Computers, Materials & Continua
DOI:10.32604/cmc.2022.020695
Article

Tech Science Press

Relation-Aware Entity Matching Using Sentence-BERT

Huchen Zhou[1], Wenfeng Huang[1], Mohan Li[1,*] and Yulin Lai[2]

[1]Cyberspace Institute of Advance Technology, Guangzhou University, China
[2]Department of Informatics, King's College London, United Kingdom
*Corresponding Author: Mohan Li. Email: limohan@gzhu.edu.cn
Received: 03 June 2021; Accepted: 15 September 2021

Abstract: A key aspect of Knowledge fusion is Entity Matching. The objective of this study was to investigate how to identify heterogeneous expressions of the same real-world entity. In recent years, some representative works have used deep learning methods for entity matching, and these methods have achieved good results. However, the common limitation of these methods is that they assume that different attribute columns of the same entity are independent, and inputting the model in the form of paired entity records will cause repeated calculations. In fact, there are often potential relations between different attribute columns of different entities. These relations can help us improve the effect of entity matching, and can perform feature extraction on a single entity record to avoid repeated calculations. To use attribute relations to assist entity matching, this paper proposes the Relation-aware Entity Matching method, which embeds attribute relations into the original entity description to form sentences, so that entity matching is transformed into a sentence-level similarity determination task, based on Sentence-BERT completes sentence similarity calculation. We have conducted experiments on structured, dirty, and textual data, and compared them with baselines in recent years. Experimental results show that the use of relational embedding is helpful for entity matching on structured and dirty data. Our method has good results on most data sets for entity matching and reduces repeated calculations.

Keywords: Knowledge fusion; entity matching; Sentence-BERT; relation aware

1 Introduction

The goal of entity matching (EM) is to identify heterogeneous expressions of the same real-world entity. For example, "Microsoft word 2007 version upgrade" and "Microsoft word 2007" both describe the product of Microsoft word 2007, and thus need to be identified as the same entity. Our goal is to find two entity records describing the same entity in the real world in two datasets. Fig. 1 shows an example of the EM task. In the Fig. 1, there are two product datasets containing three attributes: title, manufacturer, and price. The task of EM is to find two product records that may match the two datasets.

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Figure 1: Entity matching example task

Entity matching is an important issue that has been widely studied in knowledge fusion [1]. Its application scenarios include recommendation systems [2], data cleaning [3], etc. Previous work of entity matching favored the use of rule-based methods [4]. Due to the rapid development of deep learning, deep learning-based methods are used for feature extraction and entity identification. Due to the accumulation of various frameworks and engineering experience, deep learning based methods makes the original complex and tedious work easier to complete, but they also face many challenges. First, entity matching is to match two entity records describing real-world entities, and entity records may have few identical attributes, and attributes may be independent of each other, lacking potential information between attributes. Since the schemas of different datasets are different, it is difficult to calculate the text similarity of records from different datasets. Second, the current entity matching method lacks a powerful model to ensure accuracy. The previous research on entity matching was to explore matching algorithms and to construct entity matching models with high efficiency and high precision.

Previous researchers favored crowdsourcing and optimization algorithms and mostly used human-machine hybrid methods, but many of them ignored relation transfer between matches [5]. Because entity matching will produce a lot of negative examples, there are methods to utilize the locality-sensitive hashing schema to speed up the matching process and reduce the candidate pairs [6]. Many researchers have improved on crowdsourced methods [7–9]. However, the EM field has always lacked research on the construction of the overall EM system. How to match entities on multi-source heterogeneous data and build the overall system has always been a big challenge. Later, it gradually began to enter the overall system construction research of EM, such as Magellan [10] and DeepMatcher [11]. Magellan created a data preprocessing process and an end-to-end EM model construction process. DeepMactcher builds a deep learning entity matching module based on Magellan. The latest EM method is Ditto [12]. The construction idea of Ditto model is the same as the previous work. Blocking operation is performed first, and then an entity matching operation is performed. Magellan has strict requirements on the input data format, and machine learning methods are weak on dirty data and long text data. DeepMatcher and Ditto, the two representative methods in the EM field, have not solved a common problem, that is, the way they input entity records is to input a pair of entities for encoding calculations at the same time. In the entity matching task, entity sets A and B contain m and n entity records respectively, then the way of inputting entity pairs will produce $m \times n$ combined input models, which will be unnecessary calculation when the amount of data is large. During entity matching, one entity record will be cyclically compared with multiple entity records. If one entity record can be encoded only once, the input size will be reduced to $m + n$, which can effectively save time and space. The entity in the entity matching task has several attribute columns, and there are potential semantic relationships between the attributes of the entities. If they are simply concatenated, many potential semantic relationships between the attributes will be lost, and these classic works [10–12] did not use this potential information.

The method proposed in this paper is called Relation-aware Entity Matching using Sentence-BERT (REMS), which is based on the Sentence-BERT model and uses a distilled pre-trained BERT model [13]. We take advantage of the potential between the attributes in the entity record Semantic relations, and the perception of potential relations, adding relational embeddings between the attributes of entities, and constructing entity records into sentences. REMS uses a twin neural network, a model composed of a pair of parameter-sharing BERTs, which can encode each entity record separately, and each entity record will be saved after being encoded, reducing the size of the computational entity record encoding to $m + n$, where m and n represents for the size of entity records to be matched. Since BERT's pre-training model is trained on the *corpus* of plain text, it has a very good effect on processing natural language text. Taking entity records into sentences as input is similar to the pre-training process of the BERT model. We conducted experiments on 13 entity matching datasets and evaluated them. The average results of five experiments show that REMS has the best three datasets in the structured dataset, and one dataset on the dirty data reaches the best, but the performance on the long text dataset is not good enough. The difference from the previous method is that REMS uses relational embedding throughout the entire EM process and encodes entities separately, reducing time and space redundancy, and thereby improving the effect of EM. The main contributions of this paper are as follows.

- We provide a method for preprocessing the dataset. First, perceive the potential relationship between the attributes of the dataset, add relationship embedding to the entity record, and convert the entity record into the form of a sentence. As far as we know, this is a method of first fusing relational embeddings in entity matching, modeling entity records into sentence similarity tasks, and using pre-trained BERT models for separate encoding.
- We propose a framework for entity matching based on BERT. We also performed rule-based blocking on the dataset and data augment. We have independently blocked and processed each dataset in order to filter out a high-quality dataset from a large amount of negative data. In addition, the existing methods input entity records in pairs into the model, assuming that the EM dataset consists of tables *A, B* and the number of entity records in *A, B* is m and n respectively. Inputting entity records in pairs yields an encoding size of $m \times n$. We construct entity records into a sentence by relation aware and combine EM and Sentence-BERT models with a dual neural network. The purpose is to separately encode each entity record and save the result, which can reduce the encoding scale of the entity record to $m + n$.
- We conducted experiments and evaluated on 13 datasets, including structured, dirty, and textual data. We provide the performance of the BERT model in EM and some engineering experience.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 describes the details of REMS. Section 4 illustrates the experiment results. Section 5 concludes the paper and discusses future work.

2 Related Work

2.1 Entity Matching

Let D_1 and D_2 be two entity sets, the task of entity matching is to find a pair of entities that collectively describe the same thing in the real world $<e_i, e_j>$. This is a task that compares entity records and entity attributes. In this article, we assume that the datasets D_1 and D_2 have the same data pattern. Given entity record $t(A_1, A_2, \ldots, A_k)$, where A_i is the value of the *i-th* attribute of the entity t. The EM task includes two sections: Blocking and Matching. Since there are a large number of negative examples in the entity matching task, the purpose of the blocking process is to filter the entity record

candidate set S generated by $D_1 \times D_2$,and reduce the number of negative examples as much as possible. After blocking, there will still be some negative examples in the candidate set S, and the correct entity record pair needs to be selected in the matching link. Assuming that the candidate set after Blocking is S_1, the link of matching is to predict whether all the entity record pairs in the candidate set S_1 are matched or not matched.

Entity matching has always received extensive attention [14–16]. In traditional methods, more attention is paid to the construction of matching algorithms. A lot of work has been done on how to improve the accuracy of matching and reduce costs. It is possible to do research to adapt to more scenarios [17]. After that, many researchers built the entire system for the EM process and built an end-to-end model. Early research on entity matching favors the use of crowdsourcing to improve the accuracy of EM [6,8,9,18], but crowdsourcing methods are not suitable for reproduction and costly. Later, some researchers began to establish an EM system, but only Magellan established a complete EM process system. Recent research has begun to shift to the use of deep learning methods for EM. Below I elaborate on some work related to EM from three aspects.

2.1.1 Traditional Methods

Early research is doing some EM matching algorithms. Many traditional methods use crowdsourcing methods to assist entity matching. For example, some studies have constructed a transfer relationship crowdsourcing method to do entity matching. If e_1 and e_2 match, e_2 and e_3 are similar, their pairing satisfies the transfer relationship, and a heuristic annotation algorithm is also designed to label Crowdsourced sequence [5]. There are unsupervised methods for entity matching of multi-source data [6], and there is also the use of crowdsourcing to match the entities of the image [8], and some research makes the entire process of crowdsourcing does not require participation of staff [9]. There are methods to construct a probabilistic graphical model [18], which reduces the number of expert questions in crowdsourcing while maintaining the high accuracy of entity matching. There are also studies on entity matching of heterogeneous data from multiple sources [19]. There are also MapReduce-based EM methods [20] and DeepMatching methods for image matching [21]. There is a lot of potential information in EM tasks and the study of time series is becoming more and more important. There are studies of data entity matching with hidden temporal information [22] and time series [23]. Data cleaning is also particularly important in EM. Data cleaning ensures that EM matches in the cleanest possible environment. Some studies are weighted match correction rules (WMRRs) based on similarity matching to capture more errors [24] and regular expression-based data repair [25]. More recent approaches also compute the minimum coverage of ontology graph keys (OGKs) to do EM [26], and several ideas use classical algorithmic methods to help EM [27–29].

2.1.2 EM-Systems

The most classic method is Magellan [10], which is a complete EM system that covers the entire EM process and provides detailed process guidance. Later, DeepMatcher [11] established a deep learning EM scheme based on Magellan. Establishing an EM system is a very tedious process. The relatively early EM system D-Dupe [30] combines data mining and can be used for specific tasks. There are browser-based EM systems Dedoop [31] and NADEEF [32]. These studies have made a lot of contributions, but these EM systems have some shortcomings, these methods do not provide a complete EM process, and lack scalability, cannot carry out some personalized expansion of the system.

2.1.3 Deep Learning-Based Methods

In recent years, there has been more and more research using deep learning for EM. There are methods to design an end-to-end joint learning model, which adaptively selects an appropriate similarity measure for multi-source datasets [6]. The latest EM method is Ditto [12] using a pre-trained BERT model, which inputs the paired entity record sequence into the model to generate similarity. DeepER [33] is a deep learning EM system. This method constructs two deep learning models for entity matching, and also uses a distributed Blocking method. Similarly, DeepMatcher is the latest baseline, which provides a deep learning solution for EM, and establishes a deep learning entity matching method on Magellan tools, and is robust against dirty data, it provides EM fields a new idea.

2.2 BERT

With the rapid development of the BERT model [34] in the field of NLP [35], the pre-trained BERT model is widely used, and the industry and academia are very popular with BERT. For example, it has applications in text classification [36,37] and sentiment analysis [38,39]. BERT has also brought milestone changes to many fields. Sentence-BERT adds modules to BERT's pre-training model and performs fine-tuning, and has made new progress in the task of semantic text similarity [40]. BERT is a language representation model, which can achieve language representation target training, and achieve the purpose of semantic understanding through the two-way Transformer model [41]. However, because the BERT model has a large number of parameters, many distillation models and variant models have been derived, and it has maintained a high degree of accuracy while reducing the model parameters. For example, DistillBERT and ALBERT, DistillBERT has only half the number of layers of BERT, and also uses a combination of three-loss functions in terms of loss function, while retaining 95% of the performance of BERT measured by the GLUE language understanding benchmark. ALBERT [42] uses parameter sharing and improves the training speed. Under the same training time, its effect is better than BERT, but the overall inference result is still 2–3 points worse than BERT.

3 The REMS Approach

We used the twin neural network in Sentence-BERT [40], and the overall architecture diagram of the REMS model is shown in Fig. 1.

In the task of entity matching, we have constructed a method that constructs entity records into sentences through relational awareness and encodes them into vectors through the BERT model to perform entity matching calculations. The entity matching process of REMS is shown in Algorithm 1. The REMS model, it is mainly composed of three steps: relationship perception, Blocking, and Matching. In Fig. 2, Step 1 is relationship perception and preprocessing, which will perceive the relationship between the attribute columns of the entity records according to different types of datasets, add potential relationships between the attributes of the entities and form a sentence form. Then the data after the relationship perception is preprocessed to conform to the input of the model. Step 2 performs Blocking and data augmentation. Since a large number of negative cases will be generated in the EM task, Blocking is to delete some obviously mismatched entity record pairs. The dataset after Blocking can be augmented with data to expand the training set, structured and dirty data can be deleted at random, and long text datasets can be summarized and generated, and then deleted and other data augmentations can be performed. Step 3 performs Matching and fine-tuning. Fine-tuning is to input the training set into the model for training, verify the model on the validation

set, save the best model, and find the best similarity threshold parameters. After the model training is completed, perform Matching. It is to find the matching entity record pair in the two datasets. So, let's introduce these three steps below.

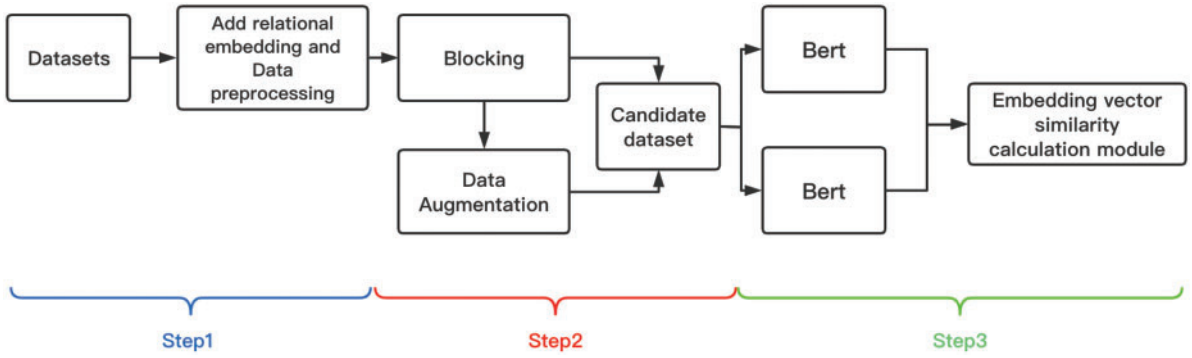

Figure 2: REMS overall architecture

Algorithm 1 REMS Entity Matching procedure

Input: EM datasets
Output: match or non-match
for all epochs **do**
  for all batches in the EM dataset **do**
    Relational Awareness of entity record $t(A_1, A_2, ..., A_k)$;
    After data preprocessing the result is entity record pair (t_A, t_B);
    if $(t_A, t_B) \geq$ *Blocking condition* **then**
      Candidate set $S \cup (t_A, t_B)$;
    end if
    for all batches in the Candidate set S **do**
      Data Augmentation of Candidate set S to generate training set T;
    end for
  end for
  for all batches in the training set T **do**
    Forward (t_A, t_B) to two separate BERT;
    Compute loss $\mathcal{L}_A, \mathcal{L}_B$ by Eq.(1);
    Update $\theta_{BERT_A}, \theta_{BERT_B}$ using $\nabla\mathcal{L}_A, \nabla\mathcal{L}_B$;
  end for
  for all batches in the prediction dataset **do**
    Forward prediction entity record (t_A, t_B) to REMS to generate result $r \in$ {match,non-match};
  end for
end for

3.1 Relation Aware

Step 1 is add relational embedding and Data preprocessing. Relational awareness is based on the attributes of entity records, adding relational words between attributes and attributes, enriching the semantic information of entity records and forming sentences. The specific method is as follows: given entity record $t(A_1, A_2, \ldots, A_k)$, where A_i is the value of the i-th attribute of the table A. *rel* (A_i, A_{i+1}) is a phrase, Its meaning indicates A_i, A_{i+1} potential relationship between. For example, the values of A_i, A_{i+1} are title and authors, and *rel* (A_i, A_{i+1}) is "made by". Potential relationship embedding will *rel* (A_i, A_{i+1})

embedded entity record $t(A_1, A_2, \ldots, A_k)$ Between the values of the corresponding attributes A_i, A_{i+1}, a sentence of the form "$A_1 rel\ (A_1, A_2)\ A_2 \ldots A_i rel\ (A_i, A_{i+1})\ A_{i+1} \ldots A_k$" is formed. The advantage of this is that the input entity record has richer semantics and the structure of the input conforms to the pre-training *corpus* of the BERT model. After the dataset is preprocessed, a relationship is added between the attributes of each column, so that a record of the entity and the attribute of the entity is converted into a sentence. For example, Microsoft word 2007 version upgrade [made by] Microsoft [cost] 109.95. The relation words "made by" and "cost" are the potential relationships we add between the attribute columns title, manufacturer, and price.

3.2 Blocking

Blocking is to remove some obviously mismatched entity record pairs. However, targeted Blocking is required for each dataset. For example, the DBLP-ACM dataset, it describes some bibliographic data of DBLP and ACM. The entity record of the dataset has a column of attributes named year. The purpose of Blocking is to delete entity record pairs with different year attribute values. The result is a significant reduction in the size of the candidate set. The Blocking tool we use is Magellan, and several different block methods are selected for preprocessing. For example, we use the rule-based Blocker L, to match the table A, B Blocking $(t_A \in A, t_B \in B)$ and then generate a set C, such as (t_A, t_B), the rule R is

$$R := (t_A.A_i \neq t_B.B_i) \wedge (t_A.A_i \geq T) \tag{1}$$

T is artificially defined fixed values, the entity records pairs that meet the rules are the elements of the set C. The final candidate set after blocking $U = A \times B \backslash C$. The data after Blocking is then augmented by methods such as random word deletion and BERT-substitute [43], etc. BERT-substitute focuses on predicting the target position i based on the context. The context is a sequence of words surrounding an original word w_i in an entity record sentence S, *i.e.*, sentences composed of surrounding words $S \backslash \{w_i\}$. The calculated replacement probability is $p\ (\cdot | S \backslash \{w_i\})$.

3.3 Matching

Finally, we use different DL entity matching methods. Most of them use neural networks such as RNN and LSTM, and rely on high-quality word embedding. However, the representation ability of traditional neural networks is no longer sufficient to face heterogeneous data source. The BERT model has a strong learning ability, can learn the semantics of the vocabulary and the overall meaning of the sentence, and has a better ability to understand sentences. We use the Sentence-BERT model as the pre-training model. The Sentence-BERT model diagram is shown in Fig. 3. After Step 2, we get the entity record pair (t_A, t_B), then forward (t_A, t_B) to two BERT models with shared parameters, use Eq. (2) to calculate the loss $\mathcal{L}_A, \mathcal{L}_B$, and then update the BERT model parameters θ_A, θ_B using $\nabla \mathcal{L}_A, \nabla \mathcal{L}_B$. We fine-tuning the pre-trained model so that the model can make more accurate predictions in a specific field. The twin neural network is used to enhance the effect of two classifications, and the loss function and parameters are optimized to make the sentence input the fine-tuning model can produce high-quality embedding vectors. After the BERT model outputs the embedding vector, connect a layer of MEAN-strategy pooling to generate a fixed-size sentence embedding. And use the loss function of mean square error.

$$\left\| label - cosine_{sim(a,b)} \right\|_2 \tag{2}$$

For each dataset, use the data augment training set to perform fine-tuning of the BERT model.

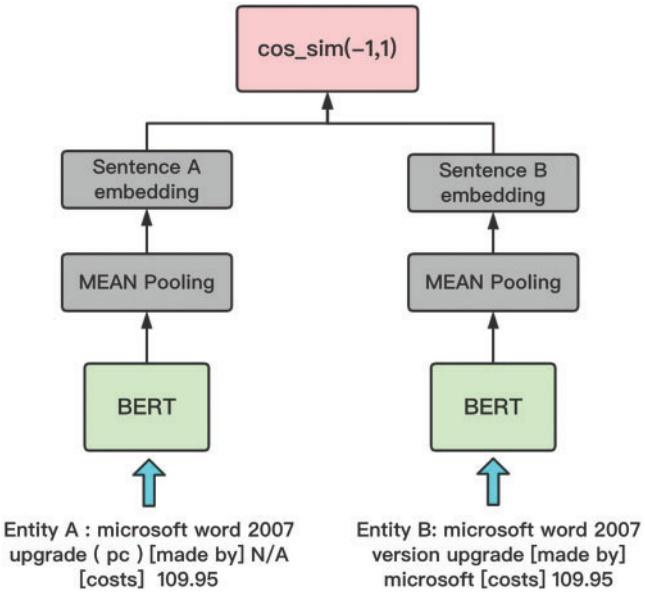

Figure 3: Sentence-BERT model

Because the process of finding the same entity record in the two datasets will produce a huge number of negative examples, if all entity record pairs are input into the BERT model for the similarity calculation method, the calculation time of this method is unacceptable. The REMS method will individually input each entity record into the BERT model for encoding and save the result. Finally, the vector encoded by the BERT model is calculated for similarity, and an optimal similarity threshold is selected.

4 Experiments

4.1 Experiment Settings

We experimented with the model on EM's benchmark datasets, which contained three kinds of problems, structured, dirty, and textual EM problem scenarios. There are 13 datasets in total. The source of the dataset is the source data downloaded from DeepMatcher's homepage[1]. Structured data includes Beer, iTunes-Amazon, Fodors-Zagats, DBLP-ACM, etc. We use four kinds of dirty datasets. The dirty thing is that there is a 50% probability that the attribute value will be moved to the same tuple "title" attribute. For example, in DBLP-ACM dataset, there is an attribute named year, and its value may be moved to the title attribute column. The original cell will be empty, and the attribute of each row will be moved to the title attribute with a 50% probability of combining with the title attribute, leaving the original cell empty. This is often encountered in real life scenes. For the textual datasets, we use Abt-Buy and the company dataset. Abt-Buy is the product data of Abt.com and Buy.com. The attributes have three columns, and we also deal with it relational. But the company dataset has only one column of attributes, so there is no relation aware processing.

[1] https://github.com/anhaidgroup/deepmatcher/blob/master/Datasets.md

The fields covered by all datasets include beer, music, restaurant, citation, etc. We first use the Magellan tool to combine blocks based on rules, and then divide the candidate dataset into training set, validation set, and test set at a ratio of 3:1:1. The statistics of datasets are given in Tab. 1.

Table 1: Dataset statistics

Data type	Name	Domain	Entity record-num (attr-num)
Structured	Beer	Beer	450 (4)
	iTunes-Amazon$_1$	Music	539 (8)
	Fodors-Zagats	Restaurant	946 (6)
	DBLP-ACM$_1$	Citation	12363 (4)
	DBLP-Scholar$_1$	Citation	28707 (4)
	Amazon-Google	Software	11460 (3)
	Walmart-Amazon$_1$	Electronics	10242 (5)
Dirty	iTunes-Amazon$_2$	Music	539 (8)
	DBLP-ACM$_2$	Citation	12363 (4)
	DBLP-Scholar$_2$	Citation	28707 (4)
	Walmart-Amazon$_2$	Electronics	10242 (5)
Textual	Abt-Buy	Product	9575 (3)
	Company	Company	112632 (1)

We compare our method with Magellan [10], an overall EM system based on machine learning, DeepMatcher [11], an overall framework based on deep learning, and the latest EM model Ditto [12] based on pre-trained BERT because Ditto uses a lot of expert knowledge and manual annotation, we choose Ditto without optimizations model as the comparison object.

4.2 Results

The results of the experiment all use the F_1 Score as the measurement standard. We use the average F_1 Score of the results of five experiments as the result. The experimental results of structured data are shown in Tab. 2.

Tab. 2 shows a comparison between REMS and several baselines and the latest method Ditto without optimizations on structured data. Structured data is relatively clean data, in the form of an entity record including several entity attribute columns, REMS performs well on structured data, and the best F_1-score is achieved on three structured datasets. According to our observations, through relational awareness, the sentence semantics of entity records were enriched by relation-awareness on the beer and iTunes-Amazon datasets, and the Beer and iTunes-Amazon$_1$ datasets are tidier and cleaner than before. After the less ambiguous sentences are subjected to relational perception, the generated entity record sentences perform better. For example, the best results were achieved on the Beer and iTunes-Amazon datasets. In particular, the Beer dataset has improved the F_1-Score of 12.06. The reason for the poor performance on the Amazon-Google and Walmart-Amazon$_1$ datasets, we believe, is that REMS is not able to grasp the relative information in response to data noise. For example, in the Walmart-Amazon$_1$ dataset, the entity records are all product records. The attributes are title, manufacturer, and price. Two of the entity records are "money premium 2007 win32 eng na mini

box us only cd, microsoft, and 118.46". And "money prem 2007 cd minibox, microsoft, 63.99". These two entities record actually describe the same entity, but because of the ambiguity in the information, many word abbreviations appear, and the meanings expressed are ambiguous. The sentence will not be rich in semantic information. Need more expert knowledge to supplement, you can refer to Ditto's method for domain knowledge injection. Because the output of the BERT model used by the model is then connected to an average pooling layer, the same attention is paid to the token vectors of all words in the sentence, so it will be insufficient to deal with more noisy information.

Table 2: Experimental results of structured data (F_1 score)

Datasets	Magellan	DeepMatcher	Ditto	REMS	ΔF_1
Beer	78.8	72.7	84.59	96.65	+12.06
iTunes-Amazon$_1$	91.2	88.5	92.28	98.18	+5.9
Fodors-Zagats	100	100	98.14	100	0.0
DBLP-ACM$_1$	98.4	98.4	98.96	98.18	−0.78
DBLP-Scholar$_1$	92.3	94.7	95.6	91.74	−3.86
Amazon-Google	49.1	69.3	74.1	65.30	−8.80
Walmart-Amazon$_1$	71.9	67.6	85.81	71.34	−14.47

On all datasets, we also used a variety of data augmentation methods to conduct experiments and increase the training set. There are five main methods, BERT-insert, BERT-substitute, swap, random-del, and summAug [43]. The two methods of BERT-insert and BERT-substitute are to input text into the model and provide a text environment for the model to find the most suitable extended words and add or replace words into the input text for data augmentation. BERT-insert is to insert extended words in the text, and BERT-substitute is to replace suitable words with extended words in the text. The Swap augmentation method is to replace the order of several words in the input text, random-del is to randomly delete the words in the input text, and the summAug augmentation method is to summarize the input text through a summary method. There are a variety of parameters to choose from for the above data augmentation methods. Tab. 3 is the F_1-Score result of REMS without data augmentation and after using data augmentation. In the structured dataset Walmart-Amazon$_1$, we use the Random-del data augmentation method. The Walmart-Amazon$_1$ dataset has the best F_1-score improvement performance after data augmentation. We observed that one of the attributes in this dataset "modelno" was important, so we made this attribute recur when generating the sentences, and then used randomly deleted data augmentation for the sentences, reducing the number of other the data augmentation of the sentence with random deletion reduces the impact of other irrelevant words on the sentence information. The results show that such an approach is effective. The Fodors-Zagats dataset is relatively simple, so no data augmentation is given. The Amazon-Google dataset did not find a suitable data augmentation method. We used a variety of methods, but the performance was not good.

In Tab. 4 is the experimental result of dirty data. Dirty data means that each column attribute has a 50% probability of being transferred to the title attribute, and the original value becomes empty. The REMS method achieves the best in one of the four datasets. The reason is that it is more robust in the

dataset with less ambiguity. The performance of iTunes-Amazon$_2$, DBLP-ACM$_2$, REMS are all good. It can be seen that Magellan's performance on dirty data is not enough, because dirty data has low string similarity and cannot provide high-quality information. Magellan is based on String similarity is used for entity matching, so the performance is not good on dirty data. DeepMatcher, Ditto, and REMS perform significantly better on dirty data than Magellan. And Ditto and REMS methods have higher robustness on dirty data, We believe that the way the sentence is input into BERT is more natural to treat BERT as an Encoder that specifically generates sentence embeddings. It is difficult to match the Walmart-Amazon$_2$ dataset. There will be many products model information and many abbreviations in the dataset. There are many ambiguities in the information, and REMS is not performing well in this piece.

Table 3: Data augmentation results of structured data (F_1 score)

Datasets	REMS (non-Aug)	REMS (Aug)	Method	ΔF_1
Beer	96.65	96.65	–	–
iTunes-Amazon$_1$	97.60	98.18	BERT-substitute	+0.58
Fodors-Zagats	100	100	–	–
DBLP-ACM$_1$	97.65	98.18	Swap	+0.53
DBLP-Scholar$_1$	91.44	91.74	Swap	+0.30
Amazon-Google	65.30	65.30	–	–
Walmart-Amazon$_1$	66.32	71.34	Random-del	+5.02

Table 4: Experimental results of dirty data (F_1 score)

Datasets	Magellan	DeepMatcher	Ditto	REMS	ΔF_1
iTunes-Amazon$_2$	46.8	79.4	92.92	94.74	+1.82
DBLP-ACM$_2$	91.9	98.1	98.92	98.19	−0.73
DBLP-Scholar$_2$	82.5	93.8	95.44	91.76	−3.68
Walmart-Amazon$_2$	37.4	53.8	82.56	65.74	−16.82

Tab. 5 is the augmentation results of dirty data. There is an average 2.48 F_1-score improvement on the four dirty datasets. BERT-insert has the best effect on the iTunes-Amazon$_2$ dataset because the input text has a good language environment and the improvement effect is more obvious. On the whole, because of the increased difficulty of entity matching for dirty datasets, the effect of using data augmentation on dirty data is better than structured data as a whole, and the effect of using data augmentation on entity matching datasets will be improved to a certain extent.

Tab. 6 is an experiment on the textual datasets. The number of attribute columns of the textual dataset will be relatively small. Especially in the Company dataset, the content is the content crawled from the company's Wikipedia page. After processing, the long text information is equivalent to describing a company. The company dataset is a long text matching. If text summarization is not used, the performance on the long text dataset will be poor, long text summary requires a lot of manual

definition operations and expert knowledge. We pursue a more automatic EM process, so we didn't use it. REMS does not perform well on textual datasets. We believe that it is because BERT has connected to the average pooling layer after outputting the embedding vector. Because the text of the data is too long, the pooling layer does not emphasize the focus of the text.

Table 5: Data augmentation results of dirty data (F_1 score)

Datasets	REMS (non-Aug)	REMS (Aug)	Method	ΔF_1
iTunes-Amazon$_2$	89.10	94.74	BERT-insert	+5.64
DBLP-ACM$_2$	97.96	98.19	BERT-insert	+0.23
DBLP-Scholar$_2$	91.66	91.76	BERT-insert	+0.10
Walmart-Amazon$_2$	61.78	65.74	Random-del	+3.96

Table 6: Experimental results of textual data (F_1 score)

Datasets	Magellan	DeepMatcher	Ditto	REMS	ΔF_1
Abt-Buy	33	55	88.85	67.4	−21.45
Company	79.8	92.7	41.00	80.73	−11.97

We use the F_1 Score as our evaluation score, and the highest F_1 model result in each dataset is marked in red. The structured data is relatively clean. After adding relational embedding, REMS achieves the best performance on the four structured entity matching datasets. There are improvements, compared with previous deep learning and machine learning methods. We found that adding the length of the relationship embedding will also have an impact. The character length of the relationship embedding should not be too long, and 1–2 words will achieve the best performance.

4.3 Training Details

Our model is tested on 13 datasets. The framework of the model is implemented using Pytorch and runs on Intel(R) Xeon(R) Silver 4110 CPU @ 2.10 GHz and Nvidia Tesla T4 GPU. The learning rate used is the default 2e-5, transformer AdamW [44]. The learning rate is not the main influencing factor. Using the default setting is the most stable. The main reason is that the warmup steps setting is 10% of the training set. The evaluation steps are twice the size of the warmup steps. This combination will be more appropriate. The batch size is set according to the configuration of the graphics card, and the batch size is set to 64. The value of Epoch is set differently for each dataset. We search for the best parameters from the initial setting of 15 and search for the best epoch for the textual datasets in 2–4. Too many epochs will cause overfitting. The word count of relational embedding is the best setting when it is 1∼2, excessively long relationship words may degrade performance.

5 Conclusions and Future Work

BERT's pre-training model has been successfully applied in many fields. In our work, we design a sentence-level entity matching model that includes relational embeddings. We recommend adding relational embeddings to entity matching, which helps to enhance the semantic relationship between attributes, using pre-trained models to train and embed the sentences recorded by the entity. The model

separately encodes and saves the entity records, reducing the size of the input. Experiments have proved that it has a certain improvement compared with the traditional method. This study identified REMS with good prospects. In the future, we will continue to study the role of relationships in entity matching and entity matching in graph structures.

Acknowledgement: We would like to thank anonymous reviewers and the associate editor for the constructive comments in improving this work.

Funding Statement: This work is funded by Guangdong Basic and Applied Basic Research Foundation (No. 2021A1515012307, 2020A1515010450), Guangzhou Basic and Applied Basic Research Foundation (No. 202102021207, 202102020867), the National Natural Science Foundation of China (No. 62072130, 61702220, 61702223), Guangdong Province Key Area R&D Program of China (No. 2019B010136003, 2019B010137004), Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2019), and Guangdong Higher Education Innovation Group (No. 2020KCXTD007) and Guangzhou Higher Education Innovation Group (No. 202032854).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] H. Zhou, M. Li and Z. Gu, "Knowledge Fusion and Spatiotemporal Data Cleaning: A Review," in *2020 IEEE Fifth Int. Conf. on Data Science in Cyberspace (DSC)*, Hong Kong, China, pp. 295–301, 2020.

[2] M. Peng, M. Li and Q. Guan, "Random shilling attacks against latent factor model for recommender system: An experimental study," in *Int. Conf. on Artificial Intelligence and Security*, Singapore, Springer, pp. 154–165, 2020.

[3] M. Li, J. Li, S. Cheng and Y. Sun, "Uncertain rule based method for determining data currency," *IEICE TRANSACTIONS on Information and Systems*, vol. 101, no. 10, pp. 2447–2457, 2018.

[4] L. Li, J. Li and H. Gao, "Rule-based method for entity resolution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 1, pp. 250–263, 2014.

[5] J. Wang, G. Li, T. Kraska, M. J. Franklin and J. Feng, "Leveraging transitive relations for crowdsourced joins," in *Proc. of the 2013 ACM SIGMOD Int. Conf. on Management of Data*, pp. 299–240, 2013.

[6] C. Kong, M. Gao, C. Xu, W. Qian and A. Zhou, "Entity matching across multiple heterogeneous data sources," in *Int. Conf. on Database Systems for Advanced Applications*, Cham, Springer, pp. 133–146, 2016.

[7] J. Wang, T. Kraska, M. J. Franklin and J. Feng, "Crowder: Crowdsourcing entity resolution," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1483–1494, 2012.

[8] S. E. Whang, J. McAuley and H. Garcia-Molina, *Compare Me Maybe: Crowd Entity Resolution Interfaces*. California: Stanford InfoLab, 2012.

[9] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli *et al.,* "Corleone: Hands-off crowdsourcing for entity matching," in *Proc. of the 2014 ACM SIGMOD Int. Conf. on Management of Data*, Snowbird Utah, United States, pp. 601–612, 2014.

[10] P. Konda, S. Das, G. C. Paul Suganthan, A. Doan, A. Ardalan *et al.,* "Magellan: Toward building entity matching management systems," *Proceedings of the VLDB Endowment*, vol. 9, no. 12, pp. 1197–1208, 2016.

[11] S. Mudgal, H. Li, T. Rekatsinas, A. Doan, Y. Park *et al.,* "Deep learning for entity matching: A design space exploration," in *Proc. of the 2018 Int. Conf. on Management of Data*, Houston TX, USA, pp. 19–34, 2018.

[12] Y. Li, J. Li, A. Doan and W. Tan, "Deep entity matching with pre-trained language models," *Proceedings of the VLDB Endowment*, vol. 14, no. 1, pp. 50–60, 2020.

[13] V. Sanh, L. Debut, J. Chaumond and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," in *NeurIPS'19 EMC2 Workshop*, Vancouver BC, Canada, 2019.

[14] H. Köpcke and E. Rahm, "Frameworks for entity matching: A comparison," *Data & Knowledge Engineering*, vol. 69, no. 2, pp. 197–210, 2010.
[15] Z. Chen and Y. He, "Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning," in *The World Wide Web Conf.*, San Francisco CA, USA, pp. 2413–2424, 2019.
[16] H. Nie, X. Han, B. He, L. Sun, B. Chen *et al.,* "Deep sequence-to-sequence entity matching for heterogeneous entity resolution," in *Proc. of the 28th ACM Int. Conf. on Information and Knowledge Management*, Beijing, China, pp. 629–638, 2019.
[17] S. Das, G. C. Paul Suganthan, A. Doan, J. F. Naughton, G. Krishnan *et al.,* "Falcon: Scaling up hands-off crowdsourced entity matching to build cloud services," in *Proc. of the 2017 ACM Int. Conf. on Management of Data*, Chicago, Illinois, USA, pp. 1431–1446, 2017.
[18] S. E. Whang, P. Lofgren and H. Garcia-Molina, "Question selection for crowd entity resolution," *Proceedings of the VLDB Endowment*, vol. 6, no. 6, pp. 349–360, 2013.
[19] Y. Lin, H. Wang, J. Li and H. Gao, "Efficient entity resolution on heterogeneous records," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 5, pp. 912–926, 2019.
[20] D. G. Mestre and C. E. S. Pires, "Improving load balancing for mapreduce-based entity matching," in *2013 IEEE Symp. on Computers and Communications (ISCC)*, Split, Croatia, IEEE, pp. 618–624, 2013.
[21] J. Revaud, P. Weinzaepfel, Z. Harchaoui and C. Schmid, "Deepmatching: Hierarchical deformable dense matching," *International Journal of Computer Vision*, vol. 120, no. 3, pp. 300–323, 2016.
[22] H. Wang, X. Ding, J. Li and H. Gao, "Rule-based entity resolution on database with hidden temporal information," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 11, pp. 2199–2212, 2018.
[23] J. Peng, H. Wang, J. Li and H. Gao, "Set-based similarity search for time series," in *Proc. of the 2016 Int. Conf. on Management of Data*, San Francisco, California, USA, pp. 2039–2052, 2016.
[24] H. Abu Ahmad and H. Wang, "Automatic weighted matching rectifying rule discovery for data repairing: Can we discover effective repairing rules automatically from dirty data?," *The VLDB Journal*, vol. 29, no. 6, pp. 1–15, 2020.
[25] Z. Li, H. Wang, W. Shao, J. Li and H. Gao, "Repairing data through regular expressions," *Proceedings of the VLDB Endowment*, vol. 9, no. 5, pp. 432–443, 2016.
[26] H. Ma, M. Alipourlangouri, Y. Wu, F. Chiang and J. Pi, "Ontology-based entity matching in attributed graphs," *Proceedings of the VLDB Endowment*, vol. 12, no. 10, pp. 1195–1207, 2019.
[27] A. A. Ahmed and B. Akay, "A survey and systematic categorization of parallel k-means and fuzzy-c-means algorithms," *Computer Systems Science and Engineering*, vol. 34, no. 5, pp. 259–281, 2019.
[28] A. A. and B. Akay, "Human activity recognition based on parallel approximation kernel k-means algorithm," *Computer Systems Science and Engineering*, vol. 35, no. 6, pp. 441–456, 2020.
[29] C. Cheng and D. Lin, "Image reconstruction based on compressed sensing measurement matrix optimization method," *Journal of Internet of Things*, vol. 2, no. 1, pp. 47–54, 2020.
[30] M. Bilgic, L. Licamele, L. Getoor and B. Shneiderman, "D-dupe: An interactive tool for entity resolution in social networks," in *2006 IEEE Symp. on Visual Analytics Science and Technology*, Baltimore, USA, pp. 43–50, 2006.
[31] L. Kolb, A. Thor and E. Rahm, "Dedoop: Efficient deduplication with hadoop," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 1878–1881, 2012.
[32] M. Dallachiesa, A. Ebaid, A. Eldawy, A. Elmagarmid, I. F. Ilyas *et al.,* "NADEEF: A commodity data cleaning system," in *Proc. of the 2013 ACM SIGMOD Int. Conf. on Management of Data*, New York, USA, pp. 541–552, 2013.
[33] M. Ebraheem, S. Thirumuruganathan, S. Joty, M. Ouzzani and N. Tang, "DeepER–deep entity resolution," *Proceedings of the VLDB Endowment*, vol. 11, no. 11, pp. 1454–1467, 2017.
[34] J. Devlin, M. W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, pp. 4171–4186, 2019.
[35] I. Tenney, D. Das and E. Pavlick, "BERT rediscovers the classical NLP pipeline," in *Proc. of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, pp. 4593–4601, 2019.

[36] C. Sun, X. Qiu, Y. Xu and X. Huang, "How to fine-tune BERT for text classification?," in *China National Conf. on Chinese Computational Linguistics*, Cham, Springer, pp. 194–206, 2019.
[37] A. Zhang, B. Li, W. Wang, S. Wan and W. Chen, "MII: A novel text classification model combining deep active learning with BERT," *Computers, Materials & Continua*, vol. 63, no. 3, pp. 1499–1514, 2020.
[38] H. Xu, B. Liu, L. Shu and P. S. Yu, "BERT post-training for review reading comprehension and aspect-based sentiment analysis," in *Proc. of NAACL-HLT 2019*, Minneapolis, Minnesota, pp. 2324–2335, 2019.
[39] P. Cen, K. X. Zhang and D. S. Zheng, "Sentiment analysis using deep learning," *Journal on Artificial Intelligence*, vol. 2, no. 1, pp. 17–27, 2020.
[40] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proc. of the 2019 Conf. on Empirical Methods in Natural Language Processing and the 9th Int. Joint Conf. on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, pp. 3982–3992, 2019.
[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones *et al.,* "Attention is all you need," in *Proc. of the 31st Int. Conf. on Neural Information Processing Systems (NIPS'17)*, Long Beach, California, USA, pp. 6000–6010, 2017.
[42] Z. Lan, M. Chen, S. Goodman, K. Gimpel and P. Sharma, "AlBERT: A lite BERT for self-supervised learning of language representations," in *ICLR*, Addis Ababa, ETHIOPIA, 2020.
[43] E. Ma, "NLP augmentation," 2019. [Online]. Available: https://github.com/makcedward/nlpaug.
[44] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue *et al.,* "HuggingFace's transformers: State-of-the-art natural language processing," in *Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing: System Demonstrations*, Online, pp. 38–45, 2019.