NORTHWESTERN
UNIVERSITY

Computer Science Department

Technical Report
Number: NU-CS-2023-11

June, 2023

Hybrid Embeddings for Relational Schema Matching

Di Mei

Abstract

Relational schema matching is a task of finding correspondences between elements from a source and a target tabular schema. It helps data scientists integrate data from different sources as well as explore inter-schema relationships, and it has been widely researched for more than twenty years. Existing schema matching methods are either schema-based, focusing on element names and relational structures, or instance-oriented, highlighting the distribution of data values. Their generalization capability is limited due to their naive assumptions about the linguistic similarity and instance type. To resolve this issue, we propose the hybrid embedding, an innovative schema matching strategy integrating both methodologies, and apply a machine learning classifier to determine a match. Through experiments on both synthetic and real-world datasets, we show that our approach outperforms existing methods and it is robust to schema variations and limited instance samples.

Keywords

Schema Matching, Machine Learning, Contextual Embedding

NORTHWESTERN UNIVERSITY

Hybrid Embeddings for Relational Schema Matching

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

MASTER OF SCIENCE

Field of Computer Science

By

Di Mei

EVANSTON, ILLINOIS

May 2023

ABSTRACT

Relational schema matching is a task of finding correspondences between elements from a source and a target tabular schema. It helps data scientists integrate data from different sources as well as explore inter-schema relationships, and it has been widely researched for more than twenty years. Existing schema matching methods are either schema-based, focusing on element names and relational structures, or instance-oriented, highlighting the distribution of data values. Their generalization capability is limited due to their naive assumptions about the linguistic similarity and instance type. To resolve this issue, we propose the hybrid embedding, an innovative schema matching strategy integrating both methodologies, and apply a machine learning classifier to determine a match. Through experiments on both synthetic and real-world datasets, we show that our approach outperforms existing methods and it is robust to schema variations and limited instance samples.

ACKNOWLEDGEMENTS

I would like to give my sincerest gratitude to my advisor and committee member, Professor Han Liu and Professor Zach Wood-Doughty. Their guidance and insights are truly valuable for my research progress and success. I am also grateful to the members of this project, Yu Zhang and Robin Luo, whose support and ideas are indispensable for the empirical study of my research. At last, it is my deepest pleasure to conduct a meaningful research at Northwestern University with many talented people and great facilities.

Glossary

BERT	Bidirectional Encoder Representations from Transformers.
BiLSTM	Bidirectional Long Short Term Memory network.
BLEU	Bilingual Evaluation Understudy.
DB	Distribution-based model.
EMD	Earth Mover's Distance.
HDX	The Humanitarian Data Exchange.
JL	Jaccard-Levenshtein model.
lcs	longest common substring.
lev	Levenshtein distance.
LLM	Large Language Model.
lsim	linguistic similarity.
SF	Similarity Flooding.
ssim	structure similarity.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

At the age of data explosion, data-driven techniques are essential for many industries to learn about market demands and build artificial intelligence applications automating their conventional businesses. The amount of information collected has never been greater[1] [1], emphasizing the massive effort spent on analytics of data with varying scales and associated fields. An unavoidable task for most data scientists is to aggregate disparate data sources and explore potential correspondences between datasets. For instance, a financial analyst wants to conduct an analysis on the annual financial condition of all US companies, and this requires to aggregate financial attributes from the 10-K form of each company. These forms are usually not standardized in different databases and it is laborious to identify attributes, though linguistically different, referring to the same concept (e.g. *Earnings per Share (Diluted)* vs. *Dilute Earns*). Schema matching is an approach for discovering matching pairs of elements or columns from a source and a target schema, which are spreadsheets or databses with a structured format (relational schema). In addition to its original use, schema matching helps rank inter-dataset relationships and map customers' data to the known industry-specific schema [2] for any further standard analysis.

Existing schema matching methods are either schema-based or instance-based. Schema-based algorithms apply schema-related information like element's linguistic similarity and relational structures to measure the relatedness of any pair of elements, while instance-based methods concern about the distribution of data instances when computing the element similarity. Either of them is limited due to their naive assumptions. Schema-based methods are highly dependent on the lin-

[1]https://www.statista.com/statistics/871513/worldwide-data-created/

guistic similarity, which is the initial stage used to infer structure similarity between schemas. They may also rely on manual descriptions of elements, and this makes the task resolution not fully automatic. Instance-based approaches have poor adaptability to instances with different types, and they assume element characteristics are purely identified by their value distribution. Thus, we propose a novel machine learning-based approach integrating both schema-based and instance-based features. This model presents a better schema matching performance on both synthetic and real-world datasets than most existing schema matching techniques. It is also shown to be robust when elements are highly varying and their instance samples are limited.

The rest of the paper is structured as follows: Chapter 2 formally defines the problem of schema matching and then discusses existing methods and our proposed model. There are two areas of existing approaches: schema-based and instance-based information of any tabular schema. Our method takes advantages of both features and has two parts, the hybrid embedding and a classifier. Chapter 3 introduces experimental datasets as well as settings for the training and inference phases of our classifier. Inference results and their performance comparison with other baseline methods are also presented in this chapter. Chapter 4 summarizes the paper and mentions potential limitations of our approach with some future insights.

CHAPTER 2
METHODOLOGY

2.1 Problem Statement

Our discussion about schema matching is under the assumption of any relational schema, which involves a set of tables and elements whose relations are organized in a structured fashion. Every schema consists of multiple entities (also called rows) or elements (also called columns or attributes), and some examples are Excel spreadsheets or tables in a MySQL database. Assume we have two schemas, a source schema S_{src} and a target schema S_{tar}, and each of them has its set of schema elements, E_{src} and E_{tar}. Schema matching can be defined as a problem building the function $f(E_{src} \rightarrow E_{tar})$, which maps some schema elements in E_{src} to their equivalent existences in E_{tar}, though their names and instances might be different. We transform this function into $f(E_{src}, E_{tar} \rightarrow \{0, 1\})$, where given a pair of elements from different schemas, compute whether they can match to each other or not. A 1 means there is a match within the pair while a 0 reflects no correspondence exists here. The core idea is to calculate a similarity between every candidate pair of elements, and this metric instructs us to select matched elements. In addition, we only consider one-to-one schema matching. Every set of matched elements is unique and they can only be matched to each other.

2.2 Schema-based Matching

As mentioned in Chapter 1, schema matching can be categorized into two aspects: schema-based and instance-based matching. A schema-based matching employs schema-related information, which includes element names, description, relationships, etc, to find out matched pairs within two different schemas. One early attempt of this methodology is Cupid [3], and it involves a linguistic matching towards element names as well as a structure matching exploring the hierarchical relations among different elements. With the reference to a thesaurus, the linguistic matching first performs tokenization upon the element names and then categorizes them based on their data types and contents. For schema elements with similar categories, Cupid computes their linguistic similarity coefficients (*lsim*) via evaluating their sub-strings' synonymy and hypernymy relationships.

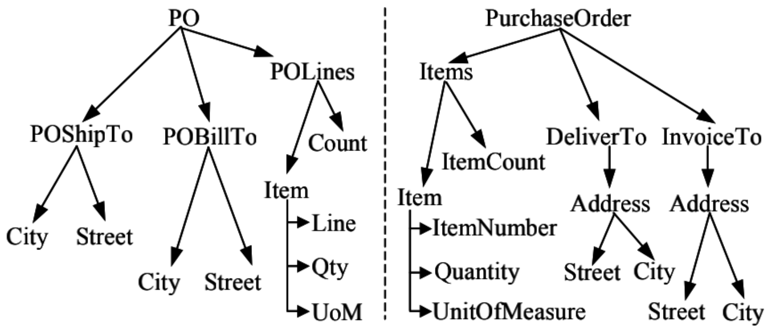

Figure 2.1: **Sample tree-like graphs of two schemas** [3]

The structure matching transforms the schema relations into tree-like graphs (see Figure 2.1). Every node refers to a schema element and every directional edge represents a containment relation. A containment means one element consists of a single or multiple sub-elements. For instance, an *Item* element has several attributes to describe it, and they are *ItemNumber*, *Quantity* and *UnitOfMeasure*. In Cupid, a structure similarity (*ssim*) is computed according to an important intuition: two elements in different schemas are structurally similar if their vicinities (ancestors or siblings) or leaf sets are highly similar [3]. For instance, elements *POLines* and *Items* in Figure 2.1, though linguistically different, are structurally similar since they both have an *Item* set with linguistically similar leaf elements. The final similarity of every element pair is a weighted sum of *lsim* and *ssim*.

Similarity flooding [4] is another schema-based matching technique applying the graph representation of schema relations. It first translates two schemas into directed graphs, where nodes and edges represent elements and relations respectively. A string matcher is used to find some initial mappings of elements based on their prefix and suffix overlaps. Starting from these matches, the similarity flooding algorithm searches for new matched elements in a propagation manner: whenever any two elements in different schemas are found to be similar, the similarity of their adjacent elements increases. This intuition is similar to that of Cupid and the propagation process continues until a fixed point is reached (an equilibrium state). Both Cupid and similarity flooding are highly dependent on the linguistic similarity of elements. For elements with the exactly same semantic representation but different corresponding instances, a high linguistic similarity can mislead the algorithm to compute the structure similarity or propagate matched pairs. In addition, under the scenario of relatively simple schemas, the intuitions about structure similarity become less robust. If their graph representations only have one root and several leaves, all elements are close to each other and we can only infer matched pairs based on their linguistic similarities.

Unlike Cupid or similarity flooding, COMA [5] proposes a schema matching system to combine multiple schema matchers in a flexible fashion when evaluating the similarity of elements under different schemas. Every schema is represented by a rooted directed acyclic graph and every element is shown as its node path from the root. There are multiple schema matchers focusing on different schema information and previous matching results, while one of them is a user-feedback matcher that is unaffected by other matchers and serves to improve the schema matching quality. Different schema matchers offer different computed similarities, and COMA provides different strategies to aggregate them (e.g. average, max, etc) as well as select matched pairs (e.g. above-threshold, top-K, etc). Its experimental results show that single schema matchers may be imprecise, but their combination can effectively improve the matching performance.

With the advancement of various deep learning techniques, the task of natural language understanding can be effectively captured by contextualized word embeddings [6]. To make use of this strength, some schema matchers first compute the embeddings of schema element names as well as their textual descriptions and then decide whether these two embeddings are related in a latent manner or not. For example, SMAT [7] applies a Bidirectional LSTM (BiLSTM) to extract latent features from two element names with their manual descriptions and then uses multiple softmax operations to check whether these two elements are matched or not. Zhang et al. [2] proposes a BERT-based approach to compute the similarity score of two schema elements, which in turn helps determine a match score through a linear classifier. Figure 2.2 shows the BERT embedding process of two schema elements, where a_s and a_t refer to the elements in the source and target schema respectively. Their names are concatenated with their descriptions, and two name-description pairs are also concatenated with a [SEP] notation. Such a format is similar to that of the next-sentence prediction (NSP) task in the pre-training phase of BERT [8]. The left-most embedding is input to a linear layer to obtain a numerical vector indicating a similarity score.

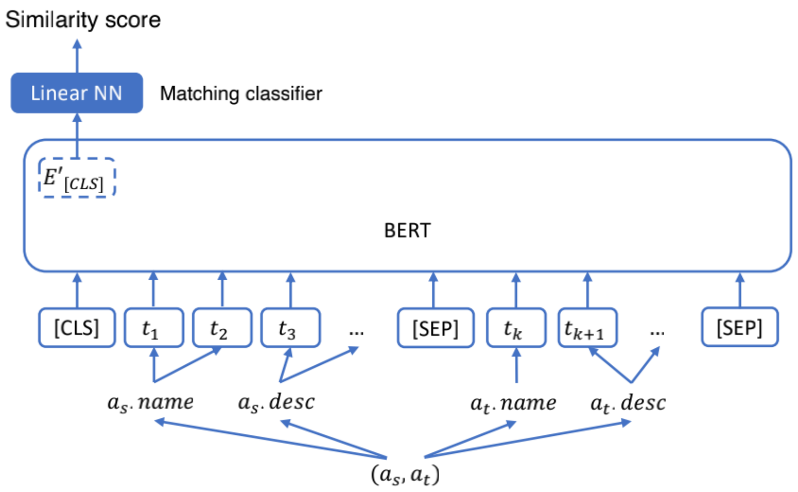

Figure 2.2: **BERT embeddings for two schema elements** [2]

Schema embedding resolves the challenge when elements are different but semantically equivalent. It is also robust to the situation that the target schema has much more number of entities and elements than the source schema [2]. However, these techniques are highly dependent upon the manual element description, which is labor-intensive and not fully automated. In addition, the model proposed and datasets experimented in [2] are still not released to the public.

2.3 Instance-based Matching

An instance-based matching is data-oriented, focusing on utilizing statistical measures to explore relationships between values of different elements. Instance-based schema matchers either apply statistics from data instances to annotate the schema [9], or directly find correlated schema elements [3]. A distribution-based schema matcher uses the Earth Mover's Distance (EMD), which indicates the minimum amount of work to convert one set of values to another, to evaluate the similarity between different elements [10]. To be specific, the work is related to the rank of values in the sorted order of the union of two instance sets. The first step is to form distribution clusters of elements based on pair-wise EMDs. Then, we decompose every cluster into individual matched pairs via the intersection EMD, a defined metric with two intuitions. The first one is schema elements with large intersection of values are highly associated and should belong to the same element. For elements with few or no intersected values, if their values are both largely overlapped with a third element, they are supposed to be matched.

Another instance-based matching technique is mentioned in Koutras et al. [1]. It directly employs the Jaccard similarity to measure the relatedness of any element pair, where any two values are identical if their Levenshtein distance is below a given threshold. Compared to the schema-based matching, an instance-based matching also displays promising results [1][10]. However, it is built upon a vital assumption that elements related to each other have some values in common.

It is always possible that without any common value, two elements can still refer to the same thing. This assumption may be valid for categorical elements, but it can be substantially less robust for numerical elements.

2.4 Schema-Instance Matching

Considering the limitations of each schema matching methodology, we propose a novel schema matching technique that is based on both schema-based and instance-based information from any pair of schemas. To be specific, for every pair of schema elements, their schema-based and instance-based features are elegantly combined via a hybrid embedding, and then a classifier takes this hybrid embedding to predict the extent whether these two elements are matched or not. Though our approach is similar to the pipelines in [2][7], our method is fully automatic and does not require any manual annotation of the schema. Our extracted schema-based features are independent of relational structures [3][4][5], so it avoids the misleading similarities brought by simple schemas. Compared to instance-based matchers purely focusing on numerical statistics [1][10], our designed embedding involves semantic information of string-type instances obtained by BERT-based encoders. The integration of schema-based information and data instances is supposed to present a more robust performance upon schema matching than matching techniques with a single focus.

2.4.1 Hybrid Embedding

Every element e_i from either the source or the target element set (E_{src} and E_{tar}) is transformed into a hybrid embedding $\mathbf{e_i}$, which consists of a semantic embedding $\mathbf{s_i}$ (schema-based information) and an instance embedding $\mathbf{v_i}$ (instance-based information). To represent the relatedness of any pair of elements from different schemas (e_i, e_j where $e_i \in E_{src}$ and $e_j \in E_{tar}$), we apply their absolute difference scaled by their sum and their cosine similarity to compute a semantic similarity $\mathbf{ls_{i,j}}$ (see Eq. 2.1) as well as a similarity embedding $\mathbf{l_{i,j}}$ (see Eq. 2.2). The circular sign in Eq. 2.2 refers to the element-wise division between two vectors with the same dimension. A classifier takes $\mathbf{l_{i,j}}$ as an input to predict if e_i and e_j are matched.

$$\mathbf{ls_{i,j}} = \left[\frac{\mathbf{s_i} \cdot \mathbf{s_j}}{\|\mathbf{s_i}\| \, \|\mathbf{s_j}\|}; \mathrm{BLEU}(e_i, e_j); \mathrm{lev}(e_i, e_j); \mathrm{lcs}(e_i, e_j)\right] \tag{2.1}$$

$$\mathbf{l_{i,j}} = \left[\mathbf{ls_{i,j}}; \frac{\mathbf{v_i} \cdot \mathbf{v_j}}{\|\mathbf{v_i}\| \, \|\mathbf{v_j}\|}; |\mathbf{v_i} - \mathbf{v_j}| \oslash (\mathbf{v_i} + \mathbf{v_j} + \varepsilon)\right] \tag{2.2}$$

The semantic similarity $\mathbf{ls_{i,j}}$ between element names is the concatenation of the cosine similarity between semantic embeddings, BLEU score, Levenshtein distance (lev) and length of the longest common substring (lcs). We use a fine-tuned pre-trained BERT-based model, Sentence-BERT [11], to help build semantic embeddings for the element names. Compared to the original BERT model, the Sentence-BERT achieves better performance on various sentence-pair regression tasks, especially semantic textual similarity, and this advantage is consistent with our goal to evaluate the similarity between two name embeddings. Its fine-tuning process is shown in Figure 2.3, where the pooling layer takes average over all output vectors from the BERT and the downstream task is defined to compute two BERT embeddings' cosine similarity.

To strengthen the robustness of a semantic embedding via full-filling the content of an element name, we try to add tags (e.g. attributes, type, function, etc) to depict elements based on their names and instances. A fine-tuned large language model (LLM) is responsible for generating element tags, and then they are concatenated with the element name to form a new linguistic representation. The LLM chosen is the BLOOMZ & mT0[1], which comes from BLOOM & mT5

[1]https://huggingface.co/bigscience/bloomz

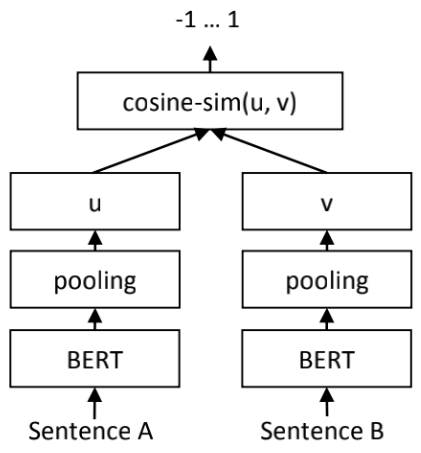

Figure 2.3: **Sentence-BERT architecture at fine-tuning** [11]

pretrained multilingual language models fine-tuned on the crosslingual task mixture [12]. Take the element *funding* as an example. Once we input this name as well as its instances into the LLM with a prompt. The generated tags will be *value*, *total* and *usd*. They will be concatenated with the name *funding* and then transformed into a semantic embedding.

The instance embedding involves different type-dependent features like the numeric feature, character feature, semantic feature, etc. These features are only valid if they are under the right type of instances; otherwise, they become {-1}-filled vectors to indicate their incompatibility. For example, an element *price* should have a valid numeric feature and an element *institution name* is supposed to have a valid character and a valid semantic feature. The numeric feature contains a set of statistical metrics such as the mean, minimum/maximum value, variance, standard deviation and number of unique instances to show the instance distribution; while the character feature computes the average ratio of white spaces, punctuation, special characters and numeric characters from string-type instances of a given element. A semantic feature is the same as the embedding from a fine-tuned Sentence-BERT. The instance embedding also has type-independent features, including the instance type by a one-hot vector and average instance length. All these data-oriented features help explore an element's latent meaning from multiple perspectives.

2.4.2 Classifier

To predict a matched pair given a similarity embedding integrating schema-based and instance-based features, we make use of the XGBoost classifier, a scalable tree boosting system [13] that delivers impressive performance on many machine learning tasks. Compared to the gradient boosting decision tree, an XGBoost is more regularized due to its in-built L1 and L2 regularization, and thus it comes with better generalization capabilities. Unlike the linear classifier proposed in Zhang et al. [2], a tree-based classifier provides much more complex decision boundaries when predicting the relatedness of two schema elements. Since its output is a match score indicating the probability that two elements are matched, we enable users to specify thresholds to determine any match. A default threshold is computed according to our model validation and labeled matched pairs during the training process.

CHAPTER 3
EXPERIMENT

3.1 Datasets

The experimental datasets have varying difficulties and they are either widely used in the field of schema matching or created based on real-world data sources. They are representative enough to create convincing evaluation for various schema matching techniques.

3.1.1 Synthetic Dataset

Possibly the most challenging issue on evaluating schema matching methods is the lack of openly available datasets with schema matching ground truth (or labels). To resolve this problem, Koutras et al. [1] proposes a methodology, *Valentine*, to create different types of schema matching datasets from a single schema. Given a tabular schema, we can split it horizontally to create *unionable* pairs, vertically to make *joinable* pairs, or in both ways, following the ideas from [14][15]. To be specific, we create a *unionable* dataset by horizontally partitioning the table with varying percentages of row overlap, and we make a *view-unionable* dataset by splitting a table both horizontally and vertically with zero row overlap and varying column overlap. In contrast to a *view-unionable* dataset, a pair of *joinable* tables should have at least one column in common and a large row overlap. The *semantically-joinable* dataset is similar to the *joinable* one, except that their element (column) names are noisy (semantic variations). In addition, all instances under different types of datasets are manually made noisy. We use the WikiData[1] in *Valentine*, which contains 4 pairs of schemas

[1]https://delftdata.github.io/valentine/

and varies from 13 to 20 columns and 5423 to 10846 rows, to evaluate the inference of our trained model.

3.1.2 Public Benchmarks

Another method to create schema matching datasets is to build element-wise mappings from two associated tabular datasets. For example, the MovieLens[2] and IMDB[3] datasets are commonly used to create schema pairs since their elements, though linguistically different, refer to the similar attributes like *rating* vs. *averageRating* or *title* vs. *originalTitle*. The MovieLens-IMDB dataset has been widely used in the field of schema matching [2][10], but there is not a standard version of it. Our MovieLens-IMDB dataset has 2 pairs of schemas and each schema has 1000 rows. Its column number varies from 4 to 10.

Through the way of element-wise mappings from related datasets, we create the first large-scale schema matching dataset on the humanitarian data. Individual tabular datasets are collected from The Humanitarian Data Exchange[4] (HDX). The HDX schema matching dataset contains 226 pairs of schemas and ranges from 6 to 41 columns, but for each schema, the row number ranges from 20 to 100. Compared with the Wikidata and MovieLens-IMDB, the HDX dataset is expected to be the most challenging dataset since it has the largest scale, the highest tabular variations but the lowest number of instances.

[2]https://grouplens.org/datasets/movielens/
[3]https://www.imdb.com/interfaces/
[4]https://data.humdata.org/

3.2 Experimental Settings

3.2.1 Models

For the semantic embedding, we use a pre-trained sentence-BERT model fine-tuned for the task of semantic similarity from the Hugging Face[5]. This model maps any piece of text to a 768 dimensional dense vector, and its name is `paraphrase-multilingual-mpnet-base-v2`[6]. It is built upon the `mpnet-base`[7] model by Microsoft and fine-tuned on a 1B sentence pairs dataset. As we have mentioned before, we try to utilize the BLOOMZ & mT0 model to generate tags associated with the schema name so as to expand its semantic embedding. The model applied here is `mt0-xl`[8], which has 3.7B parameters and is recommended for the tag prompting in English. We adopt an XGBoost model to predict the probability that two schema elements encoded in the similarity embedding are matched, and its objective is to do a binary classification (matched or not).

We also implement several rule-based schema matchers for the baseline performance evaluation. We use *WordNet*[9] as thesaurus to calculate the linguistic similarity in Cupid, and apply the levenshtein distance to build initial element mappers in the similarity flooding method. Other experimented schema matching techniques are the COMA system, distribution-based method and Jaccard-Levenshtein method.

[5]https://huggingface.co/
[6]https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2
[7]https://huggingface.co/sentence-transformers/all-mpnet-base-v2
[8]https://huggingface.co/bigscience/mt0-xl
[9]https://wordnet.princeton.edu/

3.2.2 Machine Learning Pipeline

We train our classifier on three synthetic datasets, which are the TPC-DI (180 pairs of schemas, 7492-14983 rows, 11-12 columns), Open Data (180 pairs of schemas, 11628-23255 rows, 26-51 columns) and ChEMBL (180 pairs of schemas, 7500-15000 rows, 12-23 columns) from *valentine*[10]. We want our trained model to be aware of different matching logics (e.g. *joinable*, *unionable*, etc) mentioned in [1]. Hyperparameters tuned for our XGBoost classifier involves the learning rate, maximum tree depth, number of rounds for boosting, etc. The validation and inference metrics are the macro-recall (mR), macro-precision (mP) and macro-F1 ($\mathrm{mF1}$). Assume for each schema pair, the resulted recall, precision and F1 are $\mathrm{R_i}$, $\mathrm{P_i}$ and $\mathrm{F1_i}$ and there are n pairs of schemas in total for the current dataset. The macro-metrics are computed based on Eq. 3.1, 3.2, 3.3.

$$\mathrm{mP} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}} \mathrm{P_i} \tag{3.1}$$

$$\mathrm{mR} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}} \mathrm{R_i} \tag{3.2}$$

$$\mathrm{mF1} = \frac{2 \cdot \mathrm{mP} \cdot \mathrm{mR}}{\mathrm{mP} + \mathrm{mR}} \tag{3.3}$$

3.3 Results and Analysis

The inference results of our trained hybrid embedding-based model (hybrid model) on three different datasets are presented in Tables 3.1, 3.2 and 3.3. SF, DB and JL refer to the similarity flooding, distribution-based matcher and Jaccard-Levenshtein model. The domains of these in-

[10]https://delftdata.github.io/valentine/

ference datasets are different from the training one, so we can evaluate the cross-domain performance of our machine learning-based model. Wikidata provided by *valentine* has different types of datasets based on four table-splitting strategies (see Table 3.1). Though the hybrid model's general F1 performance is close to the best one from the JL, its individual evaluation on the *joinable* dataset is much worse than JL's. In a *joinable* pair of schemas, there is a large overlap of rows. Since the JL is a naive approach matching elements based on the row-value distribution from each schema, it is not surprised that given a large number of rows in each schema (more than 5000), this model can do a perfect schema matching. This observation is also found in the performance comparison between schema-based (COMA, Cupid, SF) and instance-based (DB, JL) models, where data-oriented methods do a better job on the WikiData than the schema-based methods.

Compared to the JL that purely focuses on instances, our hybrid model considers the semantic variations of element names. As a result, in the *sem-joinable* dataset, the JL's F1 score is less than the hybrid model's. We also notice that adding generated tags to our hybrid model does not improve its F1 performance upon the WikiData, and its performance on the *sem-joinable* dataset is worse than before. Though generated tags from an LLM are supposed to enrich the semantic contents of element names, they may weaken the original semantic characteristics of element names if generated tags are highly close to each other for different elements. For example, the element *funding* and *capital* both have generated tags like *value*, *total* and *usd*. From the perspective of computational linguistics, a high textual overlap between these elements with their concatenated tags shows a high semantic similarity. However, they are not supposed to be matched.

MovieLens-IMDB and HDX are more realistic datasets, and compared to *valentine*, they are harder for instance-based matchers to handle since every schema has significantly fewer number of row values. For the MovieLens-IMDB dataset (see Table 3.2), our hybrid model achieves the best classification performance and the addition of generated tags to it does not worsen the original

	COMA	Cupid	SF	DB	JL	Hybrid	Hybrid+tags
unionable	0.52	0.95	0.92	0.92	**0.97**	**0.97**	0.95
view-unionable	0.67	0.46	0.59	0.60	**0.80**	**0.80**	0.75
joinable	0.67	0.46	0.59	0.92	**1.00**	0.78	0.86
sem-joinable	0.71	0.80	0.63	0.67	0.82	**0.88**	0.80
mean	0.64	0.68	0.68	0.78	**0.89**	0.86	0.84

Table 3.1: **Inference on WikiData (F1-score)**

performance. It is not surprising to see a perfect match here because the total number of element pairs to predict is much fewer than before. In contrast to the Table 3.1, schema-based methods (COMA, Cupid and SF) all obtain higher F1 scores than the DB, a typical instance-based method, and Cupid shows exactly the same matching quality as the JL. With fewer number of instances for references, instance-based methods' ability to find matched pairs decreases, but is still competitive in terms of JL's performance over that of COMA and SF.

	COMA	Cupid	SF	DB	JL	Hybrid	Hybrid+tags
macro-recall	0.83	0.83	0.83	0.42	0.83	**1.0**	**1.0**
macro-precision	0.71	0.83	0.58	**1.00**	0.83	**1.0**	**1.0**
macro-F1	0.74	0.83	0.69	0.59	0.83	**1.0**	**1.0**

Table 3.2: **Inference on MovieLens-IMDB**

	COMA	Cupid	SF	DB	JL	Hybrid	Hybrid+tags
macro-recall	0.49	0.66	0.72	0.22	0.42	0.73	**0.74**
macro-precision	0.78	0.76	0.69	0.62	0.64	0.79	**0.81**
macro-F1	0.60	0.71	0.70	0.33	0.51	0.76	**0.77**

Table 3.3: **Inference on HDX**

Table 3.3 displays the inference results on the HDX dataset. Compared to the WikiData and MovieLens-IMDB, the HDX has many fewer instances in each schema. Consequently, the F1 scores of all schema-based models (COMA, Cupid and SF) are higher than those of DB and JL, which are heavily dependent on the instance samples. Our hybrid model with generated tags has the best performance over all schema-matching methods. This result indicates the robustness of hybrid embeddings when the elements of different schemas have high semantic and amount variations, and the number of their corresponding instances is limited (less than 100 on average). The generated tags slightly improves the F1 performance of our hybrid model, and this indicates the semantic enrichment given by additional attributes for the element names. The HDX has the highest number of schema pairs and thus the highest number of unique element names among all experimental datasets. We can thus observe that the effectiveness of the generated tags is somehow related to the semantic variation in the dataset. These tags are more helpful for the difficult dataset (HDX) than the easy one (WikiData).

In a nutshell, our hybrid embedding-based model shows no compromised performance under any specific scenario and its schema matching quality is generally better than existing approaches. Compared to schema-based methods, our approach is not heavily dependent on the linguistic similarity of element names and its semantic embedding, which applies a BERT-based encoder with an ad hoc fuctionality to compute textual similarity, is built in a fully automatic fashion. Unlike instance-based approaches, our model involves schema-based information so it's more successful on schemas where the number of instances is low. Its core idea of integrating both schema-based and instance-based features proves to be an effective trade-off between opposite methodologies (schema-based vs. instance-based).

CHAPTER 4
CONCLUSION AND FUTURE WORK

4.1 Summary

This paper presents a novel relational schema matching method that creates hybrid embeddings for schema elements and applies XGBoost to compute the match probability. A hybrid embedding involves both schema-based features of element names, which are contextual embeddings from a BERT-based model and similarity metrics for strings, and instance-based features dependent on data types. Our model outperforms previous schema matching methods on different benchmarks, and it is robust when the schema elements are messy and their instances are limited. The integration of schema-based and instance-based information is shown to be more powerful than a single feature source (either schema or instance) for the task of schema matching. Though the improvement from the LLM-generated tags on elements is slight, its effectiveness is empirically proportional to the complexity of source and target schemas.

4.2 Future Work

Our model consists of two parts: the hybrid embedding and a classifier. Though we have formed a mature methodology on the creation of a hybrid embedding, there is still space left for us to improve the quality of the classifier. The LightGBM classifier proposed in [16] may offer a better matching efficiency and accuracy than our used one. It applies a histogram-based algorithm to reduce the time spent on finding the optimal split point of any numeric feature. Another idea is to replace the tree-based classifier with a neural network. In contrast to our current model, whose embedding and classifier are segregated at the training phase, a neural-network classifier enables us to conduct an end-to-end gradient update on not only the classifier but also the semantic embedding. This may improve the matching quality, but the training of an additional BERT-based encoder increases the computation cost.

Privacy is also a necessary concern for the industrial use of the model. In fact, many customers are reluctant to grant access to individual data records [2], which are data instances in tabular schemas. A cryptographic strategy to extract instance-based features from databases and build their hybrid embeddings requires further research effort. At last, some recent deep-learning based schema matching techniques as well as their used datasets have not been released to the public yet. We will compare their performance with ours in the future.

REFERENCES

[1] C. Koutras, G. Siachamis, A. Ionescu, *et al.*, "Valentine: Evaluating matching techniques for dataset discovery," *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pp. 468–479, 2020.

[2] Y. Zhang, "Schema matching using pre-trained language models," 2022.

[3] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic schema matching with cupid," in *Very Large Data Bases Conference*, 2001.

[4] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: A versatile graph matching algorithm and its application to schema matching," *Proceedings 18th International Conference on Data Engineering*, pp. 117–128, 2002.

[5] H. H. Do and E. Rahm, "Coma - a system for flexible combination of schema matching approaches," in *Very Large Data Bases Conference*, 2002.

[6] Q. Liu, M. J. Kusner, and P. Blunsom, "A survey on contextual embeddings," *ArXiv*, vol. abs/2003.07278, 2020.

[7] J. Zhang, B. Shin, J. D. Choi, and J. Ho, "Smat: An attention-based deep learning solution to the automation of schema matching," *Advances in databases and information systems. ADBIS*, vol. 12843, pp. 260–274, 2021.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *ArXiv*, vol. abs/1810.04805, 2019.

[9] R. J. Miller, L. M. Haas, and M. A. Hernández, "Schema mapping as query discovery," in *Very Large Data Bases Conference*, 2000.

[10] M. Zhang, M. Hadjieleftheriou, B. C. Ooi, C. M. Procopiuc, and D. Srivastava, "Automatic discovery of attributes in relational databases," in *ACM SIGMOD Conference*, 2011.

[11] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Conference on Empirical Methods in Natural Language Processing*, 2019.

[12] N. Muennighoff, T. Wang, L. Sutawika, *et al.*, “Crosslingual generalization through multitask finetuning,” *ArXiv*, vol. abs/2211.01786, 2022.

[13] T. Chen and C. Guestrin, “Xgboost : Reliable large-scale tree boosting system,” 2015.

[14] F. Nargesian, E. Zhu, K. Q. Pu, and R. J. Miller, “Table union search on open data,” *Proc. VLDB Endow.*, vol. 11, pp. 813–825, 2018.

[15] Y. Lee, M. Sayyadian, A. Doan, and A. S. Rosenthal, “Etuner: Tuning schema matching software using synthetic scenarios,” *The VLDB Journal*, vol. 16, pp. 97–122, 2007.

[16] G. Ke, Q. Meng, T. Finley, *et al.*, “Lightgbm: A highly efficient gradient boosting decision tree,” in *NIPS*, 2017.