in Figure 1 in section *Task Description*.

During our experiments, we found that using a three-step scale for *out* works best. We ask the LLM to use `yes` for a match, `no` for a non-match, and `unknown` if there is not enough information to decide. We have also experimented with numerical scores, which were difficult to interpret, and five-step scales, which were prone to hallucinations. For example, asking for a five-step scale of *no correspondence*, *low correspondence*, *medium correspondence*, *high correspondence* and *very high correspondence* frequently resulted in opinions such as *low to medium correspondence*, making a reliable interpretation highly difficult. We note that LLM output is not necessarily complete: there may be attribute pairs (a, b) for which the LLM does not give its opinion; we treat this as `unknown`.

Task Scopes. In this paper, we focus on a comparison of *task scopes*, which we define as the amount of schema information contained in a single prompt. We define four different scopes:

- **1-to-1** Each prompt contains exactly one attribute from source and one from target.
- **1-to-N** Each prompt contains a single attribute from the source schema and N attributes of the target schema, where N refers to the total number of attributes in the target schema.
- **N-to-1** Each prompt contains N attributes from the source schema and a single attribute from the target schema, where N refers to the total number of attributes in the source schema.
- **N-to-M** Each prompt contains N attributes from the source schema and M attributes from the target schema, where N and M refer to the total number of attributes in the source and target schema, respectively.

It is worth noting that the task scope choice has implications on the complexity to parse structured votes from the LLM output. While we expect a single vote (e.g. `yes` or `no`) in the `1-to-1` case, an output to the `N-to-M` task scope potentially contains $N \times M$ votes, one for each attribute pair.

We investigate both `1-to-N` and `N-to-1` as both scopes present very different contexts to the LLM. In the former, the LLM is presented with all available information about the target relation while limiting the information of the source information, and vice versa in the latter. We found that this difference impacts the quality of the matches.

String Similarity Baseline. We aim to compare the performance of the LLM-based approaches against a baseline based on a string similarity measure, a well-established baseline approach in the field

Table 1: Benchmark datasets: names of source and target tables, their respective attributes and attribute pair counts, and the number of true semantic matches.

dataset	Source	|source|	target	|target|	|pairs|	|matches|
AdCO	Admissions	16	Condition_Occurrence	16	256	2
AdVD	Admissions	16	Visit_Detail	19	304	5
AdVO	Admissions	16	Visit_Occurrence	17	272	8
DiCO	Diagnoses_ICD	5	Condition_Occurrence	16	80	2
LaMe	Labevents	10	Measurement	20	200	10
PaPe	Patients	6	Person	18	108	5
PrDE	Prescriptions	17	Drug_Exposure	23	391	6
SeVD	Services	5	Visit_Detail	19	95	5
TrVD	Transfers	7	Visit_Detail	19	133	6
				Total	1839	49

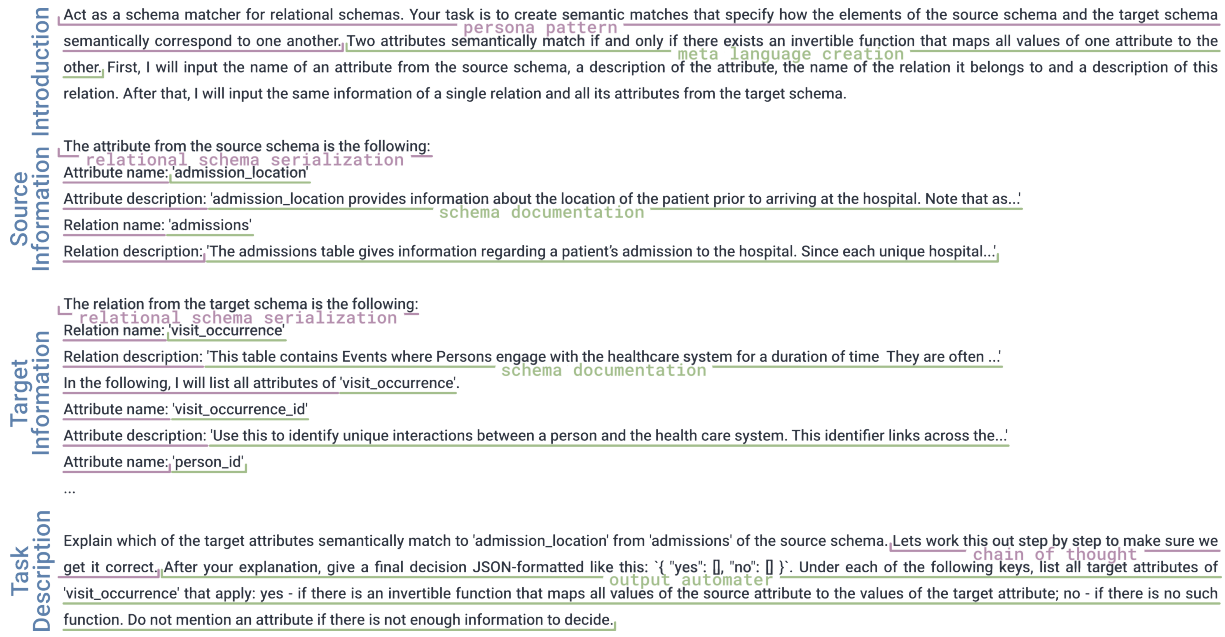

Figure 1: A truncated example of a `1-to-N` prompt. The prompt engineering best practices applied are highlighted.

of schema mapping and ontology alignment [4, 19]. To do so, we have selected edit distance-based metrics investigated by [19] and [4] and checked for their availability in the common Python library `textdistance`[2]. We aim to find commonly used similarity metrics that are readily available and identified four metrics: Jaro Winkler, Levenshtein, Monge Elkan and N-gram. We evaluated these metrics on our benchmark by calculating the similarities between the attribute names for each attribute combination in the benchmark. It is important to note that these attribute pairs are the same as those used in our results, although we do not report dataset-specific values here. We then generate a ranking of all attribute pairs and calculate the precision and recall for each threshold per similarity measure. Figure 2 displays the corresponding precision-recall curve and reveals that N-gram with $n = 3$ is the best performing metric (w.r.t. the area under curve). This string similarity metric will therefore be used in the following as a baseline.

Specifically, to obtain the baseline we calculate the N-gram string similarity $sim_{NG}(a, b)$ between all possible attribute pairs in a dataset. For each attribute name a' we obtain the set of its 3-grams a after padding with special characters as described by Sun et al. [19]. For example, the name `admittime` is transformed into the set {`##a`, `#ad`, `adm`, `dmi`, `min`, `int`, `ntt`, `tti`, `tim`, `ime`, `me%`, `e%%`}. For two sets a and b, we then calculate the Dice similarity:

$$sim_{NG}(a, b) := \frac{2 \times |a \cap b|}{|a| + |b|}$$

Since each $sim_{NG}(a, b)$ lies in the range $[0; 1]$, this defines an order on match candidates, with highest values appearing first. One can either set a threshold θ to decide which similarity value is sufficient for a match or or limit the number of matches to the top k ranked ones. Using thresholding, all pairs $sim_{NG}(a, b) \geq \theta$ will then be output as a match, and using ranking, all pairs $\{(a_0, b_0), \ldots, (a_n, b_n)\}[1 : k]$ where $sim_{NG}(a_i, b_i) \geq sim_{NG}(a_j, b_j)$ for all $i < j$ will be output as a match. We choose the former and determine a separate threshold per dataset as follows: we consider all calculated similarity values as thresholds and pick the threshold that achieves the best F1-score on the dataset. We then choose sim_{NG} with this threshold as the baseline for the considered dataset. This

[2]https://pypi.org/project/textdistance

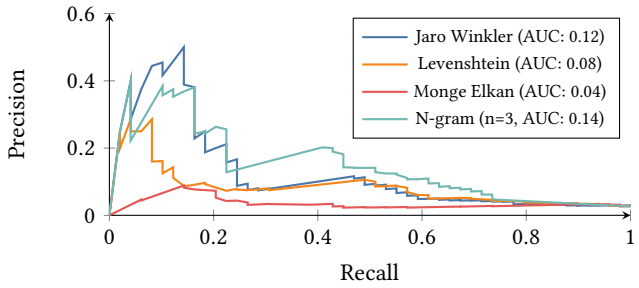

Figure 2: Precision-Recall curve of different string similarity metrics, tested on the attribute names of our benchmark.

approach favors the baseline, as it overestimates the capabilities of the N-gram string similarity for schema matching. In practice, a data engineer cannot know which threshold to use.

The choice of tresholding over ranking is motivated by the fact that the output of the LLM does not imply any ordering, we ask for a simple `yes`, `no` or `unknown` decision instead. Hence, common ranking metrics such as *recall@k* or *mean reciprocal rank* cannot be applied to the LLM results. Furthermore, we note that our approach to determine dataset-specific thresholds is equivalent to choosing a dataset-specific k that maximizes the F1-score when interpreting the sim_{NG} as a ranking.

Experimental Setup. For a fixed dataset and fixed task scope, an *experiment* consists of sending the correspoding prompt three times to the LLM. We extract three *votes* from the responses and use majority voting to minimize the effect of hallucinations. If an attribute pair is missing or there is a split decision, this pair is considered `unknown`. Each experiment is repeated five times. The results are compared against our benchmark by means of (*i*) the *F1*-score, the harmonic mean between *precision* and *recall* w.r.t. the ground-truth semantically valid matches, and (*ii*) a *decisiveness*-score, indicating the fraction of non-`unknown` votes. We use OpenAI's `gpt-3.5-turbo-0125` and `gpt-4-0125-preview` models with default settings, acknowledging the fact that performance could be improved with tuning the settings. Jupyter notebooks that we used to obtain the results can be found in our artefacts repository [15].

3 RESULTS

We next present the findings of our experimental study on schema matching using LLMs. Section 3.1 focuses on the quality of the schema matching results generated by the different separate task scopes, whereas Section 3.2 addresses their complementarity and the benefits of combining task scopes.

3.1 Quality of schema matching

We begin by evaluating the quality of schema matching results produced by the different task scopes, using F1-scores for comparison both among the LLMs and against the baseline (Section 3.1.1). Next, we assess the decisiveness of the LLMs in their opinions on attribute pairs in Section 3.1.2. Finally, we analyze the consistency of our experiments across various task scopes and datasets in Section 3.1.3, reporting the standard deviation of F1-score, precision, and recall to illustrate the expected variance when using LLMs for schema matching.

3.1.1 F1-scores. Table 2 shows the median F1-scores of each task scope per dataset, for both LLMs that we tested. The colouring indicates whether the F1-scores are higher (green) or lower (purple) than sim_{NG}. The best F1-score of each dataset is set in bold. We observe that the maximal F1-scores range from 0.364 to 0.800, highlighting a variation in the difficulty across the datasets. The bottom row displays the mean per column over all datasets and reveals the following general trends:

- all task scopes, except for `1-to-1`, outperform the baseline sim_{NG};
- each task scope shows an improvement in F1-score when moving from `GPT-3.5` to `GPT-4`; and,
- under the task scopes tested on both LLMs, `N-to-M` has the lowest mean F1-score.

Next, we conduct a more detailed analysis of each task scope in relation to the datasets.

We see that `1-to-1` is the least performing task scope: it fails to achieve the maximal F1-score on any dataset and is worse than (or on par with) the N-gram baseline, with the exception of the `DiCO` and the `LaMe` datasets. Moreover, `1-to-1` is typically outperformed by other scopes that incorporate more information in their prompts. Consequently, we assert that it lacks sufficient information for making informed, high-quality decisions.

Due to the low performance of `1-to-1` under `GPT-3.5`, combined with its high monetary cost, we decided to exclude the `1-to-1` task scope for our experiments using `GPT-4`.

For the analysis of the remainder of the task scopes, we use the following format. For a fixed task scope, we first consider `GPT-3.5`, and compare it with the baseline and the other task scopes run under `GPT-3.5`. We then make a comparison with its `GPT-4` counterpart. Finally, we consider the task scope run under `GPT-4` and compare with the baseline and the other task scopes for `GPT-4`.

The `1-to-N` task scope obtains the maximal `GPT-3.5` F1-score on the `DiCO`, `PrDE` and the `SeVD` dataset, outperforming sim_{NG} on five of nine datasets. With a single exception on the `PaPe` dataset, `1-to-N` dominates `1-to-1`. By this comparison, we deduce that adding more context information to a single prompt improves the quality of the LLM's decisions. The scores of `1-to-N` can be further improved by using `GPT-4`, the `SeVD` dataset being the single exception. This improvement can be attributed to an increase in precision on each dataset except on `DiCO` where it remains the same. With `GPT-4`, `1-to-N` outperforms sim_{NG} on eight datasets.

Using the `N-to-1` task scope we see improved F1-scores on average, achieving the maximal F1-score of the baseline and all `GPT-3.5`-based experiments four times and dominating sim_{NG} on five of nine datasets. Further, `N-to-1` dominates `1-to-1` on all datasets except for the `DiCO` dataset, reinforcing the deduction we made for `1-to-N`: adding context information improves matching quality. Using `GPT-4`, the `N-to-1` task scope dominates the N-gram baseline on every single dataset, achieving the highest F1-score on five datasets as well as the highest F1-score on average. Analogous to `1-to-N`, the improvement can be attributed to an improved precision score on every dataset while the recall decreases on one dataset.

Table 2: Median F1-scores, coloured for comparison against the N-gram similarity baseline. Green indicates an F1-score higher, purple an F1-score lower than the baseline. After each F1-score, in parenthesis (p, r), we give the precision p and the recall r. The best F1-score of each dataset is set in bold.

		GPT-3.5				GPT-4		
dataset	sim_{NG}	1-to-1	1-to-N	N-to-1	N-to-M	1-to-N	N-to-1	N-to-M
AdCO	0.286 (0.20, 0.50)	0.000 (0.00, 0.00)	0.133 (0.08, 0.50)	0.200 (0.11, 1.00)	0.286 (0.20, 0.50)	**0.400** (0.33, 0.50)	**0.400** (0.25, 1.00)	**0.400** (0.33, 0.50)
AdVD	0.125 (0.07, 0.40)	0.000 (0.00, 0.00)	0.083 (0.05, 0.20)	0.250 (0.16, 0.60)	0.286 (0.50, 0.20)	0.250 (0.18, 0.40)	0.316 (0.21, 0.60)	**0.364** (0.33, 0.40)
AdVO	0.333 (0.50, 0.25)	0.235 (0.22, 0.25)	0.320 (0.24, 0.50)	0.500 (0.38, 0.75)	0.182 (0.33, 0.12)	0.444 (0.40, 0.50)	**0.636** (0.50, 0.88)	0.308 (0.40, 0.25)
DiCO	0.400 (0.33, 0.50)	0.667 (1.00, 0.50)	**0.800** (0.67, 1.00)	0.267 (0.15, 1.00)	0.667 (0.50, 1.00)	**0.800** (0.67, 1.00)	0.667 (0.50, 1.00)	**0.800** (0.67, 1.00)
LaMe	0.333 (1.00, 0.20)	0.471 (0.57, 0.40)	0.500 (0.50, 0.50)	0.667 (0.53, 0.90)	0.500 (0.67, 0.40)	0.636 (0.58, 0.70)	**0.800** (0.67, 1.00)	0.556 (0.62, 0.50)
PaPe	0.600 (0.60, 0.60)	0.571 (1.00, 0.40)	0.500 (0.43, 0.60)	0.615 (0.50, 0.80)	0.333 (1.00, 0.20)	0.571 (1.00, 0.40)	**0.800** (0.80, 0.80)	0.571 (1.00, 0.40)
PrDE	0.333 (0.25, 0.50)	0.222 (0.33, 0.17)	0.417 (0.28, 0.83)	0.276 (0.17, 0.67)	0.200 (0.25, 0.17)	**0.556** (0.42, 0.83)	0.500 (0.36, 0.83)	0.333 (0.33, 0.33)
SeVD	0.222 (0.25, 0.20)	0.000 (0.00, 0.00)	0.400 (0.40, 0.40)	0.400 (0.30, 0.60)	0.333 (1.00, 0.20)	0.333 (1.00, 0.20)	**0.571** (1.00, 0.40)	0.286 (0.50, 0.20)
TrVD	0.381 (0.27, 0.67)	0.000 (0.00, 0.00)	0.429 (0.38, 0.50)	0.316 (0.23, 0.50)	0.600 (0.75, 0.50)	**0.667** (0.67, 0.67)	0.533 (0.44, 0.67)	0.600 (0.75, 0.50)
mean	0.335 (0.39, 0.42)	0.241 (0.35, 0.19)	0.398 (0.33, 0.56)	0.388 (0.28, 0.76)	0.376 (0.58, 0.37)	0.518 (0.58, 0.58)	**0.580** (0.53, 0.80)	0.469 (0.55, 0.45)

Finally, the N-to-M task scope achieves a maximal F1-score among the GPT-3.5 approaches three times, outperforming both sim_{NG} and 1-to-1 on five and six datasets, respectively. With the exception of SeVD, the use of GPT-4 improves the F1-scores on all datasets, resulting in the highest F1-score on three datasets. In contrast to 1-to-N and N-to-1, recall of N-to-M is better or on par with its GPT-3.5 counterpart. With GPT-4, N-to-M dominates sim_{NG} on six datasets. We hypothesize that the failure to improve the number of sim_{NG}-dominating datasets is due to the increase in complexity of the output format. While it is sufficient to simply list attribute names for 1-to-N and N-to-1, we need a list of attribute pairs for N-to-M.

Conclusion. We observe that for both LLMs, all task scopes, except for 1-to-1, outperform the baseline on average with a maximal increase of 0.245 points. However, no single task scope consistently dominates across all datasets. Across task scopes, moving from GPT-3.5 to GPT-4 increases the F1-score over all data sets (with SeVD as a single exception for 1-to-N as well as N-to-M) confirming the general accepted belief that transitioning to more advanced LLMs yields better results. Interestingly, when moving from GPT-3.5 to GPT-4 the rise in F1-score is due to an increase in precision for the task scopes 1-to-N and N-to-1, while for N-to-M it is due to an increase in recall (sometimes even at the expense of a slight drop in precision). Finally, within the same LLM, N-to-M is the least performing task scope of the three task scopes we analysed on both LLMs.

3.1.2 Decisiveness. In the course of our experiments, we observed that the LLM often fails to express an opinion on all attribute pairs requested. We summarize this behavior in the decisiveness score shown in Table 3. This score captures the ratio of attribute pairs that received a yes or no vote–so not an unknown–to all attribute pairs per dataset. As the name already indicates it measures how decisive the model is. On most datasets, the following inequality holds: 1-to-1 > N-to-1 > 1-to-N > N-to-M. We clearly see that increasing the amount of information per prompt decreases the decisiveness. With GPT-3.5, the N-to-1 task scope remains in an acceptable range, 1-to-N fluctuates between datasets while N-to-M is consistently in an unacceptable range. The use of GPT-4 improves the decisiveness considerably for N-to-1 and 1-to-N. Interestingly, the decisiveness of N-to-M does not profit from the larger model.

Table 3: Decisiveness scores (the number of attribute pairs that received a yes or no score–so not an unknown–relative to the total number of attribute pairs) per task scope and model.

	GPT-3.5				GPT-4		
dataset	1-to-1	1-to-N	N-to-1	N-to-M	1-to-N	N-to-1	N-to-M
AdCO	1.000	0.160	0.160	0.023	0.996	0.992	0.012
AdVD	0.993	0.128	0.164	0.007	0.947	1.000	0.023
AdVO	1.000	0.066	0.085	0.011	0.993	0.996	0.022
DiCO	0.988	0.312	0.362	0.050	1.000	1.000	0.037
LaMe	0.995	0.145	0.085	0.040	0.905	1.000	0.045
PaPe	1.000	0.065	0.315	0.009	0.981	1.000	0.046
PrDE	0.997	0.115	0.102	0.010	0.895	0.990	0.028
SeVD	0.989	0.053	0.421	0.011	0.989	0.947	0.021
TrVD	1.000	0.060	0.263	0.030	1.000	1.000	0.030
mean	0.996	0.123	0.218	0.021	0.967	0.992	0.029

Given the low quality of results for 1-to-1, the high decisiveness indicates that using the 1-to-1 task scope makes the wrong decision most of the time. This supports our decision to exclude 1-to-1 from further experiments with GPT-4. The extremely low decisiveness of N-to-M, however, may indicate that the complexity of the output could play a major role in the low quality of its results. As previously mentioned, the output of N-to-M is a list of tuples of attribute names while it is sufficient to simply list attribute names for 1-to-N and N-to-1.

Conclusion. An increase of context information per prompt decreases the number of attribute pairs an LLM expresses an opinion on. While this effect can be mitigated using GPT-4 for 1-to-N and N-to-1, this is not the case for N-to-M.

3.1.3 Consistency. We have been reporting results with respect to the median. Given that we conducted each experiment five times, it is interesting to investigate the consistency of the experiment results. We do so by reporting the standard deviation of F1-scores, precision and recall in Table 4. We see that, on average, 1-to-N and N-to-1 have low standard deviations with 0.074 and 0.062, respectively. Both 1-to-1 (0.141) and N-to-M (0.160) have higher standard deviations, N-to-M reaching the maximum across the whole table. Using GPT-4, the results increase in consistency. N-to-1 reaching

Table 4: The standard deviation of F1-score, precision and recall (the latter two are presented in brackets) calculated from five experiment runs. A darker green indicates a lower standard deviation for the F1-score.

scope	GPT-3.5	GPT-4
1-to-1	0.141 (0.23, 0.12)	*N/A*
1-to-N	0.074 (0.09, 0.10)	0.031 (0.07, 0.03)
N-to-1	0.062 (0.05, 0.12)	0.037 (0.05, 0.06)
N-to-M	0.160 (0.23, 0.18)	0.094 (0.10, 0.10)

the overall minimum with 0.031 followed by `1-to-N` with 0.037. `N-to-M` remains the least consistent but improves to 0.094.

Conclusion. We find that the standard deviation of the F1-scores remains in acceptable ranges (< 0.1) for `1-to-N` and `N-to-1` on both models. With `GPT-4`, all standard deviations improve further. We conclude that LLMs are consistent enough to be used in practice for schema matching.

3.2 Complementarity

It is rare to find matching methods that combine high recall with high precision. Since in practical data integration scenarios one needs to manually verify the match candidates that are proposed by an automated matching algorithm, its preferable from a practical viewpoint to use a matching algorithm that has very high recall (to ensure that no candidates are missed) while featuring a decent precision (to ensure that the verification effort remains manageable). From Table 2, we observe that using LLMs often enhances recall compared to the baseline, with this improvement being more pronounced for `GPT-4` than for `GPT-3.5`. Given this observation, we investigate in this section how *complementary* the different tasks scopes are with the baseline and each other. For, if the sets of matching candidates returned by distinct methods A and B are largely complementary (in the sense that there is little overlap between the returned sets), we could further increase recall by combining the methods A and B into a method $A\&B$: the combined method simply returns the union of the matches of A and B. We must take care, however, as while the recall of $A\&B$ may increase compared to A and B alone, its precision will almost certainly decrease. As such, we are also interested in quantifying whether the verification effort for $A\&B$ remains reasonable.

Our results in this section are computed using the following methodology. We refer to an element of {sim_{NG}, `1-1`, `1-N`, `N-1`, `N-M` } as a *method*. Remember from Section 2 that per method we have repeated each experiment five times. Consequently per pair (S_1, S_2) of distinct methods we have 25 experiment pairs (E_1, E_2). We take the union of the matches resulting from E_1 and E_2 and analyze this combined match w.r.t. the number of true positives, the recall, precision, etc. Per pair of methods (S_1, S_2) we may compute a dataset-specific average of these methods by summing the metric result over all 25 experiment pairs, and taking the average. Importantly, we only combine methods using the same LLM model (i.e. both use `GPT-3.5` or both use `GPT-4`). Concretely, in Section 3.2.1 we analyze complementarity of matches by investigating how many additional true positive semantic matches may be recovered when combining methods. In Section 3.2.2 we offset study by the verification effort required when combining methods. Finally, in Section 3.2.3, we analyze the F1-scores for every method combination.

Table 5: True semantic matches found when combining methods. A darker shade of green indicates a higher count. The diagonal shows the average count of true semantic matches found by only that method. Recall that the ground truth consists of 49 matches.

		GPT-3.5				GPT-4		
scope	sim_{NG}	1-to-1	1-to-N	N-to-1	N-to-M	1-to-N	N-to-1	N-to-M
sim_{NG}	19.0	24.2	32.4	37.8	24.2	31.6	38.4	28.6
1-to-1		9.8	29.1	36.6	20.4	*N/A*	*N/A*	*N/A*
1-to-N			27.0	40.0	29.2	28.2	39.4	29.6
N-to-1				35.4	37.2		38.4	38.5
N-to-M					14.4			21.8

3.2.1 Counts of true semantic matches. Table 5 shows the number of true semantic matches (i.e., true positives) found by combining methods. Concretely, for method combination (i, j) and each dataset we first compute the average number of true positives returned. We then sum these averages over all datasets, and report this sum in cell (i, j). The diagonals show the average count of true semantic matches found by the corresponding method alone, thus not combined with another method. This number serves as a reference for the method combinations: numbers for method combinations that are higher indicate an increase compared to using the method alone.

We first discuss the findings for `GPT-3.5` and then those for `GPT-4`. Remember that there is a total of 49 true semantic matches in the ground truth (cf. Table 1).

We observe that combining `1-to-N` and `N-to-1` yields the highest count of true semantic matches on average (40 out of 49). `N-to-1` by itself uncovers most true semantic matches on average (35.4 for `GPT-3.5`), which makes it the best task scope to combine with. We see that any combination with `N-to-1` yields more matches than any combination without `N-to-1`. As such, `N-to-1` is complementary with all other methods. This observation even holds when using the larger `GPT-4` model, even though the number of semantic matches found by `N-to-1` is higher on average than with `GPT-3.5`. The best combination of task scopes without `N-to-1` is sim_{NG} combined with `1-to-N`, yielding an average count of 32.4. We note that this value is worse than using `N-to-1` on its own. Overall, `1-to-N` is the second-best combination partner, as every average counts gets lower if we swap out `1-to-N` for any other task scope except `N-to-1`. The use of `GPT-4` does not notably improve the average counts of true semantic matches for combinations with `1-to-N` or `N-to-1`. With `N-to-M`, however, we do see an increase for all combinations.

Conclusion. Combining `1-to-N` and `N-to-1` yields the highest count of true semantic matches on average. The use of `GPT-4` does not improve this count by much.

3.2.2 Verification effort required. To assess the human effort required to verify candidate matches we report in Table 6 the size of the match candidates returned per combined method. Concretely,

Table 6: Verification effort: the count of matches that have to be inspected when combining two task scopes. The diagonals show the average matches found by a single task scope. A darker green indicates less effort. Recall that our total search space consists of 1839 attribute pairs and there are 49 matches in the ground truth.

		GPT-3.5				GPT-4		
scope	sim_{NG}	1-to-1	1-to-N	N-to-1	N-to-M	1-to-N	N-to-1	N-to-M
sim_{NG}	77.0	93.2	159.0	185.6	94.0	116.2	134.2	103.4
1-to-1		21.8	110.6	141.0	45.8	*N/A*	*N/A*	*N/A*
1-to-N			104.6	183.6	112.2	63.8	96.8	69.8
N-to-1				136.2	142.4		84.0	86.4
N-to-M					30.0			42.6

for method combination (i, j) and each dataset we first compute the average cardinality of the set of returned match candidates. We then sum these averages over all datasets, and report this sum in cell (i, j). The diagonals show the cardinality of the corresponding method alone, thus not combined with another method.

A larger cell value means that more candidates need to be inspected. Remember from Table 1 that the search space of possible matchings our benchmark consists of a total of 1839 attribute combinations and there are 49 true semantics matches in the ground truth. We include the verification effort for `1-to-1` for completeness, but do not discuss it in detail because of the low result quality.

Overall, we deem most LLM-counts acceptable for manual verification as they are much smaller (< 10%) than the entire search space of all attribute pairs. Looking specifically at methods used in isolation (shown on the diagonal) we observe that with `GPT-3.5`, `N-to-1` retrieves the most matches (136.2), followed by `1-to-N` (104.6) and `N-to-M` (30.0). Using `GPT-4`, the counts of `1-to-N` and `N-to-1` are further reduced while the count of `N-to-M` increases.

When we combine two task scopes, we observe that `1-to-N` combined with `N-to-1` retrieves the highest number of match candidates, 183.6 to be precise. We deem this acceptable for practical applications as it represents roughly only 10% of our entire search space. We note that while 183.6 candidates to inspect may still seem a lot, these numbers are aggregated over all datasets in our benchmark. When drilling down to the dataset level there are on average fewer than 20 candidates to verify using `GPT-3.5` (often much less), which reduces to fewer than 10 candidates to verify using `GPT-4`. Compared to the number of possible pairs per dataset shown in Table 1 this remains very modest.

Conclusion. Our experiments show that the number of retrieved matches is very reasonable (< 10% of the search space) and can be reduced further with the use of `GPT-4`, rendering the verification effort for all task scope combinations acceptable.

3.2.3 F1-scores. Table 7 presents F1-scores, precision and recall for all method combinations, averaged over all datasets. The diagonals represent the average scores for a single task scope and can be roughly compared to the last row of Table 2, where we report the average of the median scores. The table is meant to be read row-wise: the cell in row i and column j shows how combining method i with method j behaves compared to using method i alone.

First, let us discuss combining sim_{NG} with the different task scopes (first row of Table 7). We observe that any combination of sim_{NG} with any task scope increases the F1-score compared to using sim_{NG} alone own. Using `GPT-3.5`, `1-to-1` achieves the highest improvement, yielding an F1-score of 0.384 on average; `N-to-1` has the lowest improvement with 0.344. We see that the low F1-score of `N-to-1` can be attributed to a low precision, as recall is highest across all combinations including sim_{NG}. Using `GPT-4` increases the F1-scores further, yielding a maximum F1-score of 0.456 in combination with `N-to-1`. These numbers, hence, show how LLMs strictly improve over string-similarity-based matching.

Overall, we see that combining two task scopes improves the F1-score. Using `GPT-3.5`, we see three exceptions: (sim_{NG}, `N-to-1`), where using `N-to-1` on its own yields a higher F1-score; (`1-to-N`, `N-to-1`), where the F1-score is lower than using any task scope on its own; and (`1-to-N`, `N-to-M`), where using `1-to-N` on its own yields a higher F1-score. Further, combining `1-to-N` or `N-to-1` with sim_{NG} also reduces the F1-score while it improves for `1-to-1` and `N-to-M`. We also observe that the combination (`1-to-N`, `N-to-1`) achieves the highest recall of all combinations, even including the ones using `GPT-4`. Using the larger model, we see a similar trend as with `GPT-3.5` in that a task scope combination typically improves the average F1-score while combining with sim_{NG} worsens it. Further, combining `N-to-1` with any other task scope reduces its F1-score as well, making it the highest performing task scope based on the average F1-score. Looking at the `GPT-4` experiments in isolation, we again observe that the combination (`1-to-N`, `N-to-1`) achieves the best recall on average.

We note that while the F1-scores are generally not very high, we see that the task scopes achieve very different scores for precision and recall. Any combination with `N-to-1` generally improves recall at the cost of precision. For `1-to-N` the gap between precision and recall is similar but less pronounced. In contrast, both `1-to-1` and `N-to-M` do not contribute much to recall while keeping precision level.

Conclusion. Combining task scopes generally improves the F1-score on most combinations, with `N-to-1` using `GPT-4` achieving the highest F1-score.

4 CONCLUSION

In this study, we took an initial step towards utilizing LLMs for schema matching. We found that matching quality diminishes when there is insufficient context information (i.e., task scope `1-to-1`) and when there is an excess of context information (i.e., task scope `N-to-M`). The latter is likely hindered by the more complex output format and the larger number of pairs requiring decisions. The `1-to-N` and `N-to-1` task scopes effectively provide sufficient context to make accurate matches without overwhelming the decision-making process. This balance results in a better overall performance of which the recall can be even further enhanced by adopting a combined approach using both task scopes in tandem. This combined method successfully identifies a significant number of true semantic matches with an acceptable verification effort. As such, we recommend using the combined (`1-to-N`, `N-to-1`) method in practice. We also found that using GPT-4 over GPT-3.5 improves matching quality and consistency over all task scopes tested on

Table 7: F1-scores, precision and recall for combined methods, averaged over all datasets. The diagonals represent the average scores for a single task scope and provide the reference point for row-wise comparisons. A green colouring indicates a higher F1-score compared using the method mentioned in the row on its own, purple indicates a lower F1-score. Note that the precision and recall scores are also averages and thus do not directly correspond to the F1-scores shown.

		GPT-3.5				GPT-4		
scope	sim_{NG}	1-to-1	1-to-N	N-to-1	N-to-M	1-to-N	N-to-1	N-to-M
sim_{NG}	0.335 (0.39, 0.42)	0.384 (0.34, 0.53)	0.359 (0.27, 0.69)	0.344 (0.24, 0.78)	0.375 (0.35, 0.53)	0.406 (0.32, 0.63)	0.456 (0.35, 0.78)	0.409 (0.34, 0.59)
1-to-1		0.234 (0.35, 0.19)	0.417 (0.35, 0.62)	0.378 (0.27, 0.76)	0.425 (0.49, 0.44)	*N/A*	*N/A*	*N/A*
1-to-N			0.406 (0.35, 0.59)	0.351 (0.23, 0.83)	0.400 (0.33, 0.64)	0.505 (0.54, 0.58)	0.547 (0.46, 0.80)	0.513 (0.50, 0.61)
N-to-1				0.373 (0.27, 0.73)	0.388 (0.27, 0.78)		0.572 (0.50, 0.78)	0.562 (0.47, 0.78)
N-to-M					0.355 (0.49, 0.35)			0.486 (0.55, 0.48)

both models, and (except for `N-to-M`) increases decisiveness and reduces the verification effort. The results in this paper demonstrate that LLMs have the potential to bootstrap the schema matching process and assist data engineers in speeding up this task solely based on schema element names and descriptions, without the need for data instances and improving over attribute-name-based matching alone.

We outline some directions for future work that seem promising.

A benefit of LLMs over the string similarity baseline is that they can be instructed to provide an explanation as to why they identify a certain attribute pair as a match or a non-match. We believe that such explanations can be a valuable instrument for a data engineer tasked to construct a schema mapping, to identify and rectify misclassifications. Through initial experiments, we have observed that the LLM sometimes jumps to conclusions as it overemphasizes similarity of attribute names while disregarding the intent of the attributes as described in the provided documentation. For instance, we noticed that the LLM is eager to match two attributes solely based on the fact that they both refer to the time dimension of an event even when those events are clearly different. We are currently working on a tool that facilitates refining schema matchings via natural language feedback in a pragmatic and user-friendly way.

Our benchmark consists of publicly available schemas. In future experiments, we will apply our approach on proprietary schemas, aiming to illustrate the usefulness of using LLMs for schema matching in real-world scenarios.

ACKNOWLEDGMENTS

S. Vansummeren was supported by the Bijzonder Onderzoeksfonds (BOF) of Hasselt University under Grant No. BOF20ZAP02. This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" programme. This work was supported by Research Foundation—Flanders (FWO) for ELIXIR Belgium (I002819N). The resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

REFERENCES

[1] 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 (2023), 120405. https://doi.org/10.1016/J.ESWA.2023.120405

[2] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. 2011. Generic Schema Matching, Ten Years Later. *Proc. VLDB Endow.* 4, 11 (2011), 695–701. http://www.vldb.org/pvldb/vol4/p695-bernstein_madhavan_rahm.pdf

[3] Rishi Bommasani et al. 2022. On the Opportunities and Risks of Foundation Models. arXiv:2108.07258 [cs] https://arxiv.org/abs/2108.07258

[4] Michelle Cheatham and Pascal Hitzler. 2013. String Similarity Metrics for Ontology Alignment. In *Advanced Information Systems Engineering*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Camille Salinesi, Moira C. Norrie, and Óscar Pastor (Eds.). Vol. 7908. Springer Berlin Heidelberg, Berlin, Heidelberg, 294–309. https://doi.org/10.1007/978-3-642-41338-4_19

[5] Chen Chen, Behzad Golshan, Alon Y. Halevy, Wang-Chiew Tan, and AnHai Doan. 2018. BigGorilla: An Open-Source Ecosystem for Data Preparation and Integration. *IEEE Data Eng. Bull.* 41, 2 (2018), 10–22. http://sites.computer.org/debull/A18june/p10.pdf

[6] AnHai Doan, Alon Halevy, and Zachary G. Ives. 2012. *Principles of Data Integration.* Morgan Kaufmann, Waltham, MA. http://research.cs.wisc.edu/dibook/

[7] Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Sicheng Hao, Benjamin Moody, Brian Gow, Liwei H. Lehman, Leo A. Celi, and Roger G. Mark. 2023. MIMIC-IV, a Freely Accessible Electronic Health Record Dataset. *Scientific Data* 10, 1 (Jan. 2023), 1. https://doi.org/10.1038/s41597-022-01899-x

[8] Michael Kallfelz, Anna Tsvetkova, Tom Pollard, Manlik Kwong, Gigi Lipori, Vojtech Huser, Jeffrey Osborn, Sicheng Hao, and Andrew Williams. 2021. MIMIC-IV Demo Data in the OMOP Common Data Model. https://doi.org/10.13026/P1F5-7X35

[9] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large Language Models are Zero-Shot Reasoners. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (Eds.). http://papers.nips.cc/paper_files/paper/2022/hash/8bb0d291acd4acf06ef112099c16f326-Abstract-Conference.html

[10] Christos Koutras, Kyriakos Psarakis, George Siachamis, Andra Ionescu, Marios Fragkoulis, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine in Action: Matching Tabular Data at Scale. *Proc. VLDB Endow.* 14, 12 (2021), 2871–2874. https://doi.org/10.14778/3476311.3476366

[11] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *37th IEEE International Conference on Data Engineering, ICDE 2021, Chania, Greece, April 19-22, 2021*. IEEE, 468–479. https://doi.org/10.1109/ICDE51399.2021.00047

[12] Debayan Mukherjee, Atreya Bandyopadhyay, Rajdip Chowdhury, and Indrajit Bhattacharya. 2021. Learning Knowledge Graph for Target-driven Schema Matching. In *8th ACM IKDD CODS & 26th COMAD*. ACM, 65–73. https://doi.org/10.1145/3430984.3431013

[13] Avanika Narayan, Ines Chami, Laurel J. Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? *Proc. VLDB Endow.* 16, 4 (2022), 738–746. https://doi.org/10.14778/3574245.3574258

[14] Observational Health Data Sciences and Informatics. 2019. *The Book of OHDSI: Observational Health Data Sciences and Informatics.* OHDSI, San Bernardino, CA. http://book.ohdsi.org/

[15] Marcel Parciak, Brecht Vandevoort, Frank Neven, Liesbet M. Peeters, and Stijn Vansummeren. 2024. Artifact Repository to Schema Matching with Large Language Models: An Experimental Study. https://github.com/UHasselt-DSI-Data-Systems-Lab/code-schema-matching-LLMs-artefacs

[16] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. https://doi.org/10.3115/v1/D14-1162

[17] Erhard Rahm and Philip A. Bernstein. 2001. A Survey of Approaches to Automatic Schema Matching. *Vldb Journal* 10, 4 (2001), 334–350. https://doi.org/10.1007/S007780100057

[18] Roee Shraga, Avigdor Gal, and Haggai Roitman. 2020. ADnEV: Cross-Domain Schema Matching using Deep Similarity Matrix Adjustment and Evaluation. *Proc. VLDB Endow.* 13, 9 (2020), 1401–1415. https://doi.org/10.14778/3397230.3397237

[19] Yufei Sun, Liangli Ma, and Shuang Wang. 2015. A Comparative Evaluation of String Similarity Metrics for Ontology Alignment. *Journal of Information and Computational Science* 12, 3 (Feb. 2015), 957–964. https://doi.org/10.12733/jics20105420

[20] Jules White, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. 2023. A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382 [cs]

[21] 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. *Advances in databases and information systems. ADBIS* 12843 (Aug. 2021), 260–274. https://doi.org/10.1007/978-3-030-82472-3_19

[22] 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. In *39th IEEE International Conference on Data Engineering, ICDE 2023, Anaheim, CA, USA, April 3-7, 2023*. 1558–1571. https://doi.org/10.1109/ICDE55515.2023.00123