LLM-Matcher: A Name-Based Schema Matching Tool using Large Language Models

Peer-reviewed author version

PARCIAK, Marcel; VANDEVOORT, Brecht; NEVEN, Frank; PEETERS, Liesbet & VANSUMMEREN, Stijn (2025) LLM-Matcher: A Name-Based Schema Matching Tool using Large Language Models. In: Companion of the 2025 international conference on management of data Sigmod-Companion 2025, ASSOC Computing machinery, p. 203 -206.

DOI: 10.1145/3722212.3725112
Handle: http://hdl.handle.net/1942/47398

LLM-Matcher: a Name-Based Schema Matching Tool using Large Language Models

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

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

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

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

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

Abstract

We present LLM-Matcher, an interactive name-based schema matching system that utilizes large language models (LLMs) to identify correspondences between source and target schema elements relying solely on their names and descriptions. This tool is specifically designed for restricted environments where instance-based schema matching is not possible, such as the healthcare domain where instance access is often prohibited. LLM-Matcher is based on an extensive experimental study, showing the capabilities of LLMs in the schema matching task. Our system is specifically tailored towards users with sufficient domain knowledge and offers an interpretable initial mapping that can be further refined by providing textual feedback. This feedback allows to rectify model misconceptions as well as improve the quality of schema element descriptions. This paper provides a comprehensive overview of LLM-Matcher, explores its application in the healthcare domain, allows users to gain additional insight into our experimental study, and outlines different steps showcased in the demonstration.

CCS Concepts

• **Information systems** → **Information integration**.

Keywords

Schema matching, large language models, health data integration

ACM Reference Format:
Marcel Parciak, Brecht Vandevoort, Frank Neven, Liesbet M. Peeters, and Stijn Vansummeren. 2025. LLM-Matcher: a Name-Based Schema Matching Tool using Large Language Models. In *Companion of the 2025 International Conference on Management of Data (SIGMOD-Companion '25), June 22–27, 2025, Berlin, Germany*. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3722212.3725112

SIGMOD-Companion '25, Berlin, Germany

ACM ISBN 979-8-4007-1564-8/2025/06
https://doi.org/10.1145/3722212.3725112

1 Introduction

Schema matching [11] is a core task in data integration [4], where it refers to the problem of identifying correspondences between the elements of two schemas so that corresponding elements represent the same real-world concept. For example, a schema matcher may determine that the attribute `patient_id` in one table is semantically equivalent to the attribute `person_id` in another. Constructed matchings can be used to translate data conforming to the first schema into data conforming to the second schema, a process known as *schema mapping*. Schema matching is hence a necessary first step in integrating data from a source database into a target database and provides the required starting point for schema mapping systems like Clio [5]. We focus on schema matching, with the healthcare domain as a key application area.

Because manual schema matching is a time-consuming, error-prone and tedious process, significant research effort has been devoted to automating schema matching [2, 3, 11]. In general, however, it is not possible to fully automate schema matching, primarily because schemas in practice have specific semantics that are not formally expressed, but remain implicit in domain knowledge. Schema matching software, therefore, should only determine *match candidates*, which the user can accept, reject, or change [11].

To generate match candidates, schema matching software can exploit a wide variety of signals that hint at element correspondence: exploit syntactic similarity between attribute names; consult thesauri; go beyond the schema level and also look at actual data values and value distributions in concrete database instances; exploit database constraints; or consider past mappings [1, 3, 11]. Unfortunately, many such signals are often unavailable in real-world schemas [7]: attribute names are often cryptic and involve domain-specific abbreviations, and the use of actual data values and concrete database instances may be restricted due to legal reasons; this is the case in particular in the healthcare domain where, even within healthcare organisations, regulations such as the European General Data Protection Regulation restrict access to data instances. Hence, data engineers in the healthcare domain need approaches for instance-free, name-based schema matching.

Schema matchers like DITTO [6], LSM [16] or SMAT [15] use BERT-based models for instance-free schema matching. Such approaches necessitate that a significant part of the data, i.e. the potential matches between two schemas, are labelled beforehand

for fine-tuning. In real-world settings, true (non-)matches are typically unknown and thus unavailable. In response, we developed and extensively evaluated an approach based on *large language models* (LLMs) in [10]. LLMs are machine learning models trained on generic, web-scale textual data. They have shown to solve data wrangling tasks [8], but have not been widely used for schema matching to this date. ReMatch [12] uses a similar approach, but did not make the prompts used to perform the matching publicly available, making ReMatch unavailable to health data engineers.

In this demonstration, we present LLM-Matcher, an open-source schema matcher that utilizes an off-the-shelf large language model to perform instance-free, name-based schema matching. Key features of LLM-Matcher are the following: (*i*) Ease of deployment: LLM-Matcher stands out for its minimal deployment requirements. By operating only on schema metadata, privacy concerns are circumvented which allows to utilize any cloud-based LLM backend as opposed to needing to run the LLM locally which typically demands substantial computational resources. (*ii*) Effectiveness: The LLM interaction methodology is based on a comprehensive experimental study [10] and outperforms matching methods based on string edit distance. Given the constraints of the healthcare domain elaborated above, edit-distance based matching remains the state-of-the-art method in this context. During the demonstration, attendees will be able to assess the schema matching and refinement capabilities of LLM-Matcher and compare them to methods based on string edit distance.

This manuscript is further organized as follows. We first describe an example schema matching problem from the healthcare domain that we will use as a running example in Section 2. We next describe the components of LLM-Matcher in Section 3, and discuss concrete demonstration scenarios in Section 4.

2 Health domain example

We introduce the following running example from [10] as a practical use case to illustrate the capabilities of LLM-Matcher and to serve as a guiding scenario for the demonstration. For ease of presentation, we discuss only a handful of relations. During the demonstration, attendees will be able to explore all relations discussed in [10] or to define additional schemas themselves.

A health data engineer is tasked to transform data from a schema *Source* to a schema *Target*. *Source* describes a hospital information system, where the engineer picks the relation *Admissions* for matching. *Admissions* contains details about the patient admission, such as an admission datetime or the type of admission. In our running example, *Admissions* is taken from the MIMIC-IV data set.[1] The attributes of the considered relation are depicted in Table 1 (left); we refer to the MIMIC-IV online documentation[1] for more information. *Target* is the OMOP common data model, which harmonizes a broad range of different disease-specific information sources. We focus on the *Visit Occurrence* relation. *Visit Occurrence* contains information about events where persons engage with the healthcare system for a duration of time. An overview of all attributes of *Target* is given in Table 1 (right). Further details can be found in the online OMOP common data model documentation.[2]

[1]https://mimic.mit.edu/docs/iv/modules/hosp/
[2]https://ohdsi.github.io/CommonDataModel/cdm53.html

***Source*.Admissions**
hadm_id
subject_id
admittime
dischtime
deathtime
admission_type
admit_provider_id
admission_location
discharge_location
insurance
language
marital_status
ethnicity
edregtime
edouttime
hospital_expire_flag

***Target*.Visit Occurrence**
visit_occurrence_id
person_id
visit_concept_id
visit_start_date
visit_start_datetime
visit_end_date
visit_end_datetime
visit_type_concept_id
provider_id
care_site_id
visit_source_value
visit_source_concept_id
admitting_source_concept_id
admitting_source_value
discharge_to_concept_id
discharge_to_source_value
preceding_visit_occurrence_id

Table 1: Schema example: Overview of the relations and attributes in *Source* and *Target*.

In [9] researchers from the healthcare domain consented on facilitating Extract, Transform and Load (ETL) process development solely through sharing descriptions and summaries of data with data engineers. This approach, while challenging due to the absence of actual data instances for *Source* and *Target*, ensures that sensitive health data is processed in compliance with GDPR regulations. Hence, the health data engineer can only rely on their knowledge of the health domain and the textual descriptions of the schema elements provided by the documentation. The data engineer chooses a relation S in *Source* and a relation T in *Target* and as a first step wants to identify pairs of matching attributes. We say that a pair of attributes (a, a') *matches* when a is an attribute of S, a' is an attribute of T and there is a one-to-one correspondence between the values of a and the values of a'. We focus on one-to-one matches analogous to our experimental evaluation presented in [10]. To illustrate the inherent difficulty of this schema matching task we point out that out of the 272 possible attribute pairs only 8 pairs represent a valid match. A ground truth was obtained through inspection of a publicly available ETL pipeline transforming MIMIC data to the OMOP common data model. In [10], we considered a total of nine different combinations of S and T, ranging from 80 to 391 possible attribute pairs with 2 to 10 true matches.

3 Overview of LLM-Matcher

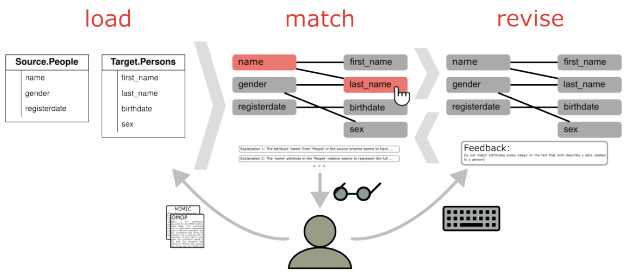

Figure 1: Overview of the user interactions in LLM-Matcher.

Figure 1 presents a graphical overview of the three steps LLM-Matcher consists of: (1) loading the description of the source and target relations; (2) computing matching attribute pairs with

associated natural language justification; and, (3) interpreting the mapping and providing natural language feedback. Steps (2) and (3) are repeated until a satisfactory mapping is achieved. We next elaborate on each of these three functionalities.

3.1 Load

To initialize the matching process, the user manually inputs a specification of a relation S in *Source* and a relation T in *Target* or loads a JSON-formatted specification. These contain, for each attribute, the attribute's name and a natural language description defining the intended semantics. After loading the specifications, the user is able to in- and/or exclude individual attributes. A common use case would be the exclusion of auto-generated identifiers in the target schema. Excluded attributes will not be shown to the LLM.

3.2 Match

When the schemas are loaded, LLM-Matcher performs an initial matching by prompting an LLM. By default we use `GPT-4o-mini`, although in principle any LLM can be used. The prompting is specified via prompt templates that describe the matching task itself, contain placeholders for the descriptions of the attributes and list the requirements for the output (for instance, that a decision is required for each attribute pair and the format of the output). The matching quality greatly depends on the chosen prompt in two ways. First, we choose prompt patterns that are known to work well to build our prompt [14]. Second, the right amount of context information is important. We defined and compared different *task scopes* in our experimental study [10]. Task scopes describe the amount of information from the source and the target schema supplied to a single prompt: *1-to-1*, *1-to-N*, *N-to-1* and *N-to-M*. A 1-to-N task scope, for example, asks for the matching between a fixed attribute in S and all attributes in T. That is, only the descriptions of one attribute a in S and the descriptions of all attributes in T are specified in the prompt by substituting the corresponding placeholders. By default, we use a combination of the 1-to-N and N-to-1 task scopes as these two have shown to provide the best matching results. Further information, rationale and evaluation with respect to task scopes can be found in our experimental study [10].

LLM-Matcher renders the task scope-specific prompt templates, initiates the requests to the LLM, and parses and processes the answers. Our prompts instruct the LLM to associate one of the labels *Yes*, *No*, or *Unknown* to every attribute pair (a, a'), indicating that there is a match, is no match, or that it can not determine whether there is a match. We refer to this output for a specific pair as a *vote*. When no decision is made by the LLM for an attribute pair, the label *Unknown* is assigned in post-processing. We obtain three votes per task scope to account for hallucinations. By default, we thus obtain six votes in total for every attribute pair in S and T. Next, we describe how LLM-Matcher visually presents the overall result and all individual votes retrieved from the LLM to the user.

3.3 Revise

LLM-Matcher visualizes the LLMs votes as a bipartite graph. The attributes of S and T are visualized as nodes. Per attribute pair, we represent the LLM's votes by drawing edges coloured by the vote label. The edge thickness indicates the count of votes per label. Consequently, each attribute pair can have up to three differently coloured edges representing the number of *Yes*, *No* and *Unknown* votes. Users can select attribute pairs to inspect the LLM's justification of each individual vote. Furthermore, LLM-Matcher presents potential matches identified by measuring the string similarity of the names of the attributes of S and T as a baseline. We default to the Dice-similarity of 3-grams as a baseline [13]. If a ground truth is provided beforehand, LLM-Matcher also shows metrics to assess the quality of the matches given by the LLM and the baseline.

A health data engineer can inspect the LLM's justifications to detect errors made by the model and provide feedback that is taken into account in the next matching iteration. Concrete examples of what can be learned are the following:

(1) False positives: 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.
(2) False negatives: for specific attributes the LLM simply does not have enough information as the documentation describing these attributes is imprecise, contains inaccuracies or is even missing.

After this inspection, natural language feedback can be provided in a text field that is added to the prompt and is taken into account when a rematching is initiated. Example feedback for case (1) above could instruct the model to attribute more importance towards similarity of the description of text fields for attributes related to time. For cases like (2), we allow users to directly modify the text describing the intent of the attributes.

4 Demonstration

With demonstrating LLM-Matcher, we aim to provide attendees with first-hand experience on the viability of large language models for instance-free, name-based schema matching. We define three scenarios to illustrate this in the following. We also supply a video recording[3] of a walkthrough of the first scenario.

4.1 First scenario: a simple introduction to LLM-Matcher.

As attendees might not be familiar with schema matching systems, we start out the demonstration with a more prototypical example to get acquainted with the tool. In this scenario, the task is to match two relations containing person information. That is, S contains the attributes *Name*, *Gender* and *Registerdate*, while T contains the attributes *First Name*, *Last Name*, *Birthdate* and *Sex*. The attendee can load these relations and has the ability to inspect the names of the attributes as well as the text fragments describing their intent as provided by the documentation. If desired, the attendee can alter these descriptions inside the tool.

The attendee can inspect LLM-Matcher's first matching result to gauge whether they are sound and whether the given natural language explanations for the matchings meet the expectations. Next

[3] https://drive.google.com/file/d/1LkyKm1ZYwJGx_Slr-fdojQBnNq0WRgLu/view

to this first matching, we present string similarity-based matches to allow attendees to compare two instance-free, name-based schema matching approaches directly using a simple example. Further, the attendee can supply specific textual feedback to trigger a regeneration of LLM-Matcher's results. The effect of the feedback can be, again, compared to the string similarity baseline to gauge the feedback's effect on the matching result.

4.2 Second scenario: diving deeper into the experimental study.

Now that the attendee is sufficiently familiar with the operations of LLM-Matcher, we turn to illustrating the effect that the task scope choice has on matching quality. The attendee will be able to freely choose a relation pair (S, T) from all real-world datasets used in our experimental study [10]; choose an LLM that will be prompted for matching; choose any number of task scopes from 1-to-N, N-to-1, or N-to-M; and choose a string similarity metric as a baseline for comparison. LLM-Matcher matches the attributes of S and T given the parameters chosen by the attendee.

The results from both LLM-Matcher and the chosen string similarity metric are presented to the attendee in direct comparison, as in the first scenario. In addition, we present the assessment metrics reported in our experimental study: F1-score, precision, recall and decisiveness (see [10] for more detail). The attendee will now be able to inspect first-hand the effect of task scopes on schema matching. First, all matches that contributed to the result of our experimental study are presented. Second, LLM-Matcher presents the natural language text explanations to provide insight into the LLM's reasoning. By varying the parameters of an experiment, the attendee will be able to conduct experiments not reported in our study, such as choosing alternative string similarity baselines. In summary, through this scenario we give novel and deeper insights into the experimental study performed in [10].

4.3 Third scenario: exploring LLM-Matcher with custom attributes.

Taking the role of a health data engineer, the attendee's task is to match attributes from S and T using the same process as already applied in the first scenario. In this scenario, the attendee will be asked to define a small number of attributes for S, or to choose a MIMIC example relation from our real-world dataset and modify a few attributes. As we expect that most attendees are not sufficiently familiar with the healthcare domain, we will assist with the creation or modification of attributes in S. However, we aim to motivate attendees to provide new, unseen examples for our LLM-based schema matching. In particular, we want to encourage attendees to provide attribute names and descriptions based on their experience, which may differ from the style we are used to in the healthcare domain. Next, we choose a relation T from the OMOP common data model (see Section 2) and define (or modify) a ground truth to faciliate the inspection of the results. S and T are matched using our LLM-based approach as well as by measuring the string similarity of attributes' names, providing, yet again, a baseline to compare our LLM-based approach against.

The attendee will be able to inspect the results similar to the previous two scenarios. The matches of LLM-Matcher are presented next to the baseline obtained by a string similarity metric, allowing attendees to assess the quality of LLM-based schema matching. In this scenario, however, we step away from the real-world schemas reported in [10]. The aim of this scenario is to highlight the viability of LLM-based schema matching with unseen schemas.

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.

[2] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm (Eds.). 2011. *Schema Matching and Mapping.* Springer.

[3] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. 2011. Generic Schema Matching, Ten Years Later. *Proc. VLDB Endow.* 4, 11 (2011), 695–701.

[4] AnHai Doan, Alon Halevy, and Zachary G. Ives. 2012. *Principles of Data Integration.* Morgan Kaufmann, Waltham, MA.

[5] Ronald Fagin, Laura M. Haas, Mauricio A. Hernández, Renée J. Miller, Lucian Popa, and Yannis Velegrakis. 2009. Clio: Schema Mapping Creation and Data Exchange. In *Conceptual Modeling: Foundations and Applications - Essays in Honor of John Mylopoulos (Lecture Notes in Computer Science, Vol. 5600),* Alexander Borgida, Vinay K. Chaudhri, Paolo Giorgini, and Eric S. K. Yu (Eds.). Springer, 198–236. doi:10.1007/978-3-642-02463-4_12

[6] Yuliang Li, Jinfeng Li, Yoshihiko Suhara, AnHai Doan, and Wang-Chiew Tan. 2020. Deep Entity Matching with Pre-Trained Language Models. *Proc. VLDB Endow.* 14, 1 (2020), 50–60. doi:10.14778/3421424.3421431

[7] 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.

[8] Avanika Narayan, Ines Chami, Laurel Orr, and Christopher Ré. 2022. Can Foundation Models Wrangle Your Data? arXiv:2205.09911 [cs]

[9] Observational Health Data Sciences and Informatics. 2019. *The Book of OHDSI.* OHDSI, San Bernardino, CA.

[10] Marcel Parciak, Brecht Vandevoort, Frank Neven, Liesbet M. Peeters, and Stijn Vansummeren. 2024. Schema Matching with Large Language Models: an Experimental Study. In *Proceedings of Workshops at the 50th International Conference on Very Large Data Bases, VLDB 2024, Guangzhou, China, August 26-30, 2024.* VLDB.org. https://vldb.org/workshops/2024/proceedings/TaDA/TaDA.8.pdf

[11] Erhard Rahm and Philip A. Bernstein. 2001. A survey of approaches to automatic schema matching. *VLDB J.* 10, 4 (2001), 334–350.

[12] Eitam Sheetrit, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. ReMatch: Retrieval Enhanced Schema Matching with LLMs. *CoRR* abs/2403.01567 (2024). doi:10.48550/ARXIV.2403.01567 arXiv:2403.01567

[13] Yufei Sun, Liangli Ma, and Shuang Wang. 2015. A comparative evaluation of string similarity metrics for ontology alignment. *Journal of Information &Computational Science* 12, 3 (2015), 957–964.

[14] 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. *CoRR* abs/2302.11382 (2023). doi:10.48550/ARXIV.2302.11382 arXiv:2302.11382

[15] Jing Zhang, Bonggun Shin, Jinho D. Choi, and Joyce C. Ho. 2021. SMAT: An Attention-Based Deep Learning Solution to the Automation of Schema Matching. In *Advances in Databases and Information Systems - 25th European Conference, ADBIS 2021, Tartu, Estonia, August 24-26, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12843),* Ladjel Bellatreche, Marlon Dumas, Panagiotis Karras, and Raimundas Matulevicius (Eds.). Springer, 260–274. doi:10.1007/978-3-030-82472-3_19

[16] 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 *ICDE.* 1558–1571.