# Schema Matching using Zero-Shot LLMs

Houming Chen*
houmingc@umich.edu
The University of Michigan
Ann Arbor, Michigan, USA

Zhuocheng Sun*
sunzc@umich.edu
The University of Michigan
Ann Arbor, Michigan, USA

Yifan Zhu*
zhuyifan@umich.edu
The University of Michigan
Ann Arbor, Michigan, USA

## ABSTRACT

Schema matching is an important problem that has garnered attention for over two decades. Motivated by recent advancements in zero-shot large language models (LLMs) like ChatGPT, we explore the application of zero-shot LLMs in schema matching.

First, we propose a simple method employing zero-shot LLMs for schema matching, which significantly outperforms existing state-of-the-art schema matching methods.

Additionally, we have conducted experiments to investigate the impact of several well-known prompt engineering techniques on our LLM-based approach. Our findings reveal that certain techniques can further improve the preformance of our method.

## CCS CONCEPTS

• **Information systems** → **Relational database model**; • **Computing methodologies** → **Natural language processing**.

## KEYWORDS

Schema Matching, Large Language Models, Prompt Engineering, GPT

**ACM Reference Format:**
Houming Chen, Zhuocheng Sun, and Yifan Zhu. 2023. Schema Matching using Zero-Shot LLMs. In *Proceedings of (EECS584).* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 INTRODUCTION

Schema matching within relational databases is an important problem that has garnered attention for over two decades. It is crucial for various database applications, including data integration, data warehousing, and e-commerce [18].

This process involves identifying relationships between components of different schemata, aiming to establish semantic equivalences between their elements. An example of schema matching is provided in Figure 1. In this example, schemata from two relational databases, labeled A and B, both contain graduate student information. Schema matching refers to pinpointing attributes that represent the same concepts in both schemata. These correspondences are indicated by arrows in the figure.

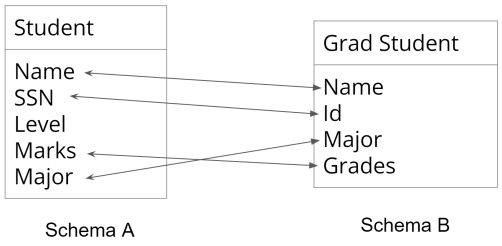



**Figure 1: A simple example of schema matching, where matched attribute pairs are labeled with arrows.**

Recently, a new trend in data analysis services has made schema matching increasingly important. Currently, organizations across diverse fields (including retail, healthcare, and manufacturing) are willing to use data-driven strategies to discover unique strategic insights [25]. To deal with these demands, numerous companies are providing industry-specific analytics solutions aiming to meet customers' domain-specific requirements [25]. For example, Microsoft Azure Industry [2] has developed data analysis frameworks for various well-defined industry-specific schemata (ISS). When customers use Azure Industry with their own data, the service operator will try to match the schema of customers' data with these ISS, enabling more effective analysis using the existing frameworks [25].

Although schema matching has been researched for more than two decades, the advent of new technologies and tools is continuously expanding the possibilities in this field. Traditional approaches to schema matching rely on basic linguistic features, structural information, or a combination of both. Later, machine learning-based methods emerged, offering better performance [20]. Notably, one very recent study has reached state-of-the-art performance on schema matching by fine-tuning a pre-trained BERT model [25].

In recent years, large language models (LLMs) like ChatGPT and GPT-4 [16] have emerged as powerful tools. These LLMs demonstrate remarkable capability in solving a variety of tasks in a zero-shot setting [16]. This prowess of LLMs has inspired many new data-mining studies [19], and some research also leverages LLM to achieve state-of-the-art outcomes in some data-mining tasks like text-to-SQL [17]. However, as far as we are aware, there has not been any research about applying zero-shot LLMs to schema matching.

Our research is motivated by the potential of using large language models (LLMs) such as ChatGPT and GPT-4 in performing schema matching in relational databases. In this paper, we first proposed a simple method employing LLMs for schema matching. To our surprise, this simple method significantly outperformed existing state-of-the-art approaches across several benchmarks. Further, we investigated several well-known prompt engineering techniques

*Both authors contributed equally to this research.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
*EECS584, FA 2023, Ann Arbor, MI*
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

to refine our LLM-based schema-matching approach. Our experiments indicate that while not every prompt engineering technique substantially boosts performance, some prompt engineering methods do improve the model's performance significantly. We report all these experimental findings and discuss them in this paper.

Our contributions can be summarized as follows:

(1) We proposed a plain LLM-based method for schema matching in relational databases. Evaluations on several benchmarks show our method significantly outperforms existing state-of-the-art approaches.
(2) We have conducted experiments to explore the effects of various well-known prompt engineering methods on our LLM-based approach. We found that some prompt engineering strategies would further improve our method. We report the results and insights gained from these experiments.

The remainder of the paper is organized as follows: In section 2, we discuss the problem setting in detail and present a formal definition of the problem. Section 3 presents the plain LLM-based method for schema matching. In Section 4, we describe how we apply various well-known prompt engineering techniques to our method. Section 6 discusses the limitations of our work and suggests potential future directions. Finally, our conclusions are presented in Section 7.

## 2 PROBLEM STATEMENT

Schema matching seeks to determine mappings between schema elements based on their semantic equivalence. A formal definition of schema matching can be described as follows.

Definition 1 (Schema matching). *In the task of schema matching, a source schema $S_s$ is given, which consists of a set of entities $E_s$ (i.e. tables), a set of attributes $A_s$ (i.e. columns), and a set of primary keys and foreign keys relationships $R_s$. For every $a \in A_s$, there is an $e \in E_s$ such that $a \in e$.*

*Similarly, a target schema $S_t$ is also given, which also consists of a set of entities $E_t$, a set of attributes $A_t$, and a set of primary keys and foreign keys relationships $R_t$. For every $a \in A_t$, there is an $e \in E_t$ such that $a \in e$.*

*Schema matching aims to discover entity matches between $S_s$ and $S_t$. An entity match $M$ is a triplet $(e_s, e_t, m)$, where $e_s \in E_s$ and $e_t \in E_t$, and $m$ is a set of attribute correspondences between $e_s$ and $e_t$ in the form of $\{(a_{s,1}, a_{t,1}), (a_{s,2}, a_{t,2}), ..., (a_{s,k}, a_{t,k})\}$. For every $i \in \{1, ..., k\}$, $a_{s,i} \in e_s$ and $a_{t,i} \in e_t$. Moreover, $a_{s,i}$ should be distinct from each other, and similarly, $a_{t,i}$ should also be distinct from each other. For each matched-up attribute pair $(a_{s,i}, a_{t,i})$, $a_{s,i}$ and $a_{t,i}$ semantically refers to the same concept.*

For example, in the schema-matching example illustrated in Figure 1, we can assume that schema A is the source schema $S_s$, and schema B is the destination schema $S_t$. In this case, $E_s$ = {Student} and $A_s$ = {Name, SSN, Level, Marks, Major}. All elements in $A_s$ belong to Student. Also, $E_t$ = {Grad Student} and $A_t$ = {Name, Id, Major, Grades}, and all elements in $A_t$ belongs to Grad Student. There is a single entity match between the two schemas, which is (Student, Grad Student, {(Name, Name), (SSN, Id), (Major, Major), (Marks, Grades))}).

A small group of schema-matching studies also consider the situations where one attribute can map to multiple attributes [18]. For example, if the attributes from the source schema are {First Name, Last Name, Address} and the attributes from the destination schema are {Name, Address}, sometimes one might match the attributes First Name and Last Name from the source schema together to the attribute Name in the destination schema. However, in our research, we only consider the one-to-one schema-matching problem defined in Definition 1. This is due to the predominance of one-to-one schema-matching in existing works. As our research aims to explore the potential of LLMs in schema matching, we specifically test their performance in one-to-one schema matching. This allows us to compare the performance of LLM-based methods to a greater number of baseline methods on more benchmarks. Notably, we did not find any public schema-matching benchmarks that allow mappings of a single attribute to multiple attributes.

## 3 A PLAIN LLM-BASED METHOD FOR SCHEMA-MATCHING

In this section, we describe our plain LLM-based method for schema-matching. This method is a very simple and direct usage of LLM in solving the schema-matching problem. We input the schemata into a LLM, like ChatGPT, and ask it to perform the schema matching. Figure 2 shows an example of this process.

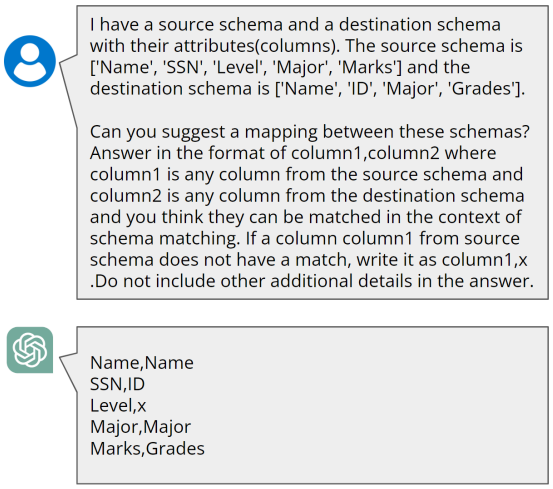



**Figure 2: An example of the plain LLM-based method for schema-matching**

Formally, let $A_s$ be the attributes from the source schema and $A_t$ be the attributes from the target schema. We will present the following prompt to the LLM. Subsequently, the LLM will generate the matching result. Typically, each line of the matching result produced by the LLM comprises a paired set of attributes, separated by a single comma.

> I have a source schema and a destination schema with their attributes(columns). The source schema is $A_s$ and the destination schema is $A_t$.
> Can you suggest a mapping between these schemas? Answer in the format of column1,column2 where column1 is any column from the source schema and column2 is any column from the destination schema and you think they can be matched in the context of schema matching. If a column column1 from source schema does not have a match, write it as column1,x .Do not include other additional details in the answer.

It is important to note that we have expended considerable length of text in describing the output format of the mapping results. These descriptions are essential as they constrain the output of LLMs to a specific format.

We were surprised to discover that such a simple method surpasses state-of-the-art schema matching methods on multiple benchmarks. These evaluation results will be detailed in Section 5.

# 4 ADDING ADDITIONAL PROMPT ENGINEERING TECHNIQUES TO THE LLM-BASED METHOD

The method described in Section 3 presents a very straightforward approach to utilizing LLMs for solving the schema-matching problem. Hoping to improve our basic LLM-based method, we explored various prompt engineering techniques. This section will first briefly introduce several state-of-the-art prompt engineering techniques. Following that, we will thoroughly describe our attempts to integrate these techniques into our LLM-based schema-matching method.

## 4.1 Prompt Engineering

Since the rise of large language models, a newly emerged research area known as prompt engineering has been growing rapidly. In 2021, researchers demonstrated that a well-structured prompt can improve the reasoning ability of LLMs, leading to improved performance on tasks demanding logical thinking [21]. A number of prompt engineering techniques have been developed ever since. In the following sections, we will cover the three major techniques applied in our schema matching solutions.

*4.1.1 In-context Learning.* While using LLMs for schema matching is considered zero-shot, a prompt engineering technique known as in-context learning or few-shot prompting can be applied to further improve the performance of LLMs. Instead of asking the LLM questions directly, in-context learning requires the prompt to include one or more examples first. Studies have shown that in-context learning is one of the emergent abilities of larger models [22]. Compared to fine-tuning LLMs for a specific task, in-context learning allows LLMs to perform temporary learning from prompted examples, and does not carry bias from time to time.

*4.1.2 Chain-of-thoughts (CoT).* In 2022, a team from Google Brain published the chain-of-thought prompting technique [23]. This technique decomposes the initial task into a series of intermediate steps, thereby boosting reasoning performance for complex tasks. Their experiments showed a positive linear relationship between the problem-solving rate and the number of parameters. Notably, applying chain-of-thought on models with more than 100B parameters yields pure gain in the problem-solving rate.

*4.1.3 Self-Refine.* Another simple yet effective technique is known as self-refine [10]. This technique mimics the human behavior of problem-solving. It can iteratively improve LLM's output by feeding the initial response back into the model and requesting feedback. The feedback is then incorporated into subsequent iterations, generating increasingly refined outputs. This technique provides a drastically increased performance improvement at the first feedback-refine iteration, but also experiences diminishing returns with subsequent iterations.

## 4.2 LLM-based Schema Matching with In-context Learning

To apply the in-context learning technique to our Large Language Model (LLM)-based method, we provide the LLM with an example prompt followed by an example response. This approach enables the LLM to learn the task in context before it is asked to solve the actual problem.

While an LLM could be utilized to implement a chat-box assistant, its nature allows it to seamlessly continue any conversation, regardless of whether the content produced earlier in the role of the assistant was generated by the LLM. This property allows us to initially present an example prompt to the LLM, and following this, we demonstrate an example response by adopting the assistant's role, pretending to be the LLM itself and simulating the LLM's response. Subsequently, we present the actual schema-matching problem we wish to address and await the LLM's response. Figure 3 shows an example of this process.

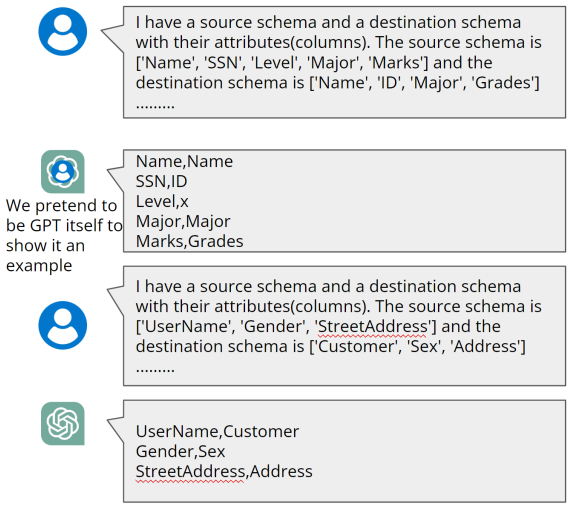



**Figure 3: An example of the in-context learning prompt engineering**

With the help of the OpenAI API [15], we can implement this process with the following code snippet.

```
messages = [
    {"role": "user", "content": example_prmpt},
    {"role": "assistant", "content": example_response},
    {"role": "user", "content": true_prompt},
]
response = client.chat.completions.create(
    model=model,
    messages=messages,
)
return response.choices[0].message.content
```

## 4.3 LLM-based Schema Matching with CoT

The application of CoT [23] to the LLM-based method bears similarities to the use of in-context learning in the method. The key distinction lies in the example responses provided. Unlike in-context learning, where the LLM is given example responses containing only the answers, the application of CoT involves presenting an example response that outlines a step-by-step solution, which demonstrates how to systematically tackle the schema matching problem.

Nevertheless, there are many different methods for systematically solving a schema-matching problem in a step-by-step manner. In this research, we explored two approaches that apply CoT to the LLM-based method. Each approach presents the LLM with a distinct method for solving the schema-matching problem step by step.

*4.3.1 CoT Method 1.* In CoT method 1, We show the LLM an example response that performs matches one by one, providing explanations for each. Figure 4 shows an example of this process.

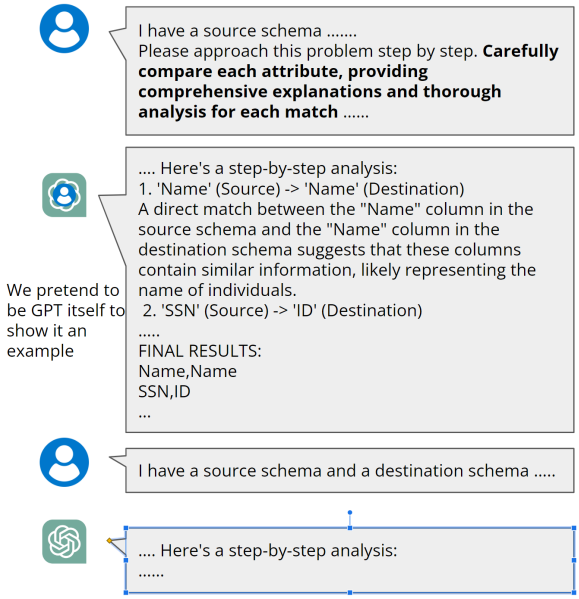



**Figure 4: An example of the CoT method 1 prompt engineering**

*4.3.2 CoT Method 2.* In CoT method 2, we show the LLM an example response that first analyzes the meaning of each attribute from each schema, providing explanations for each. Figure 5 shows an example of this process.

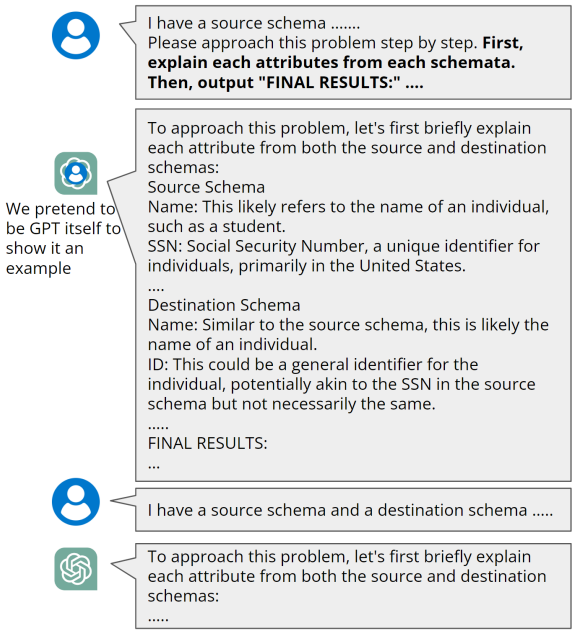



**Figure 5: An example of the CoT method 2 prompt engineering**

## 4.4 LLM-based Schema Matching with Self-Refine

To implement the self-refine technique [10] to the LLM-based schema matching method, we first simply input the schemata into the LLM and ask it to perform the schema matching, just like what we do for the plain LLM-based method. Nonetheless, after we get the initial response from the LLM, we inquire whether the LLM can make any refinements to its original answer. Subsequently, after receiving the LLM's response to this inquiry, we then request its final answer. Figure 6 shows an example of this process.

# 5 EVALUATION

## 5.1 Baselines: Existing State-of-the-art Methods

*5.1.1 Traditional Approaches.* Schema matching has been researched for over two decades. Some early state-of-the-art approaches includes COMA [4], CUPID [11], S-MATCH [6], and Similarity Flooding [12]. These methods leverage linguistic information, or structural information, or both to perform schema matching. The ways they utilize linguistic information are relatively traditional - they use methods like synonym dictionaries, edit distances, n-grams, WordNet [13], etc [4, 6, 11].

While these approaches may be considered traditional in the field of schema matching, they continue to hold significant value for studying this problem. They are frequently employed as baseline models in recent research publications, serving as essential benchmarks for result comparison. As an illustration, the study conducted

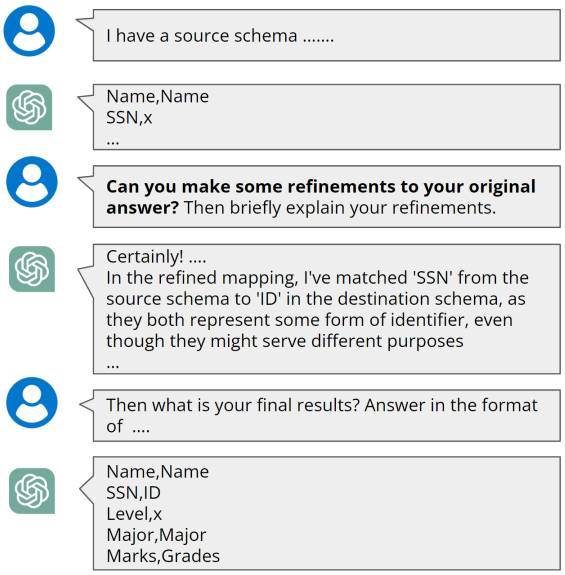



**Figure 6: An example of the Self-Refine prompt engineering**

by Zhang et al. in 2023 ([25]) incorporated COMA, CUPID, S-Match, and Similarity Flooding as reference methods to evaluate their own LSM (Learned Schema Matcher) approach. This practice of comparing new methods against established ones helps researchers assess the advancements and effectiveness of novel techniques in the ever-evolving landscape of schema matching research.

Therefore, we also incorporate COMA, Cupid and Similarity Flooding as three representatives of traditional approaches in our research for evaluation and result comparison. A brief introduction of these methods is presented below:

- ***COMA***: COMA (Correspondence Management) utilizes a diverse set of matchers, encompassing various linguistic similarity metrics like affix, n-gram, Soundex, and edit distance, to determine correspondences between attributes in different schemas. COMA has high flexibility in choosing from a range of aggregation functions, including min, max, and average, to effectively combine these similarities. This adaptability makes COMA a versatile tool for schema alignment and data integration tasks, enabling it to cater to specific matching requirements in diverse information systems and data management contexts [4]. In our research, we use a Python wrapper around COMA 3.0 Community Edition.
- ***CUPID***: CUPID addresses the challenge of aligning XML and relational schemata by employing a unique approach that amalgamates linguistic and structural data. It tackles the schema matching problem by harnessing both linguistic insights, facilitated through a synonym dictionary, and structural information, and then computes a weighted sum of the resulting similarities [11]. In our research, we use a Python implementation of the same paper [11].
- ***Similarity Flooding***: Similarity Flooding (SF) employs a unique approach to establish correspondences between elements in source and target schemata. SF begins by constructing a pairwise connectivity graph, which is based on the structural relationships within the source and target schemata. In this graph, each element pair serves as a node, and nodes (x1, y1) and (x2, y2) are connected if x1, x2, and y1, y2 are neighbors in the original schema graphs, respectively. SF then initializes similarity scores for these element pairs and iteratively propagates and refines these scores by traversing the edges in the connectivity graph. This process allows Similarity Flooding to discover meaningful correspondences by considering both structural context and similarity metrics [12]. In our research, we use a Python implementation of the same paper [12].

*5.1.2 ML-based approacheses.* One early work in 2000s leveraged machine learning for schema matching using only schema information, employing traditional machine learning methods such as TF-IDF encodings and Naive Bayes based [5]. In recent years, as machine learning has rapidly evolved, an increasing number of researchers are using machine learning approaches for schema matching. For instance, one study proposes MLM, a machine learning-based method that uses both the schema information and the data information to do schema matching [20]. Nozaki et al. explored the use of word vectors in schema matching [14]. Another study investigated the application of neural embeddings for the same purpose [8].

As language models have become popular in recent years, one very recent work utilized a pre-trained language model for schema matching [25]. They introduced the Learned Schema Matcher (LSM). LSM presents an innovative linguistic schema matching approach, utilizing semi-supervised and active learning techniques to iteratively and interactively align source and target schemas. LSM begins by generating candidate pairs between attribute sets, which are iteratively labeled by the user. During featurization, a modular pipeline converts candidate pairs into numerical vectors, incorporating various similarity metrics, including a fine-tuned BERT [3] featurizer. An ML model is trained on partially labeled data to predict correct matches and generate top-k suggestions for unmapped source attributes. The user reviews and refines suggestions, and informative attributes are selected for labeling to reduce human effort. LSM's iterative process continually enhances schema matching accuracy while involving user feedback, making it a powerful tool for schema alignment and integration tasks [25].

## 5.2 Evaluation Datasets and Benchmarks

For evaluation, we implemented our zero-shot LLM methods with two LLMs - the ChatGPT (GPT3.5) and GPT4 [16].

The RDA Crosswalk benchmark [24] is an ideal benchmark for our evaluation for various reasons. We primarily utilize the RDA Crosswalk benchmark [24] to assess the performance of our proposed LLM-based methods. We use this benchmark to evaluate the performance of all the LLM-based methods we proposed and compare their performance against each other. We have also implemented the three traditional schema matching approaches - COMA [4], CUPID [11], and Similarity Flooding [12] - on this

dataset and compare our methods against these three traditional methods.

We also intended to juxtapose our methods with the latest state-of-the-art approach, LSM [24], utilizing the RDA Crosswalk benchmark. However, LSM, being a BERT-based method, has not made its model or training data public. Therefore, we cannot implement it on the RDA Crosswalk dataset. The authors of LSM evaluated their fine-tuned BERT model on real-world schemata from Microsoft customers in retail, and additionally, three small publicly available schemata [24]. Lacking access to the Microsoft customers' schemata, we can only compare the performance of our methods against LSM's results on these three small publicly accessible schemata.

*5.2.1 Evaluation on the RDA Crosswalk Benchmark.* Firstly, we benchmark our approach against conventional schema matching techniques using the RDA Crosswalk dataset from OAEI (Ontology Alignment Evaluation Initiative) [24]. The RDA Crosswalks Data Schema Matching dataset is developed by the RDA Research Metadata Schemas Working Group, serves as a benchmark for assessing systems' proficiency in schema metadata matching tasks, particularly aligning 15 research data schemas with Schema.org [24]. This dataset follows the schedule of OAEI 2023 and includes a diverse range of source schemas, covering both discipline-agnostic (e.g., Dublin Core, DCAT) and discipline-specific (e.g., ISO19115-1, DATS) metadata schemas, as well as CodeMeta for software. Its primary purpose is to aid data repositories in crosswalk development and provide insights into semantic interoperability among various schemas [24]. Therefore, it is a valuable resource and a perfect candidate for our evaluation.

In addition to the RDA Crosswalks Dataset being large and comprehensive, it offers another advantage: it was published in December 2021[24], beyond the scope of the training data for GPT-3.5 and GPT-4 [16]. Consequently, GPT-3.5 and GPT-4 do not possess prior knowledge about this dataset.

Given that this dataset exclusively comprises metadata schemas without any actual data content, our evaluation exclusively relies on metadata information. And we used 12 schemas out of 15 in our evaluation, including DCAT-AP, DCATv3, Datacite, Dataverse, DATS, DC and DC Term, BioSchema, B2FIND, DDI, ECRIN Clinical Research Metadata Schema, SPASE, CodeMeta. Please refer to OAEI [24] for more information about each data schema.

In this particular phase of our evaluation, we employed Schema.org's property schema as the source schema, and the aforementioned 12 schema served as the target schema for the matching process. Let's consider an example where we chose the COMA matching strategy, and our goal was to match the source schema (Schema.org's property schema) to DCAT-AP. In this scenario, we leveraged the COMA implementation to facilitate the matching process and generated a comprehensive set of metrics as output based on the ground truth, which is provided in the RDA Crosswalk dataset.

*5.2.2 Zero-shot LLM VS fine-tuned BERT.* In the second part of our evaluation, we compared our Zero-shot LLM (GPT) approach against the fine-tuned BERT model mentioned before [25]. As we mentioned above, we are only able to compare our methods with LSM on three small public datasets due to the limited publicity and the reproducibility of LSM:

- RDB-Star: it is a collection of data designed for testing and evaluating the RDB-Star extension and it is used in [11].
- IPFQR: it comprises a comprehensive collection of healthcare data, encompassing information related to the quality of care delivered by inpatient psychiatric facilities participating in the Inpatient Psychiatric Facility Quality Reporting Program (IPFQR) [1].
- MovieLens-IMDB: Mapping between the Movie-Lens schema [7] and the IMDB schema.

In their model, they offer a set of matching suggestions and assess the accuracy of these suggestions using the top-k accuracy metric on the unlabeled portion of the source schema [25]. Top-k accuracy represents the likelihood that the correct match is among the top-k suggestions. For consistency in our analysis, we also employ top-k accuracy as our evaluation metric for this phase. However, since we have configured our GPT model to provide only one or zero suggestions for each column in the source schema through our prompting approach, we specifically utilize top-1 accuracy as our performance evaluation metric.

## 5.3 Experiment Results

*5.3.1 Evaluation on the RDA Crosswalk Benchmark.* In Table 1, 2, and 3, we have provided F1-scores to evaluate the quality of matches between the source schema (Schema.org's property schema) and each of the target schema on the RDA Crosswalk benchmark. Table 1 provides the F1-scores of the three traditional methods, Table 2 provides the F1-scores of the LLM-based methods that use ChatGPT(GPT3.5), and the Table 3 provides the F1-scores of the LLM-based methods that use ChatGPT(GPT4).

The choice of F1-score as our primary metric is grounded in the importance of considering both precision and recall when assessing the performance of LLM. This is particularly significant because LLMs have the capability to generate matches that may not exist in reality (this phenomenon is also called hallucinations of LLMs [9]), which can impact the overall quality of schema alignment. By using the F1-score, we provide a comprehensive evaluation that considers both the accuracy of matches (precision) and the extent to which all relevant matches are identified (recall). This balanced approach allows us to better understand the performance of LLMs in the schema alignment task, particularly in scenarios where erroneous or fictitious matches may be produced.

From the tables, we can see that all of LLM-based methods outperform these traditional approaches significantly. Even the most simple method - the plain LLM-based method with GPT3.5 as the model - has reached an average F1-score of 0.758. This score is markedly higher than the average F1-scores of 0.52, 0.47, and 0.56 achieved by the three traditional methods.

Overall, GPT4-base methods performs significantly better than GPT3.5-based methods. This is expected as GPT4 is a more advanced LLM [16].

Another intriguing aspect of our experiments is the discovery that not every prompt engineering technique substantially enhances the performance of LLM-based methods. The Self-Refine technique even slightly diminishes the performance of the GPT-4-based method. However, in-context learning and the second CoT

| Schema Name | COMA | Cupid | Similarity Flooding |
|---|---|---|---|
| DCAT-AP | 0.50 | 0.21 | 0.67 |
| DCATv3 | 0.54 | 0.26 | 0.56 |
| Datacite | 0.46 | 0.46 | 0.42 |
| Dataverse | 0.42 | 0.34 | 0.42 |
| DATS | 0.69 | 0.48 | 0.69 |
| DC | 0.45 | 0.41 | 0.43 |
| BioSchema | 0.70 | 0.78 | 0.98 |
| B2FIND | 0.55 | 0.63 | 0.60 |
| DDI | 0.48 | 0.46 | 0.47 |
| ECRIN | 0.34 | 0.35 | 0.24 |
| SPASE | 0.47 | 0.25 | 0.32 |
| CodeMeta | 0.69 | 0.95 | 0.95 |
| Average F1 score | 0.52 | 0.47 | 0.56 |

**Table 1: F1-Scores for some traditional methods on RDA Crosswalk benchmark**

| Schema Name | GPT3.5 Plain | GPT3.5 in-context | GPT3.5 CoT1 | GPT3.5 CoT2 | GPT3.5 Self-Refine |
|---|---|---|---|---|---|
| DCAT-AP | 0.69 | 0.82 | 0.81 | 0.83 | 0.75 |
| DCATv3 | 0.85 | 0.87 | 0.78 | 0.87 | 0.82 |
| Datacite | 0.77 | 0.70 | 0.67 | 0.73 | 0.79 |
| Dataverse | 0.78 | 0.85 | 0.75 | 0.83 | 0.83 |
| DATS | 0.67 | 0.73 | 0.67 | 0.73 | 0.69 |
| DC | 0.66 | 0.71 | 0.69 | 0.70 | 0.66 |
| BioSchema | 0.98 | 0.98 | 0.94 | 0.98 | 0.94 |
| B2FIND | 0.86 | 0.89 | 0.89 | 0.98 | 0.86 |
| DDI | 0.54 | 0.87 | 0.67 | 0.70 | 0.75 |
| ECRIN | 0.49 | 0.60 | 0.72 | 0.54 | 0.52 |
| SPASE | 0.84 | 0.83 | 0.72 | 0.96 | 0.79 |
| CodeMeta | 0.98 | 0.98 | 0.98 | 1.0 | 0.83 |
| Average F1 score | 0.758 | **0.819** | 0.772 | 0.820 | 0.769 |

**Table 2: F1-Scores for the LLM-based methods that uses ChatGPT(GPT3.5) on RDA Crosswalk benchmark**

| Schema Name | GPT4 Plain | GPT4 in-context | GPT4 CoT1 | GPT4 CoT2 | GPT4 Self-Refine |
|---|---|---|---|---|---|
| DCAT-AP | 0.84 | 0.92 | 0.90 | 0.92 | 0.83 |
| DCATv3 | 0.94 | 0.89 | 0.92 | 0.94 | 0.89 |
| Datacite | 0.78 | 0.89 | 0.74 | 0.76 | 0.77 |
| Dataverse | 0.85 | 0.86 | 0.84 | 0.86 | 0.69 |
| DATS | 0.76 | 0.77 | 0.73 | 0.75 | 0.64 |
| DC | 0.72 | 0.76 | 0.70 | 0.71 | 0.78 |
| BioSchema | 1.00 | 0.98 | 0.98 | 1.00 | 0.87 |
| B2FIND | 0.79 | 0.93 | 0.95 | 0.95 | 0.84 |
| DDI | 0.88 | 0.88 | 0.80 | 0.77 | 0.77 |
| ECRIN | 0.65 | 0.64 | 0.65 | 0.67 | 0.54 |
| SPASE | 0.87 | 0.91 | 0.82 | 0.87 | 0.58 |
| CodeMeta | 1.00 | 1.00 | 1.00 | 1.00 | 0.91 |
| Average F1 score | 0.835 | **0.868** | 0.835 | 0.851 | 0.782 |

**Table 3: F1-Scores for the LLM-based methods that uses GPT4 on RDA Crosswalk benchmark**

method significantly improve LLM-based performance, with in-context learning being particularly effective. Remarkably, integrating in-context learning as a prompt engineering technique with GPT-4 has yielded the highest performance among all methods, achieving an average F1-score of 0.868.

| | The Best Baseline Reported in the LSM Paper [25] | LSM [25] | GPT3.5 Plain | GPT4 in-context |
|---|---|---|---|---|
| RDB-Star | 0.95 | 0.98 | 0.98 | 1.00 |
| IPFQR | 1.00 | 1.00 | 1.00 | 1.00 |
| MovieLens-IMDB | 0.53 | 0.65 | 0.79 | 1.00 |

**Table 4: Top-1 Accuracy of LSM and GPT approaches on the public schemata**

*5.3.2 Zero-shot LLM VS fine-tuned BERT.* The paper of LSM uses raw accuracy as their evaluation metric, which calculates the proportion of the attributes from the source schema that are correctly matched [25]. To align our evaluations with theirs, we adopted this accuracy metric for assessing our LLM-based methods across the RDB-Star, IPFQR, and MovieLens-IMDB schemata. Consequently, hallucinated matches are no longer penalized in this metric.

Since we have already compared different versions of LLM-based results on a larger, more comprehensive, and more recent benchmark - the RDA Crosswalk Benchmark. For these three small benchmarks, we have only evaluated the performance of the plain LLM-based method utilizing GPT3.5, as well as that of the LLM-based method incorporating in-context learning prompt engineering techniques with GPT4. We uses these evaluation results to compare with the results reported in the paper of LSM [25].

Although the paper of LSM [25]uses a variety of baseline methods, which includes COMA [4], CUPID [11], S-MATCH [6], Similarity Flooding [12], LSD [5], and MLM [20], they have not reported the performance of all of these methods. Instead, they have only reported the best baseline results among all of their baseline results as a comparison to their results.

Table 4 shows the best baseline used in the LSM paper the performance of LSM as reported in the LSM paper, the performance of plain the plain LLM-based method utilizing GPT3.5, and the performance of the LLM-based method incorporating in-context learning prompt engineering techniques with GPT4. From these results, we can see that our methods significantly outperform LSM.

## 6 DISCUSSION, LIMITATION, AND FUTURE WORKS

- **The mystery of the effect of various prompt engineering techniques:** Our experiments reveal that not all state-of-the-art prompt engineering techniques enhance the performance of our LLM-based model. This is an intriguing phenomenon.
  We notice that the Chain of Thought (CoT) method may not always significantly improve our approach. We hypothesized that this could be due to the schema matching task's reliance more on the LLM model's knowledge about schemas rather than on the ability to conduct step-by-step logical or mathematical analysis. Consequently, CoT's strength in guiding the LLM through sequential thinking does not substantially boost the model's performance. Moreover, as the model is required to generate more content, other issues, such as the hallucination problems, may become more severe.
  Additionally, we observed that the self-refine method has a minimal impact on our model's performance. This might be because self-refinement is typically employed in artistic generative tasks, where it allows the LLMs to progressively enhance outputs like images, poems, etc. By continuously prompting the LLM to refine its creation, it can gradually perfect its output. However, in schema matching, a problem-solving task, the model must provide a single, precise answer. Thus, allowing the LLM to refine its work gradually may not be beneficial. Similarly to the CoT, producing more output could also lead to more pronounced hallucination issues.
  However, these thoughts are purely hypothetical. Further experiments could be conducted to explore how different prompt engineering techniques influence the performance of LLMs in various tasks. Nevertheless, due to the constrained timeframe of this research project, and considering that the fact that not all prompt engineering methods would enhance the LLM-based models is beyond our initial expectations and plans, we are unable to dedicate additional time to investigating this problem.
- **Leveraging data for enhanced schema matching:** In our schema matching process, we've only relied on schema information due to data absence in the RDA crosswalk dataset. However, we see the potential to significantly boost Language Model-based Schema Matching by incorporating actual data alongside schema. Drawing inspiration from traditional methods like COMA, which utilize data instances for schema matching, could provide valuable insights and improve overall accuracy.
- **End-to-end schema matching pipeline:** Since the output of LLMs is non-deterministic, it can be challenging to parse the result and automate the end-to-end schema matching process. This is especially challenging when prompt techniques like chain-of-thought are applied. In the example we showed earlier, LLM's response consists of a mix of raw results and attribute explanations. Currently, manual intervention is required in some cases to parse the LLM output. This poses a challenge when incorporating our schema-matching solution into an automated workflow, e.g. a build-in service for data lakes. In the future, we could consider prompt engineering techniques to further constrain the behavior of LLMs, or build a more intelligent parser to deal with non-deterministic result formats. Another potential problem to consider is the automatic selection of source and destination schema. While it falls outside the scope of this paper, having automatic discovery of input schemas will also benefit the end-to-end process mentioned above.
- **Various metrics for evaluation:** While the F1-score is a common metric for schema matching, it assumes equal importance of precision and recall, which may not always hold true in different applications. To ensure a more accurate evaluation of schema matching, it is essential to consider alternative metrics that can be tailored to specific use cases,

accounting for variations in the significance or cost associated with precision and recall.

## 7 CONCLUSION

Over the past few years, large language models (LLMs) including ChatGPT have undergone rapid development and enabled new directions to the classic schema matching problem. Our approach demonstrates the potential of LLMs which outperform traditional methods including COMA, CUPID, LSM, and similarity flooding in both F1-score and top-1 accuracy metrics.

Furthermore, we implemented three major prompting techniques, including in-context learning, chain-of-thought, and self-refine. Among these three, in-context learning and chain-of-thought appear to be the most effective techniques. We also implemented two versions of chain-of-thought to compare and reason about their respective outcomes.

To the best of our knowledge, we are the first to incorporate zero-shot LLMs into solving the classic schema matching problem. Our baseline implementation consistently outperforms traditional methods across all scenarios, while leveraging prompt engineering techniques further increases our solution's accuracy.

## REFERENCES


[1] [n. d.]. IPFQR Data. https://data.cms.gov/provider-data/archived-data/hospitals.
[2] 2022. Azure Industry Solutions. https://azure.microsoft.com/enus/industries/.
[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).
[4] Hong-Hai Do and Erhard Rahm. 2002. COMA—a system for flexible combination of schema matching approaches. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*. Elsevier, 610–621.
[5] AnHai Doan, Pedro M Domingos, and Alon Y Levy. 2000. Learning Source Description for Data Integration.. In *WebDB (informal proceedings)*. 81–86.
[6] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. 2004. S-Match: an algorithm and an implementation of semantic matching. In *The Semantic Web: Research and Applications: First European Semantic Web Symposium, ESWS 2004 Heraklion, Crete, Greece, May 10-12, 2004. Proceedings 1*. Springer, 61–75.
[7] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 5, 4 (2015), 1–19.
[8] Benjamin Hättasch, Michael Truong-Ngoc, Andreas Schmidt, and Carsten Binnig. 2022. It's AI Match: A Two-Step Approach for Schema Matching Using Embeddings. *arXiv preprint arXiv:2203.04366* (2022).
[9] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. 2023. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232* (2023).
[10] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-Refine: Iterative Refinement with Self-Feedback. arXiv:2303.17651 [cs.CL]
[11] Jayant Madhavan, Philip A Bernstein, and Erhard Rahm. 2001. Generic schema matching with cupid. In *vldb*, Vol. 1. 49–58.
[12] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. 2002. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings 18th international conference on data engineering*. IEEE, 117–128.
[13] George A Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (1995), 39–41.
[14] Kenji Nozaki, Teruhisa Hochin, and Hiroki Nomiya. 2019. Semantic schema matching for string attribute with word vectors. In *2019 6th International Conference on Computational Science/Intelligence and Applied Informatics (CSII)*. IEEE, 25–30.
[15] OpenAI. 2023. OpenAI API. https://beta.openai.com/. Accessed: [Your Access Date Here].
[16] R OpenAI. 2023. GPT-4 technical report. *arXiv* (2023), 2303–08774.
[17] Mohammadreza Pourreza and Davood Rafiei. 2023. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *arXiv preprint arXiv:2304.11015* (2023).
[18] Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *the VLDB Journal* 10 (2001), 334–350.
[19] Mohammed Saeed, Nicola De Cao, and Paolo Papotti. 2023. Querying Large Language Models with SQL. *arXiv preprint arXiv:2304.00472* (2023).
[20] Tanvi Sahay, Ankita Mehta, and Shruti Jadon. 2020. Schema matching using machine learning. In *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*. IEEE, 359–366.
[21] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask Prompted Training Enables Zero-Shot Task Generalization. arXiv:2110.08207 [cs.LG]
[22] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent Abilities of Large Language Models. arXiv:2206.07682 [cs.CL]
[23] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL]
[24] M. Wu, P. Hagan, B. Cecconi, S. M. Richard, C. Verhey, and RDA Research Metadata Schemas WG. 2023. *A Collection of Crosswalks from Fifteen Research Data Schemas to Schema.org*. Research Data Alliance. https://doi.org/10.15497/RDA00069
[25] 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 *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1558–1571.