

# SEALM: Semantically Enriched Attributes with Language Models for Linkage Recommendation

Leonard Traeger<sup>1,2</sup><sup>a</sup>, Andreas Behrend<sup>2</sup> and George Karabatis<sup>1</sup><sup>b</sup>

<sup>1</sup>Department of Information Systems, University of Maryland, Baltimore County, U.S.A.

<sup>2</sup>Institute of Computer and Communication Technology (ICCT), Technical University of Cologne, Germany  
{leonard.traeger, georgek}@umbc.edu, andreas.behrend@th-koeln.de


**Keywords:** Database Integration, Schema Matching, Language Models, Semantic Enrichment.


**Abstract:** Matching attributes from different repositories is an important step in the process of schema integration to consolidate heterogeneous data silos. In order to recommend linkages between relevant attributes, a contextually rich representation of each attribute is quite essential, particularly when more than two database schemas are to be integrated. This paper introduces the SEALM approach to generate a data catalog of semantically rich attribute descriptions using Generative Language Models based on a new technique that employs six variations of available metadata information. Instead of using raw attribute metadata, we generate SEALM descriptions, which are used to recommend linkages with an unsupervised matching pipeline that involves a novel multi-source Blocking algorithm. Experiments on multiple schemas yield a 5% to 20% recall improvement in recommending linkages with SEALM-based attribute descriptions generated by the tiniest Llama3.1:8B model compared to existing techniques. With SEALM, we only need to process the small fraction of attributes to be integrated rather than exhaustively inspecting all combinations of potential linkages.

## 1 INTRODUCTION

Schema Matching is a core discipline in data management, especially when dealing with integration tasks. Matching multiple and heterogeneous relational database schemas requires finding the semantic linkages between the different tables and attributes in order to query the respective records in an integrated view, which is an important pre-processing step for multi-source data search, query transformations, and data fusion (Bleiholder and Naumann, 2009). Automatically identifying true linkages in the large search space of candidates is a challenging task, particularly with more than two schemas (Saeedi et al., 2021). Consequently, the more schemas need to be integrated (volume), the more critical the context and representation of the tables and attributes becomes (veracity). **Motivating Example.** If two attributes represent the same semantic concept, they should be linked together since their signatures (numerical embeddings representing the attributes) are similar. Therefore, when computing the similarities among all potential pairs of attribute signatures, the linkages with the highest similarity score (lowest distance) are consid-

ered to be the true ones. We provide an example in Figure 1 with attributes  $a_{k_i}$  from three different schemas  $k$ : 1 (red), 2 (yellow), and 3 (green) and their signatures. On the left side of the figure, the linkages of the attribute signatures that are based on the shortest distances are inaccurate. The reason for these mismatches is that textual attributes containing domain-specific abbreviations (e.g., CUST abbreviates CUSTOMER in schema 2) do not result in a meaningful signature that represents the attribute. Currently, existing approaches use the textual descriptors “as-is” to generate signatures of schema elements for linkage generation (Cappuzzo et al., 2020) (Zeakis et al., 2023) (Peeters and Bizer, 2023) which makes the encoding to link attributes that are semantically dissimilar. Therefore, we need a solution to this problem as meaningful attribute representations containing relevant context are fundamental for accurate linkages towards data integration (Papadakis et al., 2020) (Zeng et al., 2024). For example, the attributes OFFICE\_CITY and CUST\_CITY are textually similar and may both contain the concept CITY, but they should not be linked together because they have different semantics ( $\text{OFFICE} \neq \text{CUST}$ ), as shown in Figure 1. Therefore, name-based attribute representations are insufficient for accurate linkages. Therefore,

<sup>a</sup> <https://orcid.org/0009-0000-3039-0685>

<sup>b</sup> <https://orcid.org/0000-0002-2208-0801>

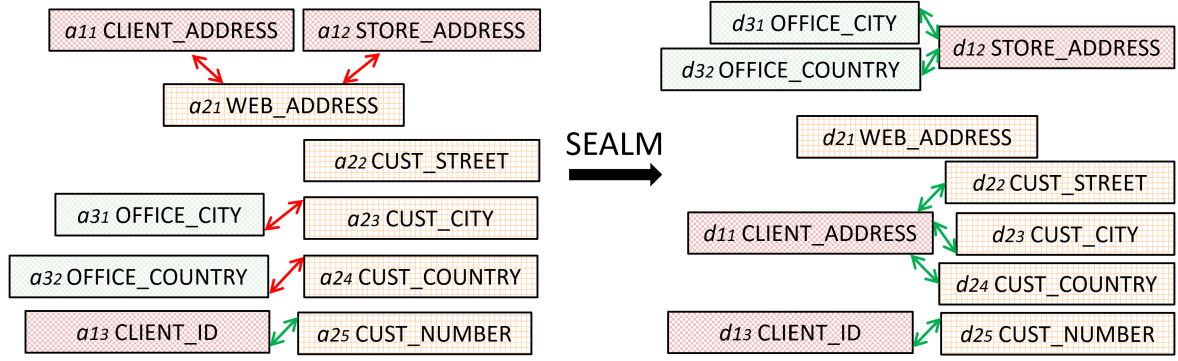


Figure 1: Attribute Signatures placed in a Metric Space with Linkages before and after SEALM based on OC3-HR Schemas.

they must be enriched with context through available metadata such as table names, datatypes, constraints, samples of cell values, domain names, or an expert description in order to recommend relevant linkages.

Additionally, existing approaches explore the full search space of linkage candidates (Narayan et al., 2022) (Peeters and Bizer, 2023) (Remadi et al., 2024) in an attempt to link source-to-target entities between two databases (Zezhou Huang et al., 2024) (Sheetrit et al., 2024). Motivated by the need for meaningful representations of database entities for multiple schemas, we introduce **Semantic Enrichment of Attributes using Language Models**, an approach to generate effective representations and efficient linkages among them. With SEALM, the attributes are represented using semantically rich descriptions  $d_{k_i}$  resulting in corresponding signatures that are more precise, leading to true/accurate linkages as shown on the right-hand side in Figure 1. This is achieved with the help of Generative Language Models, which enrich the description of attributes and lead to higher similarity scores between the relevant linkages (green versus red arrow links). This paper addresses the problem of discovering true linkages between attributes, and its contributions are:

- Defining a scheme of incremental enhancements on attribute signature quality based on metadata information that is available, which varies from little to full access (Section 3.2).
- Introducing SEALM, a method to generate effective attribute descriptions for relational database schema catalogs (Section 3.3).
- Utilizing Blocking in a novel approach that efficiently generates attribute linkages between multi-source database schemas (Section 3.4).
- Evaluating the efficiency of SEALM descriptions and the effectiveness of linkage recommendation between the “OC3-HR” schemas (Section 4).

## 2 RELATED WORK

**Database Enrichment** dates back to (Castellanos and Saltor, 1991) converting relational schemas into an expressive object-oriented model that reflects inclusion and exclusion dependencies of database entities to minimize interaction with a user towards interoperability between multiple and heterogeneous databases. Similarly, (Abdelsalam Maatuk et al., 2010) propose to generate an enhanced Relational Schema Representation (RSR) transposed into a model that captures the essential database characteristics suitable for migration. In our previous work, we introduced *Inteplato*, an unsupervised linkage approach that links similar tables and attributes among different schemas. To boost the linkage accuracy, we enriched the database schemas by retrieving synonyms for table and attribute names to overcome the semantic ambiguity of naming choices by database designers (Traeger et al., 2022).

With the recent advances in **Language Models**, (Fernandez et al., 2023) and (Halevy and Dwivedi-Yu, 2023) envision more automation on data integration as Language Models provide a new paradigm to challenge the underlying syntactical and semantic heterogeneities of data repositories. In the past, research on Entity Resolution and Schema Matching had already used encoder-based Language Models to create embeddings (signature) of schema elements or records that can be used in supervised (Loster et al., 2021) (Zeakis et al., 2023) and unsupervised (Cappuzzo et al., 2020), (Hättasch et al., 2022), (Zeng et al., 2024) linkage approaches, all with the limitation of using the data input “as-is”. In more recent work, (Narayan et al., 2022), (Peeters and Bizer, 2023), and (Remadi et al., 2024) delegate the pair-wise linkage task between two databases to Generative Language Models (GLM) via prompting all potential pairs as a binary classification task. (Sheetrit et al.,

2024) use encoder-based Language Models to generate table and attribute signatures of source and target databases filtered on top linkage candidates subsequently classified by ChatGPT. The previous work that involved GLMs were able to effectively classify linkages. However, in this paper we show that a more efficient linkage candidate selection is needed to provide a scalable solution for multi-source database integration while still being able to use the language synthesis capabilities of GLMs. In this context, (Mihindukulasooriya et al., 2023) and (Zezhou Huang et al., 2024) enrich the source database to a target database or to a business glossary by generating “descriptive table captions, tags, expanded column names that can be mapped to concepts” with a GLM. In contrast, our approach systematically matches more than two database schemas without a target schema given. Recently, (Vogel et al., 2024) collected a corpus of 100,000 real-world databases dubbed “WikiDBs” and renamed tables and attributes using GPT-4o to provide more context. *With SEALM, we generate attribute descriptions at different metadata availability conditions and adapt algorithms that generate linkages between multiple (more than two) databases without a given target ontology or schema (target-free), or pre-annotated linkages (unsupervised), reflecting a real-world schema integration setting.*

### 3 METHODOLOGY

We first define the problem of attribute linkages (Section 3.1) and then define a scheme of information availability of the attribute metadata that varies from little to full access (Section 3.2). Afterwards, we

present our novel approach to **Semantically Enrich Attributes** from relational database schemas with **Language Models** to generate a data catalog with meaningful textual description (Section 3.3). We continue to generate attribute linkages between schemas by generating signatures, a novel multi-source Blocking algorithm, and Filtering (Section 3.4). We assume a schema-aware, multi-source, target-free, and unsupervised linkage environment. Table 1 provides an overview of the notations used in this section.

#### 3.1 Problem Definition

*Attribute Linkages.* We are given  $k$  database schemas  $S_1, S_2, \dots, S_k$  that each contains a heterogeneous set of attributes  $S_k = \{a_{k_1}, a_{k_2}, \dots, a_{k_i}\}$ . The goal is to find all linkages  $L(S) = \{(a_{k_i}, a_{m_j})\}$  between the attributes within the attribute collection of all schemas  $S = S_1 \cup S_2 \cup \dots \cup S_k$ . The true set of attribute linkages contains the attribute pairs that are congruent  $(a_{k_i} \cong a_{m_j}) \Rightarrow r$  by representing a real-world concept with sub-typed or identical semantics.

Given the set of attributes in Figure 1 for example, the attributes  $a_{1_3}$  CLIENT\_ID and  $a_{2_5}$  CUST\_NUMBER are identical, while  $a_{1_1}$  CLIENT\_ADDRESS are sub-typed to  $a_{2_2}$  CUST\_STREET and  $a_{2_3}$  CUST\_CITY because the latter two contain partial semantics of  $a_{1_1}$ .

In this paper, for simplicity we identify linkages between attributes. For data integration based on the Local-as-View paradigm, additional SQL-based transformations are required to generate table linkages between schemas and joins within the schemas (Bleiholder and Naumann, 2009). Although SEALM is a general approach that can also be applied to generate linkages between tables, we leave these extensions for future work.

Table 1: Notations.

Symbol	Description
$S_k = \{a_{k_1}, a_{k_2}, \dots, a_{k_i}\}$	Attributes in schema $k$ .
$a_{k_i} = (n_{k_i}, tn_{k_i}, dc_{k_i}, cv_{k_i}, sn_{k_i})$	Attribute and object-values (name, table name, data type and constraint, record cell values, schema name).
$LM(a_{k_i}, c) \Rightarrow d_{k_i}^c$	SEALM description where $c$ represents the condition on attribute metadata availability.
$S = S_1 \cup S_2 \cup \dots \cup S_k$ for $k$ schemas	Attribute collection from all schemas.
$E(\{d_{k_i}^c = LM(a_{k_i}, c)   \forall a_{k_i} \in S\}) \Rightarrow AS^c = \{\vec{v}_{1_1}, \dots, \vec{v}_{k_i}\}$	Attribute signature set conditioned on $c$ with $ \vec{v}_{k_i} $ based on encoder $E$ .
$B(AS^c, n) \Rightarrow BL = \{(\vec{v}_{k_i}, \vec{v}_{m_j})\}$ where $k \neq m$	Blocking $n$ linkage candidates for each attribute.
$F(BL, k) \Rightarrow FL = \{(\vec{v}_{k_i}, \vec{v}_{m_j}, s)\}$ where $s$ is similarity	Filtering top- $k$ similarity score linkages.
$(a_{k_i} \cong a_{m_j}) \Rightarrow r$ where $k \neq m$	Two congruent attributes representing a real-world concept with sub-typed or identical semantic.
$L(S) = \{(a_{k_i}, a_{m_j})\}$ where $(a_{k_i} \cong a_{m_j}) \Rightarrow r \wedge k \neq m$	All true attribute linkages between schemas.

### 3.2 Metadata Availability on Attributes

We define an attribute as  $a_{k_i}$  containing the object values on the attribute name  $n_{k_i}$ , table name  $tn_{k_i}$ , data type with relational constraints (if one exists)  $dc_{k_i}$ , a sample of maximum five cell values  $cv_{k_i}$ , and schema name  $sn_{k_i}$ . Having full access to metadata within a database environment is a desirable condition for comprehensive data management. However, full access to schema metadata is often impractical due to security and operational risks. We can see this in (Mihindukulasooriya et al., 2023) who observe that “most organizations only permit semantic enrichment processes to access to the table metadata such as column headers and not actual data (i.e., cell values) due to privacy and access control regulations”. Furthermore, not all metadata information might be available, adequately defined, or helpful for linkage recommendation tasks. Motivated by these observations, we create a scheme with six conditions  $C = (c_1, c_2, c_3, c_4, c_5, c_6)$  to represent different types or conditions, of available metadata on attributes, as shown in Table 2.

- $c_1 = (n_{k_i}, tn_{k_i})$ : represents a condition with minimal metadata information exposing only the attribute name  $n_{k_i}$ , and the table name  $tn_{k_i}$ , e.g.,  $a_{23}^{c_1} = \text{“CITY CUSTOMERS”}$ . At this stage, types, constraints, and data are not disclosed, limiting tasks to schema maintenance and auditing.
- $c_2 = (n_{k_i}, tn_{k_i}, dc_{k_i})$ : extends  $c_1$  by exposing data types and constraints  $dc_{k_i}$  of attributes, e.g.,  $a_{23}^{c_2} = \text{“CITY CUSTOMERS STRING”}$ . The data content is still protected but generally less secure as structural constraints relevant to data integrity are revealed.
- $c_3 = (n_{k_i}, tn_{k_i}, dc_{k_i}, cv_{k_i})$ : extends  $c_2$  by exposing the cell values  $cv_{k_i}$  of attributes, e.g.,  $a_{23}^{c_3} = \text{“CITY CUSTOMERS STRING [Strasbourg, ..., Koeln]”}$ . This condition necessitates access control mechanisms to prevent unauthorized data exposure.
- $c_4 = (n_{k_i}, tn_{k_i}, sn_{k_i})$ : extends  $c_1$  basic structural metadata by revealing the schema context  $sn_{k_i}$ , e.g.,  $a_{23}^{c_4} = \text{“CITY CUSTOMERS Order-Customers”}$ . This condition allows for high-level schema documentation without data exposure.

- $c_5 = (n_{k_i}, tn_{k_i}, cv_{k_i}, sn_{k_i})$ : represents a unique condition with all available metadata except the data type and constraints  $dc_{k_i}$  of the attributes often found in Data Lakes, e.g.,  $a_{23}^{c_5} = \text{“CITY CUSTOMERS [Strasbourg, ..., Koeln] Order-Customers”}$ . Data Lakes do not enforce schema constraints in order to handle large amounts of data in a flexible manner.
- $c_6 = (n_{k_i}, tn_{k_i}, dc_{k_i}, cv_{k_i}, sn_{k_i})$ : represents highly usable and least secure access to all attribute metadata, e.g.,  $a_{23}^{c_6} = \text{“CITY CUSTOMERS STRING [Strasbourg, ..., Koeln] Order-Customers”}$ .

### 3.3 SEALM

Trying to solve this problem by directly prompting a Generative Language Model to match all potential attribute linkage candidates (e.g., “Do the attributes ‘CITY’ and ‘ADDRESS’ represent the same concept? Answer with ‘yes’ if they do and ‘no’ if they do not.” as proposed in (Narayan et al., 2022), (Peeters and Bizer, 2023), and (Remadi et al., 2024)) is problematic as  $|S_1| \times |S_2| \times \dots \times |S_k|$  prompts are necessary to cover and classify the full linkage search space, which is not scalable (Section 4). To provide a scalable solution and still use the rich context that Generative Language Models were trained on, we propose to use GLMs to generate a comprehensible data catalog with meaningful attribute descriptions. Adopting the SEALM approach, the number of required prompts is significantly reduced down to the total number of attributes we aim to link  $|S_1| + |S_2| + \dots + |S_k|$ . Afterward, we continue to efficiently recommend linkages using the SEALM-generated descriptions (Section 3.4).

**Prompt-Engineering.** We generate SEALM prompts  $p$  based on the “Prompt” column in Table 2 for each attribute. Given the metadata availability condition  $c \in C$ , we uniformly chain the contextual information of the attribute metadata to specify the textual task (prompt  $p$ ) for a Generative Language Model. For example, the rule-based prompt for  $a_{k_3}$  at condition  $c_1$  and  $c_6$  looks as follows:

- $p_{k_3}^{c_1}$ : “Briefly describe the attribute ‘CITY’ stored in table ‘CUSTOMERS’.”

Table 2: Scheme of Attribute Metadata Availability Condition and SEALM Prompt Constellation.

$a_{k_i}$	Attribute Objects	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$	Prompt (with object-value inserted at *)
$n_{k_i}$	attribute name	✓	✓	✓	✓	✓	✓	Briefly describe the attribute ‘*’
$tn_{k_i}$	table name	✓	✓	✓	✓	✓	✓	stored in the table ‘*’
$dc_{k_i}$	data type and constraint		✓	✓			✓	with the data type ‘*’ and constraint ‘*’
$cv_{k_i}$	cell values			✓		✓	✓	with sampled record cells ‘[*]’
$sn_{k_i}$	schema name				✓	✓	✓	in the context of a ‘*’ database schema and how it might be used



- $p_{k_3}^{c_6}$ : “Briefly describe the attribute ‘CITY’ stored in the table ‘CUSTOMERS’ with the data type ‘STRING’ with sampled records [Strasbourg, Saint Petersburg, Glendale, Nashua, Koeln] in the context of a ‘Order-Customers’ database schema and how it might be used.”

#### Using Generative Language Models (GLMs).

GLMs are ideal for language generation tasks where the input is partial text that requires completion or extensive language synthesis. These models are widely used for creative content generation and chat-bots, which motivates this work to explore their use in automatically generating attribute descriptions in a data catalog. Given the above two prompt examples  $p_{k_3}^{c_1}$  and  $p_{k_3}^{c_6}$ , GLM<sup>1</sup> generated attribute descriptions can look as follows:

- $LM(a_{k_3}, c_1) = d_{k_3}^{c_1}$ : “(...) It seems you’re referring to a hypothetical database table. Assuming that ‘CITY’ is indeed an attribute in the table ‘CUSTOMERS’, here’s a brief description: The ‘CITY’ attribute in the ‘CUSTOMERS’ table likely stores the city where each customer resides. This could be a string value, such as “New York”, “Los Angeles”, or “Chicago”. The data type of this attribute might be a character-based type like VARCHAR or CHAR, with a fixed length or varying lengths depending on the database management system being used.”
- $LM(a_{k_3}, c_6) = d_{k_3}^{c_6}$ : “In the ‘ORDER-CUSTOMERS’ database schema, the ‘CITY’ attribute is stored in the ‘CUSTOMERS’ table with a data type of ‘STRING’. This attribute represents the city where each customer is located. Sampled records: Strasbourg, Saint Petersburg, Glendale, Nashua, Koeln. The ‘CITY’ attribute might be used to: 1. Geographic targeting: The database can be queried to identify customers in specific cities, allowing for targeted marketing campaigns or promotions. 2. Customer clustering: By analyzing the distribution of cities among customer records, the database can group similar customers together based on their geographic locations, enabling tailored sales strategies or loyalty programs. 3. Order fulfillment and logistics: The ‘CITY’ attribute can be used to optimize shipping routes, delivery times, and (...). In summary, the ‘CITY’ attribute in the ‘CUSTOMERS’ table provides valuable insights into customers’ geographic locations, enabling more effective marketing, sales, and logistics strategies.”

<sup>1</sup>Llama3.1:8B on Ollama (Version ID 365c0bd3c000)

A GLM may generate plausible yet nonfactual content, also known as *hallucination* (Huang et al., 2024). We address SEALM’s related attribute enrichment process using GLMs based on our scheme of metadata availability conditions. Given restricted metadata availability (e.g.  $c_1$ ), which is an indisputably legitimate privacy concern, the generated attribute descriptions may suffer from plausible but untrue artifacts, that is, the GLM wrongly generates an attribute’s data type or cell values when compared to the actual database state (hallucinating). In such a case, an inaccurate attribute description may be incorrectly identified as similar to another one. However, as we increase the context of the GLM prompt by supplying metadata conditions  $c_2, \dots, c_6$ , the additional context information of these conditions prevents the GLM from synthesizing inaccurate database schema design choices and lowers the possibility of hallucinations, leading to a contextually accurate attribute description. We compare the raw attribute object value constellation  $a_{k_i}^c \in S$  at a certain condition  $c \in C$  with the corresponding SEALM generated descriptions  $d_{k_i}^c$  towards generating effective linkages in Section 4.

### 3.4 Unsupervised Linkage Recommendation

Our goal is to recommend a set of attribute linkages between multiple schemas without trainable linkage examples. Therefore, we propose a matching pipeline with Signature, a novel multi-source LSH-Blocking algorithm that extends the approach of (Sheetrit et al., 2024), and Filtering in sequential order.

**1. Signature.** In the previous section on SEALM, we used GLMs to generate an attribute description  $d_{k_i}^c$  for a given prompt  $p_{k_i}^c$  at metadata availability condition  $c$ . These models utilize a **decoder-only** transformer-based architecture that internally represents a prompt as an auto-regressive response task to be answered by predicting each word based on all previously generated words. Since we intend to compare the attributes or **SEALM descriptions**, e.g.,  $d_{k_1}^{c_1}$  and  $d_{m_1}^{c_1}$  on similarities, there is a need to transform these back into a structured numeric embedding that can be compared efficiently.

This is where **encoder-decoder** based **Language Models** become necessary. The encoder component of these models is designed to take input text and encode it into a multi-dimensional and fixed-sized vector (**signature**) that captures the semantic and syntactic nuances of the attribute object-values  $a_{k_1}$  or the SEALM descriptions  $d_{k_1}$  that we define as  $\vec{v}_{k_i}$ . Subsequently, the decoder’s role is to regenerate the

original text with a low reconstruction error. In this work, we mainly focus on the encoder function from compatible Language Models that we define as  $E$ . This function is uniformly applied to each attribute  $a_{k_i}^c \in S$  or description  $d_{k_i}^c \in S$  given a condition  $c \in C$  on metadata availability and results in a set of attribute signatures  $AS^c$ .

**2. Multi-source LSH-Blocking.** In the case of integration with multiple schemas, the computational cost of pair-wise distances and similarities becomes impractical. Computing  $|S_1| \times |S_2| \times \dots \times |S_k|$  comparisons quickly becomes infeasible to scale with large numbers of schemas and attributes. Approximate Nearest Neighborhood (ANN) algorithms reduce this complexity to handle large-scale data. We focus in this work on the ANN-related **locality-sensitive hashing (LSH)** technique that hashes  $n$  most similar signatures into a “bucket” with high probability. Various generic algorithms have been implemented by companies such as Meta with FAISS (Facebook AI Similarity Search)<sup>2</sup> and Spotify with Voyager<sup>3</sup>. These methods drastically reduce the number of comparisons needed to efficiently provide effective recommendations in the social and audio domains.

---

Algorithm 1: Multi-source LSH-Blocking.

---

**Input:**  $AS^c = \{\vec{v}_1, \dots, \vec{v}_{k_i}\}, n \triangleright$  Attribute signature set, custom number linkage candidates  
**Output:**  $BL = \{(\vec{v}_{k_i}, \vec{v}_{m_j})\}$  where  $k \neq m \triangleright$  Set of  $n$  blocked linkages per attribute signature

```

1:  $SK \leftarrow \emptyset$   $\triangleright$  Initialize schema key set
2:  $BL \leftarrow \emptyset$   $\triangleright$  Initialize blocked linkage set
3: for  $k$  in schemas do
4:    $SK \leftarrow SK \cup \{k\}$   $\triangleright$  Add schema identifier
5:    $I \leftarrow \text{LSH}(|\vec{v}|)$   $\triangleright$  Initialize LSH index with uniformed signature length predefined by encoder-LM
6:    $I.\text{set}\{\vec{v}_{k_i} | \forall \vec{v}_{k_i} \in AS^c \wedge k \text{ not in } SK\}$   $\triangleright$  Set signatures from different schemas to LSH index
7:   for  $i$  in attribute signatures do
8:      $BL_{k_i} \leftarrow I.\text{search}(\vec{v}_{k_i}, n)$   $\triangleright$  Search for  $n$  most similar signatures through index and set linkage candidates bucket  $(\vec{b}_1, \dots, \vec{b}_n)$ 
9:     for  $\vec{b}$  in  $BL_{k_i}$  do
10:       $BL \leftarrow BL \cup \{(\vec{v}_{k_i}, \vec{b})\}$   $\triangleright$  Add linkage candidate
11:   end for
12: end for
13: end for
14: return  $BL$ 

```

---

<sup>2</sup><https://ai.meta.com/tools/faiss/>

<sup>3</sup><https://spotify.github.io/voyager/>

Due to the large search space of potential attribute linkages, we adapt the LSH method in **Algorithm 1** to efficiently recommend a bucket of inter-schema linkage candidates  $BL = \{(\vec{v}_{k_i}, \vec{v}_{m_j})\}$  and accommodate **multiple schemas** as input so that  $k \neq m$ . The inputs to our Blocking algorithm  $B$  are  $AS^c$ , the encoded set of attribute signatures, and  $n$ , the custom number of the most similar signatures per attribute. We begin to iterate over each schema  $k$  (Line 3) and assign the set of signatures that do not originate from the same schema as  $k$  to the LSH index. This assignment task includes one or multiple LSH functions that compress the high-dimensional signatures into a lower dimension so that similar ones are hashed into the same bucket with a higher probability (Lines 4-6). Then, each attribute signature  $\vec{v}_{k_i}$  from schema  $k$  is set as a query item. *At search, the query item is also hashed to check for potential neighboring signatures with similar hash keys, which, consequently, avoids directly comparing the query item with every other signature* (Lines 7-8). The result is a bucket set  $BL_{k_i} = (\vec{b}_1, \dots, \vec{b}_n)$  of size  $n$  with the most similar attributes as linkage candidates of which each is added as a tuple of signature pairs  $(\vec{v}_{k_i}, \vec{b})$  where  $\vec{b} \in BL_{k_i}$  to the globally blocked linkage set  $BL$  (Lines 9-10). We highlight that our algorithm generates linkages with attribute pairs  $\{(\vec{v}_{k_i}, \vec{v}_{m_j})\}$  in the order of the iteration of the schemas as we want to avoid recommending an identical attribute linkage twice (e.g.,  $(a_{k_1}^{c_1}, a_{m_1}^{c_1})$  and  $(a_{m_1}^{c_1}, a_{k_1}^{c_1})$ , just in reverse order). *As a result, the attributes that have been used as a query item do not need to be set to the LSH index.*

**3. Filtering.** Now that we have a much more condensed set of  $n$  likely matching linkages for each attribute,  $BL$  still contains several linkages, of which not all are relevant or correct. For example, Blocking does not consider that an attribute may only be linked to a single different schema and none of the others. Furthermore, some attributes may not be linked to any other attributes as they represent a unique concept to its originating schema.

To consider the above-described cases and prioritize recommending very similar (close distanced) attribute signatures, we apply Filtering to generate a more relevant linkage set  $FL$  that we describe in **Algorithm 2**. The inputs to this algorithm are  $BL$ , the blocked linkage attributes, and top- $k$ , a custom number that cuts off irrelevant linkages. For each blocked attribute linkage, we compute a similarity function  $s$  (e.g. Cosine similarity  $\in [0..1]$ ) to quantitatively express how distant or close two attribute signatures from different schemas are, resulting in the linkage set  $FL = \{(\vec{v}_{k_i}, \vec{v}_{m_j}, s)\}$  (Lines 1-5). We continue to sort

Algorithm 2: Filtering top- $k$  Attribute Linkages.

---

**Input:**  $BL = \{(\vec{v}_{k_i}, \vec{v}_{m_j})\}$ , top- $k$   $\triangleright$  Blocked linkage set, custom number to filter top- $k$  scored linkages

**Output:**  $FL = \{(\vec{v}_{k_i}, \vec{v}_{m_j}, s)\}$  where  $s$  is similarity  $\triangleright$  Set of top- $k$  similarity filtered linkages

- 1:  $FL \leftarrow \emptyset$   $\triangleright$  Initialize filtered linkage set
- 2: **for**  $(\vec{v}_{k_i}, \vec{v}_{m_j})$  in  $BL$  **do**
- 3:    $s \leftarrow \text{similarity}(\vec{v}_{k_i}, \vec{v}_{m_j})$   $\triangleright$  Compute similarity
- 4:    $FL \leftarrow FL \cup \{(\vec{v}_{k_i}, \vec{v}_{m_j}, s)\}$
- 5: **end for**
- 6:  $FL \leftarrow \{(\vec{v}_a, \vec{v}_b, s_y), (\vec{v}_c, \vec{v}_d, s_z), \dots\} \in FL \wedge s_y > s_z$   $\triangleright$  Sort linkage triplet descending on similarity score
- 7:  $FL \leftarrow \{t_1, t_2, \dots\}$  with  $t_k \in FL \wedge k \leq \text{top-}k$   $\triangleright$  Filter top- $k$  similarity scored linkages
- 8: **return**  $FL$

---

the linkage triplets based on the similarity score  $s$  in descending order (Line 6) and subsequently filter the top- $k$  linkages (Line 7). In a more abstract sense, Filtering effectively minimizes the operational and cognitive overload on a human by recommending a precise linkage set needed for data integration. We refer the reader to the survey paper by (Papadakis et al., 2020) for more details on Blocking and Filtering in the context of Entity Resolution and Linkage.

## 4 EVALUATION

In this section, we present the evaluation of our proposed research based on the experiments we conducted. We first describe the experimental dataset and then provide the configuration details of the SEALM, Signature, Blocking, and Filtering methods. Then, we present the evaluation metrics. To the best of our knowledge, we are the first to apply SEALM and its methods within the unsupervised, multi-source, and target-free Schema Matching research space. All experiments are conducted in a Python Jupyter Notebook on an Intel i7-1265U CPU with 32GB memory. The datasets and code can be found at <https://github.com/leotraeg/SEALM>.

**Dataset:** We conduct distinct experiments with two datasets that contain multiple schemas on **Orders**, **Customers**, and **Human Resources** from the three different database vendors Oracle, MySQL, and SAP HANA (Traeger et al., 2024).

- The “OC3” dataset contains a domain-specific set of three Order-Customer schemas (43+59+40=142 attributes) with **47** true inter-schema linkages out of **6617** potential linkage candidates.
- The “OC3-HR” dataset extends the domain-

specific schemas with a Human-Resources schema, which comes from a completely different application domain (142+35=177 attributes) that contains 15 additional inter-schema linkages and results in overall **62** true inter-schema linkages out of **11587** potential linkage candidates.

**Methods.** We compare the effectiveness of attribute linkages for OC3 and OC3-HR using the state-of-the-art (SOTA) approach based on *attribute signatures* encoded on the raw object values (name, table name, data type and constraint, record cell values, schema name) versus linkages using *description signatures* encoded on the SEALM-generated descriptions.

- **SEALM:** We engineer prompts at six different conditions  $c \in C$  corresponding to various levels of metadata availability as described in Section 3.3. For each schema, we initialize Meta’s tiniest open-source GLM Llama3.1:8B (Version ID 365c0bd3c000) that we locally host via Ollama<sup>4</sup> and prompt “Your task is to describe attributes from heterogeneous relational databases based on extracted schema metadata to improve linkages for Data Integration.” Then, we sequentially prompt and retrieve the respective attribute descriptions  $d_{k_i}^c$ .
- **Signatures:** We generate attribute signatures using the attribute  $d_{k_i}^c$  object values (SOTA) and compare these with the SEALM-based description signatures  $d_{k_i}^c$  over the range of all six conditions  $c \in C$  on metadata levels of availability. Therefore, we use the encoder-based Sentence Transformer Bert Language Model (Reimers and Gurevych, 2019)<sup>5</sup> often used in the Entity Resolution research area (Cappuzzo et al., 2020) (Zeakis et al., 2023) (Peeters and Bizer, 2023) to encode the various textual attribute descriptions into fixed-sized 768-dimensional signatures. Finally, we normalize the signatures for each conditioned set  $AS^c$  dimension-wisely into a  $[0..1]$  range.
- **Blocking:** We implement Algorithm 1 with Meta’s LSH-based similarity-search module (FAISS) as it has been used in recent research on source-to-target Entity Resolution for records (Papadakis et al., 2020), (Paulsen et al., 2023), (Zeakis et al., 2023). Our algorithm generates the linkage set  $BL$  with the approximate  $n$  nearest attributes between multiple schemas.
- **Filtering:** We implement Algorithm 2 in order to further reduce the linkage set  $BL$  to the top- $k$  similarity scored linkages  $FL$ .

<sup>4</sup><https://ollama.com>

<sup>5</sup><https://sbnet.net/grtr5-base>

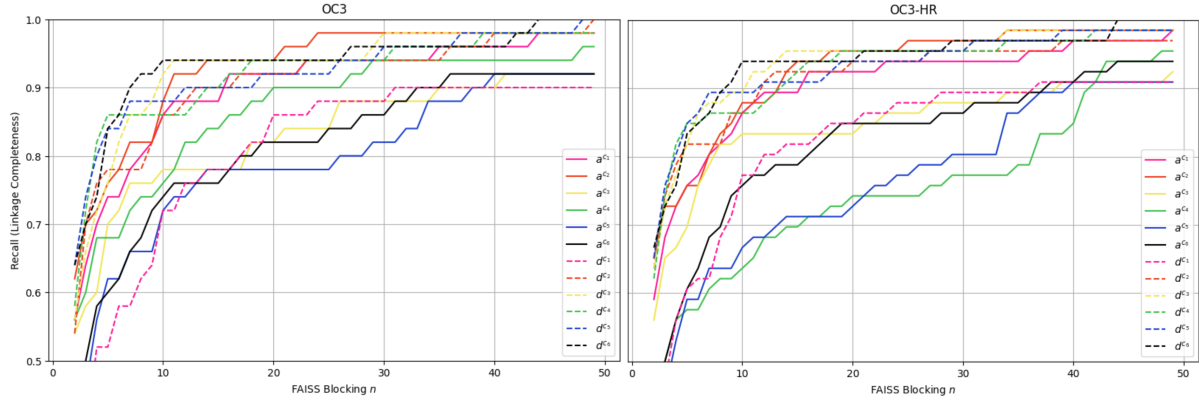


Figure 2: Evaluating Recall on Blocking  $n \in [2..50]$  with Attribute  $a^c$  and SEALM  $d^c$  Signatures in OC3-HR Schemas by Metadata Availability Condition.

**Metrics.** Using OC3 and OC3-HR datasets we compare the recommended set of blocked  $BL$  and subsequently filtered linkages  $FL$  to the set of ground truth linkages.

- Recall (*Linkage Completeness*): First, we measure the recall of the generated linkages over the range of the  $n$  nearest attributes for the blocked linkages  $|BL_{true}|/|L(S)_{true}|$  and, subsequently, over the top- $k$  for the filtered linkages  $|FL_{true}|/|L(S)_{true}|$ .
- Precision (*Linkage Quality*): We measure the precision of the generated linkages over the top- $k$  filtered linkages  $|FL_{true}|/k$ . We refrain from evaluating the precision of blocked linkages as, without Filtering, too many false linkages remain in the blocked  $BL$  set.

**Blocking Results.** The experimental results on Blocking attribute linkages in OC3 (left) and OC3-HR (right) schemas are shown in Figure 2. We measure the recall (linkage completeness) at the y-axis for  $n \in [2..50]$  linkage candidates at the x-axis for the signatures with the raw attribute values  $a^c$  (straight lines) and SEALM-generated descriptions  $d^c$  (dashed lines), colored in the six different metadata availability conditions  $c_1$  (pink),  $c_2$  (orange),  $c_3$  (yellow),  $c_4$  (green),  $c_5$  (blue), and  $c_6$  (black). On OC3 schemas (left), attribute  $a^{c3}$ ,  $a^{c4}$ ,  $a^{c5}$ ,  $a^{c6}$  and SEALM  $d^{c1}$ ,  $d^{c2}$  signatures generate fewer true attribute linkages than others over the full range of  $n$ . The gap in recall for these signatures becomes even more visible for the OC3-HR schemas. We conclude that Blocking attribute signatures yield the best recall if they are encoded based on their name, table name  $a^{c1}$ , and include the data type and constraint  $a^{c2}$ . On the contrary, Blocking SEALM signatures yield the best recall when the GLM additionally processes the data type and constraint with cell

values  $d^{c3}$ , the schema name  $d^{c4}$ , or in combination with  $d^{c5}$  and  $d^{c6}$ . At  $n \in [5..10]$ , blocking SEALM signatures  $d^{c3-6}$  reach 85% to 95% recall while the best performing attribute signatures  $a^{c1}$  and  $a^{c2}$  generate 10% fewer true linkages. At  $n = 25$ , blocking signatures  $a^{c1}$ ,  $d^{c3}$ ,  $d^{c4}$ ,  $d^{c5}$ ,  $d^{c6}$  generate 95% recall with only the  $a^{c2}$  exceeding by approximately 4%/2% for the OC3/OC3-HR schemas. At  $n = 30$ , the SEALM signature  $d^{c3}$  (OC3) and signatures  $d^{c3}$ ,  $d^{c5}$ , and  $d^{c6}$  (OC3-HR) align with the recall performance of  $a^{c2}$ . Afterward, blocking  $d^{c6}$  signatures gradually performs better in recall, reaching 100% at  $n = 44$  for both OC3 and OC3-HR schemas.

In real-world integration scenarios, knowing the blocking value  $n$  beforehand would imply knowing the ground truth of attribute linkages. To fairly compare the performance of Blocking signatures and select a relevant subset between the twelve signatures for the subsequent Filtering phase, we compute the Area Under the Curve (AUC) Recall in Table 3. *Summary:* Over the range of  $n \in [2..50]$ , the six best AUC Recall for Blocking linkages in OC3 and OC3-HR schemas use attribute signatures  $a^{c1}$  and  $a^{c2}$  and SEALM signatures  $d^{c3}$ ,  $d^{c4}$ ,  $d^{c5}$ , and  $d^{c6}$ . Blocking the respective SEALM signatures over the full range of  $n$  yields higher recall than with the attribute signatures except at  $n \in [25..30]$ . At higher  $n$  with Blocking, SEALM signatures contain all true linkages, while attribute signatures cut off a few ones that we discuss in the next paragraph.

Table 3: Evaluating Area Under Curve Recall on Blocking  $n \in [2..50]$  with Attribute  $a^c$  and SEALM  $d^c$  Signatures in OC3-HR Schemas by Metadata Availability Condition.

Schemas	Signature	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$
OC3	SOTA: $d_{k_i}$	<b>42.51</b>	<b>43.82</b>	39.19	40.90	37.38	38.39
OC3	SEALM: $d_{k_i}$	38.00	42.77	<b>43.75</b>	<b>43.42</b>	<b>43.44</b>	<b>44.04</b>
OC3-HR	SOTA: $d_{k_i}$	<b>42.73</b>	<b>43.73</b>	39.97	35.42	36.13	38.83
OC3-HR	SEALM: $d_{k_i}$	38.97	43.32	<b>44.23</b>	<b>43.77</b>	<b>43.88</b>	<b>44.19</b>



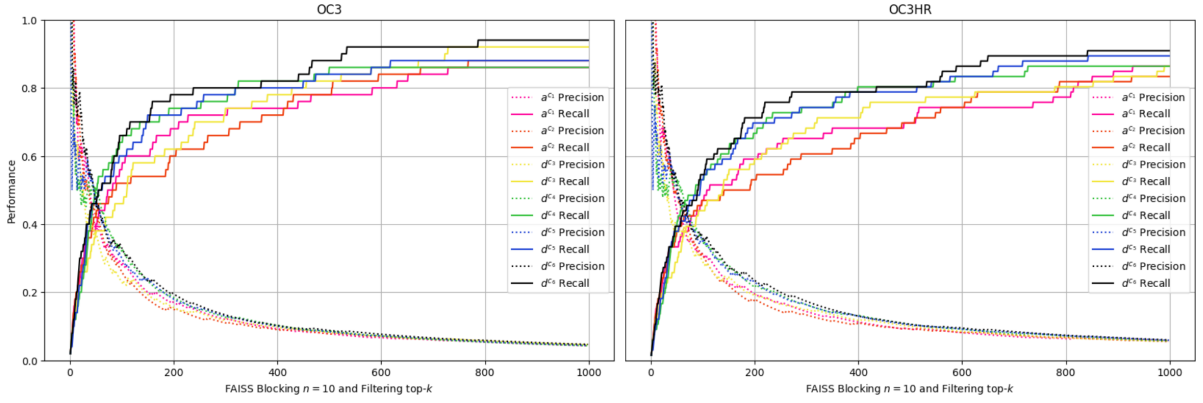


Figure 3: Evaluating Precision and Recall on Blocking  $n = 10$  and Filtering top- $k \in [1..1000]$  with Attribute  $a^{c1}$  and  $a^{c2}$  and SEALM  $d^{c3}, d^{c4}, d^{c5}$ , and  $d^{c6}$  Signatures in OC3-HR Schemas by Metadata Availability Condition.

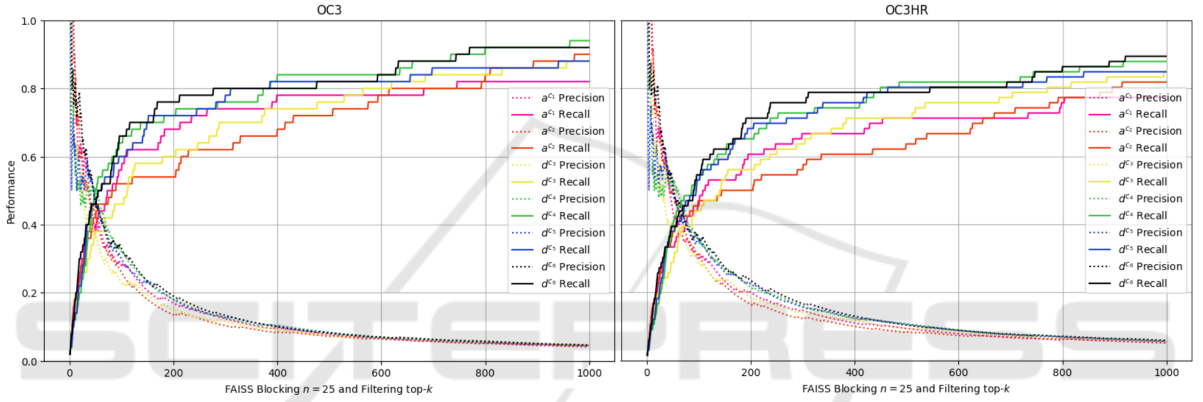


Figure 4: Evaluating Precision and Recall on Blocking  $n = 25$  and Filtering top- $k \in [1..1000]$  with Attribute  $a^{c1}$  and  $a^{c2}$  and SEALM  $d^{c3}, d^{c4}, d^{c5}$ , and  $d^{c6}$  Signatures in OC3-HR Schemas by Metadata Availability Condition.

**Filtering Results.** We evaluate the precision (dotted) and recall (straight) performance (y-axis) on Filtering the top- $k \in [1..1000]$  linkages (x-axis) using the Cosine similarity for each of the top-performing Blocking subset of attribute  $a^{c1}$  (pink) and  $a^{c2}$  (orange) and SEALM-based  $d^{c3}$  (yellow),  $d^{c4}$  (green),  $d^{c5}$  (blue), and  $d^{c6}$  (black) signatures. As we do not know the ideal  $n$  value for the prior Blocking phase for the OC3 (left) and OC3-HR (right) schemas, we evaluate two Filtering experiments with blocked linkage sets at  $n = 10$  (Figure 3) and at  $n = 25$  (Figure 4). At the start of top- $k$ , all experiments filter linkage sets with 100% precision. Notably, the SEALM signatures  $d^{c3}$  and  $d^{c4}$  fluctuate and fall below the precision performance of the signatures  $a^{c1}$ ,  $a^{c2}$ ,  $d^{c5}$ , and  $d^{c6}$ . At approximately  $k = 44$ , the precision and recall graphs of all signatures intersect at 45-50% (OC3) and 38-43% (OC3-HR) on the y-axis, reflecting that OC3-HR schemas are the more challenging integration scenario. Within the range  $k \in [44..1000]$ , Blocking and Filtering the SEALM signatures  $d^{c4}$ ,  $d^{c5}$ , and  $d^{c6}$  yields 5% to

20% improvement in recall compared to  $d^{c3}$  and the attribute signatures  $a^{c1}$  and  $a^{c2}$ . Even though there is a minor recall improvement in Blocking the attribute signatures  $a^{c2}$  at  $n = 25$ , Filtering the corresponding top-scored linkages generates more false linkages than with SEALM-based description signatures.

**Summary:** Overall, the SEALM description signature  $d^{c6}$  performs the best within the Blocking and Filtering pipeline among both OC3 and OC3-HR schemas with minor SEALM exceptions  $d^{c4}$  and  $d^{c5}$ . The corresponding SEALM description of  $d^{c6}$  requires full metadata availability that includes the attribute name, table name, data type and constraints, cell values, and schema name. If cell values of the attributes are not disclosed due to access control regulations, the SEALM signature  $d^{c4}$  should be used for representing the attributes followed by  $a^{c1}$  and  $a^{c2}$ . Overall, neither the attribute nor SEALM signatures provide a linkage set that contains all true linkages due to the parameters of Blocking ( $n \leq 26$ ) and Filtering (top- $k \leq 1000$ ) cutting off a few ground truth linkages with low similarities.

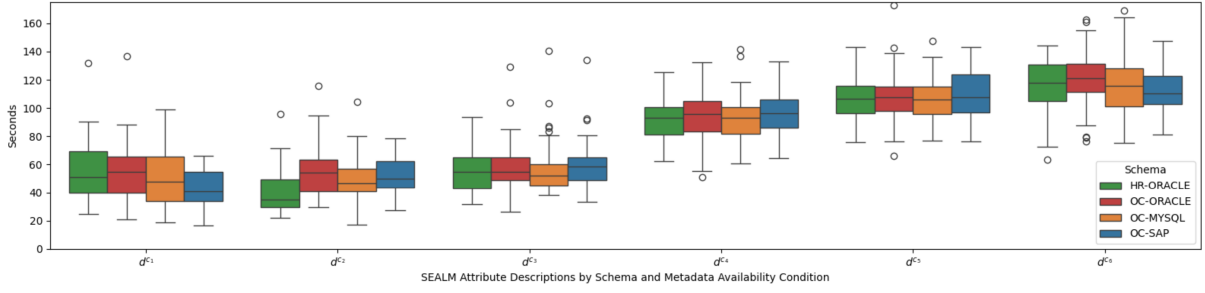


Figure 5: Processing Time for Invoked SEALM Attribute Descriptions in OC3-HR Schemas by Metadata Availability Condition with Llama3.1:8B.

Expanding the linkage search space by setting these parameters higher would generate more linkage candidates that cover all ground truths. The more complex OC3-HR linkages that were cut off contain semantically challenging attribute pairs such as REPORTS TO and MANAGER ID or TERRITORY and REGION NAME. With SEALM-generated descriptions, we identify and recommend more of such sub-typed attribute linkages that are more nuanced.

The improvement in recall for linkage recommendation is attributed to the semantically rich descriptions that are used for encoding the attribute signatures, bringing similar attributes closer together and distinguishing dissimilar ones more effectively. Despite the computational cost to invoke GLM prompts, the resulting increase in the quality of attribute signatures significantly enhances the recommendation process. We show the time needed in seconds for invoking SEALM prompts using the tiniest open-source GLM by Meta (Llama3.1:8B) hosted on a laptop without GPU acceleration in Figure 5 as colored boxplots of the HR-ORACLE (green), OC-ORACLE (red), OC-MYSQL (yellow), and OC-SAP (blue) schemas. With this experimental set-up, Llama3.1:8B requires approximately 50 to 58 seconds to describe attributes with SEALM prompts based on their name, table name, data type and constraint, and cell values at conditions  $c_1$ ,  $c_2$ , and  $c_3$ . By adding the schema name with the suffix “and how it might be used”, represented with prompts at conditions  $c_4$ ,  $c_5$ , and  $c_6$ , we observe that the processing time doubles to ap-

proximately 93 to 115 seconds. No pattern indicates a faster or slower generation of attribute descriptions among the schemas, whether it is related to Orders, Customers, or Human Resources.

**Scalability:** Additionally, we show the mean and overall processing time for invoking the GLM for SEALM attribute descriptions for the OC3 and OC3-HR schemas in Table 4. We can see that SEALM prompts are processed for each attribute from all the schemas to be integrated  $|S_1| + |S_2| + \dots + |S_k|$ . We show this linearity in Figure 6 with the cumulative sum in hours for invoked SEALM descriptions by the attributes in OC3-HR schemas.

Let us assume that we compare  $k$  different schemas to be integrated  $S_1, S_2, \dots, S_k$ , there is one schema with the maximum number of attributes, which we denote as  $M = |S_k|$ . Then, the resulting number of needed SEALM prompts requires at most linear complexity  $O(M \cdot k)$ . This becomes relevant if we consider an evolving multi-database system that needs to identify correct linkages in a reoccurring integration process with  $k + 1$  newly participating schemas. We highlight that the generated SEALM descriptions of a schema  $k$ , once processed, can be continuously reused as they are unaffected by attributes from a new schema  $k + 1$ . However, attribute descriptions that were enriched with different GLMs or signatures that were encoded with different encoder-based Language Models may not be comparable and, thus, may indicate a weak similarity for actually similar attributes. In addition to using the GLMs’ descriptive language capabilities with SEALM, our unsu-

Table 4: Processing Time for SEALM Prompts in OC3-HR Schemas by Metadata Availability Condition.

Phase/Condition	$c_1$	$c_2$	$c_3$	$c_4$	$c_5$	$c_6$
SEALM Prompt( $d_{k_i}$ ) (LM=Llama3.1:8B) $\mu$ in seconds	51.84	49.64	57.49	93.79	108.32	115.75
OC3 ×142 Attributes $\Sigma$ in hours	1.96	2.05	2.28	3.72	4.29	4.58
OC3-HR ×177 Attributes $\Sigma$ in hours	2.53	2.46	2.82	4.61	5.32	5.70

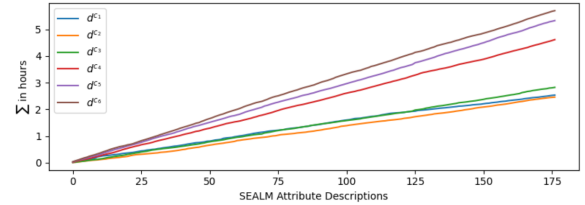


Figure 6: Linear Cumulative Processing Time for Invoked SEALM Attribute Descriptions in OC3-HR Schemas by Metadata Availability Condition with Llama3.1:8B.

pervised Signature, Blocking, and Filtering approach efficiently covers the entire attribute linkage search space to provide relevant linkage recommendations.

Based on work by (Narayan et al., 2022), (Peeters and Bizer, 2023), and (Remadi et al., 2024), the alternative GLM approach would be to compare and classify all potential attribute pairs that may exist, such as with the prompt “Do the attributes ‘ $a_{k_i}^c$ ’ and ‘ $a_{m_j}^c$ ’ represent the same concept? Answer with ‘yes’ if they do and ‘no’ if they do not.” While their approach would also cover the entire linkage search space, it represents a brute-force solution that requires  $|S_1| \times |S_2| \times \dots \times |S_k|$  comparisons in order to solve the *Attribute Linkages* problem defined in Section 3.1. The multiplicative nature of brute-force comparisons between attributes implies exponential growth in the order of  $O(M^k)$ . *Consequently, in the context of multi-source schema integration, a brute-force approach is not scalable. For example, exploring all 11578 possible linkages for the relatively small OC3-HR schemas for the condition  $c_6$  on metadata availability would lead to more than 100 computation hours. Also, randomly selecting linkage samples does not guarantee coverage of the entire space, leading to the loss of all linkages that are outside of the sample.*

On the contrary, our approach is quite scalable growing in a linear fashion since we first generate semantically enriched attribute descriptions via SEALM and then recommend relevant linkages via Signature, Blocking, and Filtering.

## 5 CONCLUSIONS

This paper introduces SEALM, a new method in the EL pipeline that generates Semantically Enriched Attribute descriptions using Language Models based on various levels of metadata availability ranging from highly-secure to full-exposure access. SEALM-generated attribute descriptions can be used as Signatures to efficiently generate linkages between multiple heterogeneous schemas by taking advantage of our novel Blocking algorithm, and Filtering. We evaluated the raw attribute metadata values (SOTA) with SEALM descriptions between two different multi-source schema matching scenarios using the OC3 and OC3-HR schemas at different ranges of Blocking and Filtering configurations and observed a significant 5% to 20% recall improvement in linkage recommendations.

The SEALM methodology can be applied to arbitrary data repositories, and its approach can be adapted to generate linkages for different schema components. Dealing with a search space of link-

ages scales problematically with the Cartesian product size when integrating more than two database schemas. Our SEALM approach uses Generative Language Models that need to process only a smaller number of attributes of the integrated schemas, thus scaling up nicely in a linear fashion. We efficiently combine the rich language synthesis capabilities of Generative Language Models with scalable Schema Matching and Entity Linkage methods, a major deviation from prior research techniques.

Looking ahead, several improvements can be made through (1) Prompt Engineering with more powerful Generative Models. Specializing GLMs to relational schemas on recently available real-world database corpora, such as GitSchemas (Döhmen et al., 2022) and WikiDBs (Vogel et al., 2024), could lead to improved data cataloging capabilities. (2) Similarly, encoding the descriptions into more effective signatures may be achieved by fine-tuned encoder-based Language Models on the basis of database corpora. (3) Finally, additional methods such as Scoping (Traeger et al., 2024) can improve the efficiency and effectiveness of the linkage pipeline.

## ACKNOWLEDGEMENTS

Leonard Traeger was partially supported by a Technology Catalyst Fund TCF24KAR11131049602 by UMBC and a grant project PAn.CV (reference number 03FHP109) by the German Federal Ministry of Education and Research (BMBF) and Joint Science Conference (GWK).

## REFERENCES

- Abdelsalam Maatuk, M., Ali, A., and Rossiter, N. (2010). Semantic Enrichment: The First Phase of Relational Database Migration. In *Innovations and Advances in Computer Sciences and Engineering*, pages 373–378, Dordrecht. Springer Netherlands.
- Bleiholder, J. and Naumann, F. (2009). Data fusion. *ACM Computing Surveys*, 41(1):1–41.
- Cappuzzo, R., Papotti, P., and Thirumuruganathan, S. (2020). Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. *SIGMOD ’20*, pages 1335–1349, New York, NY, USA. ACM.
- Castellanos, M. and Saltor, F. (1991). Semantic enrichment of database schemes: an object oriented approach. In *[1991] Proceedings. First International Workshop on Interoperability in Multidatabase Systems*.
- Döhmen, T., Hulsebos, M., Beecks, C., and Schelter, S. (2022). GitSchemas: A Dataset for Automating Relational Data Preparation Tasks. In *2022 IEEE 38th*

- International Conference on Data Engineering Workshops (ICDEW)*, pages 74–78. ISSN: 2473-3490.
- Fernandez, R. C., Elmore, A. J., Franklin, M. J., Krishnan, S., and Tan, C. (2023). How Large Language Models Will Disrupt Data Management. *VLDB*, 16(11):3302–3309.
- Halevy, A. and Dwivedi-Yu, J. (2023). Learnings from Data Integration for Augmented Language Models. arXiv:2304.04576 [cs].
- Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B., and Liu, T. (2024). A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*, page 3703155.
- Hättasch, B., Truong-Ngoc, M., Schmidt, A., and Binnig, C. (2022). It's AI Match: A Two-Step Approach for Schema Matching Using Embeddings. arXiv:2203.04366 [cs].
- Loster, M., Koumarelas, I., and Naumann, F. (2021). Knowledge Transfer for Entity Resolution with Siamese Neural Networks. *Journal of Data and Information Quality*, 13(1):1–25.
- Mihindukulasooriya, N., Dash, S., and Bagchi, S. (2023). Unleashing the Potential of Data Lakes with Semantic Enrichment Using Foundation Models. In *ISWC Industry Track CEURWP*, Athens, Greece.
- Narayan, A., Chami, I., Orr, L., Arora, S., and Ré, C. (2022). Can Foundation Models Wrangle Your Data? arXiv:2205.09911 [cs].
- Papadakis, G., Skoutas, D., Thanos, E., and Palpanas, T. (2020). Blocking and Filtering Techniques for Entity Resolution: A Survey. *ACM Comput. Surv.*, 53(2):31:1–31:42.
- Paulsen, D., Govind, Y., and Doan, A. (2023). Sparkly: A Simple yet Surprisingly Strong TF/IDF Blocker for Entity Matching. *VLDB*, 16(6):1507–1519.
- Peeters, R. and Bizer, C. (2023). Using ChatGPT for Entity Matching. arXiv:2305.03423 [cs].
- Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.
- Remadi, A., El Hage, K., Hobeika, Y., and Bugiotti, F. (2024). To prompt or not to prompt: Navigating the use of Large Language Models for integrating and modeling heterogeneous data. *Data & Knowledge Engineering*, 152:102313.
- Saeedi, A., David, L., and Rahm, E. (2021). Matching Entities from Multiple Sources with Hierarchical Agglomerative Clustering. In *Proceedings of the 13th IC3K*, pages 40–50. SciTePress.
- Sheetrit, E., Brief, M., Mishaeli, M., and Elisha, O. (2024). ReMatch: Retrieval Enhanced Schema Matching with LLMs. arXiv:2403.01567 [cs].
- Traeger, L., Behrend, A., and Karabatis, G. (2022). In-teplato: Generating mappings of heterogeneous relational schemas using unsupervised learning. In *2022 CSCI*, pages 426–431.
- Traeger, L., Behrend, A., and Karabatis, G. (2024). Scoping: Towards Streamlined Entity Collections for Multi-Sourced Entity Resolution with Self-Supervised Agents. pages 107–115.
- Vogel, L., Bodensohn, J.-M., and Binnig, C. (2024). WikiDBs: A Large-Scale Corpus Of Relational Databases From Wikidata.
- Zeakis, A., Papadakis, G., Skoutas, D., and Koubarakis, M. (2023). Pre-Trained Embeddings for Entity Resolution: An Experimental Analysis. *VLDB*, 16(9):2225–2238.
- Zeng, X., Wang, P., Mao, Y., Chen, L., Liu, X., and Gao, Y. (2024). MultiEM: Efficient and Effective Unsupervised Multi-Table Entity Matching. pages 3421–3434. IEEE Computer Society.
- Ze Zhou Huang, Guo, Jia, and Wu, Eugene (2024). Transform Table to Database Using Large Language Models. *2nd International Workshop TaDA@VLDB*.