An Extensible Block Scheme-Based Method for Entity Matching

DI2KG 2020 Challenge Winner Paper

Jiawei Wang[*]
College of Cybersecurity
Jinan University, Guangzhou, China
jiaweiwang.aa@qq.com

Haizhou Ye[*]
College of Cybersecurity
Jinan University, Guangzhou, China
13632259569@163.com

Jianhui Huang[*]
College of Cybersecurity
Jinan University, Guangzhou, China
924285628@qq.com

ABSTRACT

Entity resolution (ER) is of great importance in many real-world applications, including information retrieval, web search, natural language processing, web information integration, multi-source data analysis, etc. In this paper, we present an extensible block scheme-based method for entity resolution. Specifically, at first we preprocess data records from multiple datasets, so as to remove ambiguity and maintain data attributes that are helpful for entity resolution. Then, the processed records are assigned to different blocks, after which similarity between records within a same block is computed. Experiment results on challenge datasets from DI2KG show that our proposed method is promising.

KEYWORDS

Entity Matching, Block-scheme, Block Aggregation, Entity Similarity

1 INTRODUCTION

Entity Resolution (ER) is to match data records from two or more data sources, by analyzing their contents that describe identical entities in real-world [5], and it remains as a challenging problem in data management [9], information retrieval [8], data mining [7], natural language processing, etc. Entity resolution has a wide range of applications, including data cleansing, electronic commerce [6], web search, and public/government data analysis [11].

In this paper, we focus on the problem of matching products recorded in electronic commerce website. Performing the entity matching task on an e-commerce database is challenging, because of 1) product data is highly heterogeneous; 2) structure of the data is loosely defined; 3) there are multiple data sources (for example, in the DI2KG challenge we need to process different data sources from different stores). One of the critical missions of ER on large data sets is how to reduce the computational complexity to improve efficiency. Currently, there are three prominent solutions, i.e., Blocking [10], Windowing [4], and Hybriding [3]. In this paper we use the first one. By assigning each product to one or more blocks and computing the similarity for pairs of products in these blocks, blocking schemes can reduce the number of pair-wise similarities that need to be computed. Due to challenges of ER on multiple source data, we need to solve several problems, as described below:

- How to perform data preprocess and cleansing
- How to design efficient block scheme and block. aggregation method
- How to compute entity similarity within a block, and how to determine transitivity of similarity between entities

Inspired by the previous work of rule-based ER method [2], we develop a scalable ER system framework. Specifically, the core of our framework consists of a block scheme for the data set and similarity computation of the product pairs within a block. The block scheme includes various preprocessing steps, different keyword extraction methods, as well as multiple block scheme aggregators. We solve the ER tasks of DI2KG 2020 Challenges that involve product entities of Monitor and Notebook, by using our block scheme-based framework.

The paper is organized as follows. In Section 1 we give an introduction of entity resolution and the challenge task in DI2KG. We present our framework for entity resolution in Section 2, including the three main components of our framework. In Section 3, we conduct experiments on the challenge datasets to evaluate the effectiveness of our method. Finally, in Section 4 we summarize this paper and give our future work on entity resolution.

2 THE PROPOSED METHOD

Our method is based on block scheme and scheme aggregation. As shown in Figure 1, our extensible entity resolution system framework mainly includes three submodules, i.e., Preprocess, Block Scheme and Scheme Aggregation (BS-SA), Similarity and Clustering. In this section, we described three sub-modules in detail.

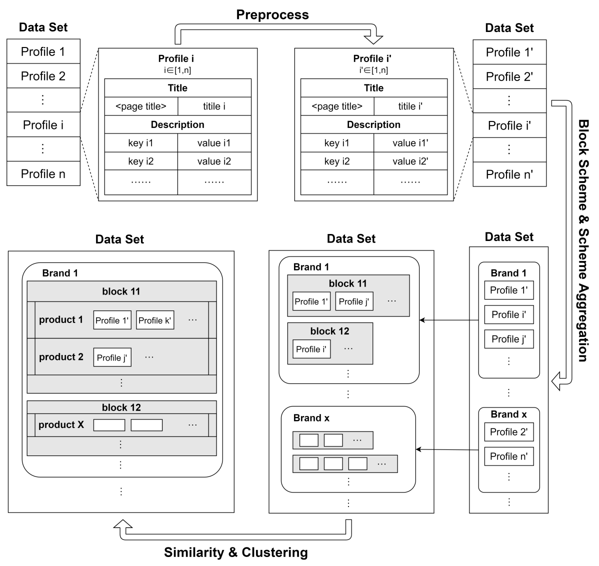

Figure 1: Our extensible entity resolution system framework

[*]All authors contributed equally to this research.

2.1 Data Preprocessing

In data preprocessing step, we remove some less important terms from page titles, and less informative properties from the product description text. Specifically, from table 1 we can see that there are differences between product profiles. For example, the first product possesses some property feature while the second product contains neither this property nor the corresponding value. We also observe that for the same products, their model names are the same. So, we try to find model names for entity matching. Before finding the model name, there are two problems to deal with, as given below:

- Which property contains the model name
- How to deal with the situation that the extracted model words cannot be extracted from the predetermined key value or the extracted model names

Table 1: Example Raw Data

property	value
<page title>	New ELO 1928L 19inch TFT 1280x1024 LCD Non Touch Dual Monitor E939583 7411493021428 | eBay
aspect ratio	5:4
brand	Elo
contrast ratio	1300:1
model	1928L
screen size	19
<page title>	Elo TouchSystems 1928L – MBE
contrast ratio typical	1300:1
display resolution	1280 x 1024 pixels
horizontal scan range	31.5 - 80 kHz
model name	1928L

Table 2: Example Data after Preprocessing

property	value
<page title>	new elo 1928l tft lcd non touch dual monitor e939583 7411493021428 | ebay
brand	elo
model	1928l
<page title>	elo touchsystems 1928l – mbe
model name	1928l

To solve the above problems, at first some preprocessing steps are necessary, such as lowercasing strings and removing some special symbols. Generally, the model names have at least one number and one non-numeric character. Hence, we use regular expression to match the model name. We find that the model name will be obtained precisely in the properties, like model, model name and so on. And we retain these properties as new descriptions of product. When one is unable to find these properties, we consider property '<page title>', because there is part of information which need to be removed, such as resolution and screen size in the '<page title>'. On the other hand, we consider the situation that the model name is separated by spaces, so we will remove the space between non-numeric characters and numeric characters to create a model word (i.e. the model name is something like 'P 230', we will change it to 'p230').

2.2 Block Scheme and Scheme Aggregation

After data preprocessing step, the processed data will be passed to submodule Block Scheme and Scheme Aggregation, through which each product is assigned to a block. Referring to Figure 1, this submodule consists of two parts: first, the products are classified based on the brand list that obtained from the Internet [2] and supplemented and modified manually. Just as [12] said, the brand list provides the algorithm with specific task information. Then, for the products of each brand, we classify products to different blocks, by extracting model word and using the blocks aggregator [13].

In this challenge, the model word is defined as a word that contains at least one number and one non-numeric character, and the entire word is completely extracted as one model word. We adopt two block schemes, i.e., extracting a single model word for title and description, and then aggregating the two schemes by the union aggregator. Eventually, all documents of each brand will be mapped to multiple product entities and each document will only be mapped to one product entity.

2.3 Entity Similarity within Blocks

The method we proposed in this paper for computing entity similarity is based on model words and set judgment. The definition of model word is as described in Section 2.2, and we compare the model words set of product a and the model words set of product b, which measures the similarity of two products. Our proposed similarity function consists of two parts.

The first part evaluates whether the models of product a and b are the same, by using function $diffModel(a, b, M)$. Here, M is a list we constructed from product descriptions, which consists of attributes where the model may appear. We count the model words extracted from the attributes in the list M, and select the model word that appears the most frequently as the product model. Moreover, if there is no any model word, then we take the first model word from '<page title>' as the model. The rationale is that the product model in the title usually appears first.

The second part determines whether synonymous attributes of product a and product b are similar, by calling the function $mwSim(a, b, L_i, O_i)$ multiple times. Here, parameter $L = [L_1, , L_n]$, where $L_i \in L$ contains synonymous attributes, and $O = [O_1, , O_n]$, where operator $Oi \in O$ is either $'and'$ or $'or'$, which is applied to L_i. Each time $mwSim(a, b, L_i, O_i)$ is invoked, for the model word set extracted from all the attributes of the product in list L_i, we determine the relationship between two sets, according to the content of operator O_i. Specifically, if O_i is $'and'$, then the two sets must be the same, otherwise one set must be the subset of the other set. For example, a parameter configuration is shown below:

$$mwSim(a, b, ['< pagetitle >', ' model', ' modelname'], ' or')$$

Algorithm 1 shows a high-level overview of how to compute product similarity. Specifically, if the model names of the two products are the same, the algorithm will compare whether certain attributes of the product are similar according to the parameter L and Q.

```
Algorithm 1 Product Similarity Method
Input: Product profile $a$ and $b$; List $L = [L_1, , L_n]$, where $L_i \in L$
    contains some synonyms attributes; List $O = [O_1, , O_n]$, where
    operator $Oi \in O$ is either $'and'$ or $'or'$, which corresponds to
    $L_i$; Attributes list $M$; Furthermore, the following functions are
    used:
        -$diffModel(a, b)$ is true if models of product $a$ and $b$ are
    different
        -$mwSim(a, b, L_i, O_i)$ the model words similarity between
    the products $i$ and $j$ using the key list $L_i$ and operator $O_i$.
Output: True if product $a$ and product $b$ are the same; otherwise,
    False.
1: function ProdSim($a, b, M, L, O$)
2:     if $diffModel(a, b, M)$ then return False
3:     end if
4:     for all $L_i, O_i$ in $L, O$ do
5:         if not $mwSim(a, b, L_i, O_i)$ then return False
6:         end if
7:     end for
8:     return True
9: end function
```

Algorithm 2 judges whether the two products have the same model. It regards the most frequently occurring model word or the first model word in the title as the model name, and compares the model names of the two products.

Function $mwSim(a, b, L_i, O_i)$ in Algorithm 3 is used to compute the similarity between product description of a and b, according to some special product attributes. The number of times this function is called depends on the length of the parameter L_i and O_i. In this algorithm, the model words from some attributes of product a and b are compared. If O_i is '*and*', the model words must be the same, otherwise, one model word set must be the subset of the other set.

Due to transitivity property between products, if A is similar to B, and B is similar to C, then we regard it that A is similar to C. We perform clustering while calculating the similarity, instead of clustering after calculating the similarity between all products. This may sacrifice accuracy a little bit, but can significantly improve efficiency.

3 EXPERIMENT RESULTS

To evaluate performance of our proposed method, we conduct experiments on the challenge datasets provided by DI2KG. We summarize the best experiment results of our proposed method in Table 3, i.e., on training Dataset Y the Precision, Recall and F-Measure are 1.000, 0.964 and 0.982, respectively, whereas on Dataset X Precision, Recall and F-Measure are 0.915, 0.967 and 0.940, respectively.

In addition, we tested our algorithm on the camera dataset and the Precision, Recall and F-Measure are 0.98, 0.97 and 0.98 [1]. In general, we can see that our algorithm achieved good performance on both data sets, that is, it has good extensivity. In order to be suitable for different datasets, we can set the parameters such as brand list, product descriptions list and synchronous attributes list,

```
Algorithm 2 Model Comparison Method
Input: Product profile $a$ and $b$; Attributes list $M$;
        -$mw(a, M)$ extracts the model words from the key-value
    pair of product a if the key appears in list $M$.
        -$MaxModel(L)$ find the model word appear most fre-
    quently in the model word list $L$.
        -$isEmpty(L)$ true if $L$ is empty.
Output: True if the models of product $a$ and product $b$ are different;
    otherwise, False.
1: function diffModel($a, b, M$)
2:     $discriptionAmw = mw(a, M)$
3:     $discriptionBmw = mw(b, M)$
4:     $titleAmw = mw(a, ['$<page title>$'])$
5:     $titleBmw = mw(b, ['$<page title>$'])$
6:     if $isEmpty(discriptionAMW)$ then
7:         if $titleAmw[0]$ in $titleBmw$ then return False
8:         end if
9:         $modelA = titleAmw[0]$
10:    else
11:        $modelA = MaxModel(discriptionAmw)$
12:    end if
13:    if $isEmpty(discriptionBmw)$ then
14:        if $titleBmw[0]$ in $titleAmw$ then return False
15:        end if
16:        $modelB = titleBmw[0]$
17:    else
18:        $modelB = MaxModel(discriptionBmw)$
19:    end if
20:    return not ($modelA == modelB$)
21: end function
```

```
Algorithm 3 Model Word Similarity Method
Input: Product profile $a$ and $b$; Synonyms attributes list $L_i$; Opera-
    tor $O_i$ is either $'and'$ or $'or'$, which corresponds to $L_i$; $mw(w)$
    extracts the model words for a given set of words $w$.
Output: True if the model words of product $a$ and $b$ extracted from
    $L_i$ are the same under operator $O_i$; otherwise, False.
1: function mwSim($a, b, L_i, O_i$)
2:     $setA, setB = mw(a, L_i), mw(b, L_i)$
3:     if $O_i ==' and'$ then
4:         if $setA.issubset(setB)$ and $setB.issubset(setA)$ then
5:             return True
6:         end if
7:     end if
8:     if $O_i ==' or'$ then
9:         if $setA.issubset(setB)$ or $setB.issubset(setA)$ then
10:            return True
11:        end if
12:    end if
13:    return False
14: end function
```

Table 3: Performance of Our Method on Challenge Datasets

	Precision	Recall	F-Measure
Dataset Y	1.000	0.964	0.982
Dataset X	0.915	0.967	0.940

which can be obtained based on rules or machine learning. It is worth mentioning that our algorithm is mainly based on model words, thus, it is better for data sets that mainly rely on product models for matching, such as cameras, displays and other electronic products.

Our observation is that the key to data preprocessing is to remove some redundant information. Since in Algorithm 2, we try to use the first model word in '<page title>' for comparison, it is highly recommended to make the first model word in the '<page title>' to be the model number of the product.

On the other hand, in the block scheme the acquisition of brand information requires some manual intervention. Although it can be obtained through machine learning or rules-based methods, we found that there are some special circumstances that need to deal with. For instance, Alienware was acquired by Dell, so Alienware and Dell belong to the same brand. Since brand classification is the first step in the BS-SA, manual intervention for such special situations can improve the recall rate at lower cost.

4 CONCLUSION

In this paper, we proposed an extensible block-scheme based method for entity resolution. Specifically, our method consists of three components, i.e., data preprocessing, block-scheme and scheme aggregation, and similarity and clustering. Experiments on DI2KG challenge datasets show that our method can achieve 0.982 and 0.94 F-measure values on Dataset X and Dataset Y, respectively.

In our future work, we intend to combine machine learning techniques and consider weighting key properties, so as to find as many as possible the best <key, value> pairs of products. Because by using those strategies, we can make full use of training datasets, by mapping product similarity to the range [0, 1].

ACKNOWLEDGMENTS

The authors would like to thank the reviewers for their comments. This paper is partially supported by NSFC under grant Number 61972177.

REFERENCES

[1] [n.d.]. ACM SIGMOD 2020 Programming Contest Leaders. http://www.inf.uniroma3.it/db/sigmod2020contest/leaders.html.

[2] [n.d.]. Wikipedia: The free encyclopedia. https://en.wikipedia.org/wiki/Category:Lists_of_consumer_electronics_manufacturers.

[3] Youssef Aassem, Imad Hafidi, and Noureddine Aboutabit. 2020. Enhanced Duplicate Count Strategy: Towards New Algorithms to Improve Duplicate Detection. In *Proceedings of the 3rd International Conference on Networking, Information Systems & Security*. 1–7.

[4] P Christen. [n.d.]. Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection. 2012.

[5] Gianni Costa, Giuseppe Manco, and Riccardo Ortale. 2010. An incremental clustering scheme for data de-duplication. *Data Mining and Knowledge Discovery* 20, 1 (2010), 152.

[6] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F Naughton, Narasimhan Rampalli, Jude Shavlik, and Xiaojin Zhu. 2014. Corleone: hands-off crowdsourcing for entity matching. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 601–612.

[7] Jungo Kasai, Kun Qian, Sairam Gurajada, Yunyao Li, and Lucian Popa. 2019. Low-resource deep entity resolution with transfer and active learning. *arXiv preprint arXiv:1906.08042* (2019).

[8] Selasi Kwashie, Lin Liu, Jixue Liu, Markus Stumptner, Jiuyong Li, and Lujing Yang. 2019. Certus: an effective entity resolution approach with graph differential dependencies (GDDs). *Proceedings of the VLDB Endowment* 12, 6 (2019), 653–666.

[9] Guoliang Li, Jiannan Wang, Yudian Zheng, and Michael J Franklin. 2016. Crowdsourced data management: A survey. *IEEE Transactions on Knowledge and Data Engineering* 28, 9 (2016), 2296–2319.

[10] George Papadakis, Ekaterini Ioannou, Themis Palpanas, Claudia Niederee, and Wolfgang Nejdl. 2012. A blocking framework for entity resolution in highly heterogeneous information spaces. *IEEE Transactions on Knowledge and Data Engineering* 25, 12 (2012), 2665–2682.

[11] Tomer Sagi, Avigdor Gal, Omer Barkol, Ruth Bergman, and Alexander Avram. 2017. Multi-source uncertain entity resolution: Transforming holocaust victim reports into people. *Information Systems* 65 (2017), 124–136.

[12] Ronald Van Bezu, Sjoerd Borst, Rick Rijkse, Jim Verhagen, Damir Vandic, and Flavius Frasincar. 2015. Multi-component similarity method for web product duplicate detection. In *Proceedings of the 30th annual ACM symposium on applied computing*. 761–768.

[13] Damir Vandic, Flavius Frasincar, Uzay Kaymak, and Mark Riezebos. 2020. Scalable entity resolution for Web product descriptions. *Information Fusion* 53 (2020), 103–111.