Entity Matching Using Different Level Similarity for Different Attributes

Guochao Song
School of Computer Science
Beijing University 0/Posts and Telecommunications
Beijing.China
sgc@bupt.edu.cn

Lei Zhang and Pengfei Wang
School of Computer Science
Beijing University 0/Posts and Telecommunications
Beijing.China
{zlei&wangpengfei} @bupt.edu.cn

Abstract—**Entity** matching (EM) recognizes records in one or different databases that represent the same entity in real-world. It is an essential part of data cleaning and data integration. Most existing studies determine whether two records match by calculating the similarity of the corresponding attribute values in the two records. Then series of similarity metrics are proposed, But according to our investigation, we found that no one has considered combining semantic level similarity with string level similarity. In some data tables, some attributes values are long text, such as product descrlptlons, and they are more semantically similar in different records that refer to a same entity. Other numeric or noun attributes, such as prlce and person name, are more suitable for string level similarity calculations. Therefore, we propose a model to calculate the similarity of the two types of attributes using modules of different similarity levels, and assign them different weights. Learning by labeled data, we get a model that can effectively solve the entity matching task ,

Keywords-entity matching; database; string similarity; semantic similarity; data integration;

I. INTRODUCTION

Entity matching (also named as duplicate detection, entity resolution, record linkage) finds records referring to the same real-world entity in the databases. It is an essential part of data cleaning and data integration. Different representations of the same entity may result from typographical errors, misspellings, abbreviations, as well as integration of multiple data sources. Properly merging these records and the information they represent is a critical step in generating sufficient quality data for mining. As a result, extensive research on entity matching has been made, by machine learning[I,2] and rule-based[3,4] approaches mainly. Although much progress has been made, a better solution has yet to be proposed.

Matching a record with multiple attributes can be resolved by matching each of their attribute values separately. A number of studies provide a variety of string distance metrics for attribute values matching[5]. Each of them works well for particular types of error. But they are all considered on the character level or string level. It can not capture the similarity in semantics. A recent work through word embedding get distribute representations of records to capture the semantic similarity of attribute values[6], but some attribute such as price, person's name, location's name, etc, have no strong semantics. All of these work ignore a problem that attribute in different types should be treat on different level. Such as, for a descriptive attribute of an entity, it will be a long text. Even if different records are misspelled or come from different data sources, they have similar semantics. This type of attribute applies to semantic level similarity. But for some numeric or noun attributes, their semantics are not strong, such as price, person's name, location's name, etc. These attributes compute in string level or character level preferred.

So we propose an entity matching model that contains two categories modules. One module is used to calculate the semantic similarity of the textual attributes values, and another module is used to calculate the string similarity of numeric or noun attributes values. Each module will be assigned a weight which can be learn through train data. Finally, we will complete the entity matching task based on the weighted sum of the output of each module.

Some studies on deep learning have found that neural networks can solve text matching problems to some extent[7,8]. Liang Pang[9] proposed a new method for text matching in a way similar to image recognition. They utilized a convolutional neural network to capture word, phrase and sentence level matching signals. It achieved good results. In this paper, we use this method to calculate the similarity in semantic level and a variant of eidt distance to calculate the similarity in string level.

The rest of this paper is organized as follows. Section II will introduce the related work of entity matching. Section III will introduce the specific details of our method. The experimental results will be discussed in the section N. In the last part, we will give a brief summary of this work and talk about the future work.

II. RELATED WORK

The notion of Entity Matching(EM) was firstly proposed by Newcombe as record linkage to identify medical records of the same patient from different periods[lO]. Then the EM has been researched on the term of duplicate detection, entity resolution, object identification, merge/purge and others. There is a good overview in surveys such as [5,11]. Existing research can be roughly classified as based on machine learning[12,13], rules and crowd based[14]. Fellegi and Sunter [1] developed a formal theory and offered statistical methods. Most of the machine learning approaches are improvements of it.

978-1-5386-6565-7/18/$31.00 ©2018 IEEE

Many works treat the EM problem as a binary classification problem and then compute the probability of each class, which in this case is match or no-match. Generally, they calculate the similarity of each corresponding attribute of two entities, then train a classifier with this similarity vector, or set a threshold, compare the similarity of each attribute with this threshold to determine whether the two entities are match. Many similarity metrics are proposed[15].

Some rule-based methods worked well in some specific domam. Experts in that domain develop a series of matching rules to match the data. Although this kind of method can achieve good results in their field, it requires experts in the corresponding field to develop these rules, which consumes a lot human efforts. Singh R[16] proposed a way to automatically select matching functions and thresholds by learning and generate an explanation which makes the rules interpretable.

III. METHODOLOGY

The process to solve the entity matching problem is as follow. First we use the labeled pairs for model training. The data to be matched is taken from the database and constructed into data pairs. Then pass the pairs into the model to calculate the matching result. The details of our model will be discussed in the following sections.

A. *Problem definition*

Let R = {n, rz, ... , rn} be a relational table or a relational table composited by two table from different databases with corresponding attributes aligned. Assuming it has n records and m attributes A_1, A_2, ... ,Am We describe the value of attribute A_i of record rj as rj[A_i]. The problem of entity matching is, given all distinct record pairs (n, rj) constructed from R where n ≠ rj, determining which pairs of records refer to the same real-word entities.

We treat it as a binary classification problem, in which the classes are match or non-match. Our model has two categories modules, The one which is used for calculating the semantic level similarity of the attributes values is called semantic-level module. The other one which is used for calculating the string level similarity of the attributes values is called string-level module. For a pair (n, rj), we input each of its long textual attribute values ri[Am] and rj[Am] into the semantic-level module to get the semantic similarity of Am in r_i and rj, and input each of its numerical or noun attribute values r_i[An] and rj[An] into the stnng-level module to get the string similarity of An, and each module outputs two scores. Each category module shares the same weight, and the output of the two categories of modules are weighted to determine whether the two records match.

B. *Semantic-level module*

We. use the model proposed in Pang Liang[9] as the semantic-level module to calculate the similarity of long text attributes. They converted two input texts into a matching matrix M, with each element Mij represents the similarity of word a, and bj (a, and bj is the i-th and j-th word in the two input text). In our work, we use the cosine similarity to denote the similarity of each word. Look up the embedding of each word $\vec{a}_i = \varphi(a_i)$ and $\vec{b}_j = \varphi(b_j)$, from the Word2Vec, the Mjj can be denoted as:

$$M_{ij} = \frac{\vec{a}_i^{\mathrm{T}}\vec{b}_j}{\|a_i\|\cdot\|b_j\|} \tag{1}$$

We added some processing flow when constructing the matching matrix according to our own needs. In entity matching scenario, duplicate entity records may raise from typographical errors, misspellings, abbreviations. These words usually do not appear in the dictionary of Word2Vec. This problem is generally called OOV and usually these words will be replaced by an UNK which has been trained as a word. However, in the context of entity matching, we should try to avoid errors caused by typographical errors, misspellings, etc. So we did some of our own processing when constructing the matching matrix. If aj or bj is out of vocabulary, calculate the ratio of the edit distance of the two words to the maximum of the length of the two words. If it is less than a threshold λ, the word out of vocabulary is considered to be a misspelling of another word. At this time, M_{ij} is set to 1, otherwise it is set to O. Let ϕ (ai, bj) be the function to determine whether a, and bj are the same string when one of them is out of vocabulary at least. So it can be denoted as (2):

$$\phi(a_i,b_j) = \frac{Ed(a_i,b_j)}{Max(length(a_i),length(b_j))} \tag{2}$$

where Ed(ai, b_j) denotes the edit distance of word a, and bj, and length/a) denotes the length of a_i. Algorithm 1 shows the pseudocode for our approach of constructing the matching matrix.

Algorithm 1 matching matrix construct

```
Input: rj[Ak], rj [Ak]
Output: matching matrix M of pairtr; rj) in $A_k$
begin
  for m in range(length(rj[Ak]» do
      for n in range(length(rj[Ak])) do
          Look up vector for ri[Ak][m] and rj [Ak][n]
          ifri[ Ak][m] or rj [Ak][n] out ofvocabul ary
             if $\phi$ (ri[Ak][m] ,rj [Ak][n]) < $\lambda$
                  $M_{mn}$ = I
             else $M_{mn}$ = 0
          else $M_{mn}$ = cosine(V(r,[Ak][m]),V(rj[Ak][m]))
      end for
  end for
end begin
```

And then the matching matrix will be treat as a image input to a neural network as mentioned in that work. By CNN, square kernel, ReLU as the active function and Dynamic pooling strategy is used to deal with the text length, the network captures the matching signals in word level, phrase level and sentence level and get a 2-dimensional matching score vector:

$$(s_0, s_1)^{\mathrm{T}} = W_2\sigma(W_1 z + b_1) + b_2 \quad (3)$$

So and si are the scores of match and non-match and WI, W_z, b., b_2 are parameters in the network which can be learned when training. σ is the activation function. In our work we will compute scores po and PI from So, si by softrnax function, and po, pi will be passed to the entire network as the output of this module.

C. *String-level module*

Numeric or noun attributes contains weak semantic information We think they are more suitable for the similarity rnetrics at the string level. In this work, we use a variant of edit distance to compute the similarity in string level.

Edit Distance is a dynamic programming algorithin used to calculate the similarity of two strings. It refers to the minimum number of edit operations required for one string transforming to the other. Licensed editing operations including replacing one character with another, inserting one character, and deleting one character, each operation cost 1. We think that when the edit distance of two strings is fixed, the longer the length of the string, the higher the similarity between the two strings. For example, "restaurant" and "restaurent", "cat" and “cut”, their editing distance is the same, 1 ,due to replacing by one character, but obviously the former is more similar than the latter. So we compute the similarity of two strings by the ratio of the edit distance of the two strings and the maximum value of the length of the two strings, as in (2). Obviously, this value is between and 1. We use $Po = 1-\phi(r_i[A_k], r_j[A_k])$ as the match score and $p, = \phi(r_i[A_{k]}, r_j[A_k$D as the non-match score of the attribute Akof record pairs.

D. *Model architecture*

In the previous two sections, we have described our semantic-level module and string-levels module. At this section we will introduce how these two categories module works in the entire process.

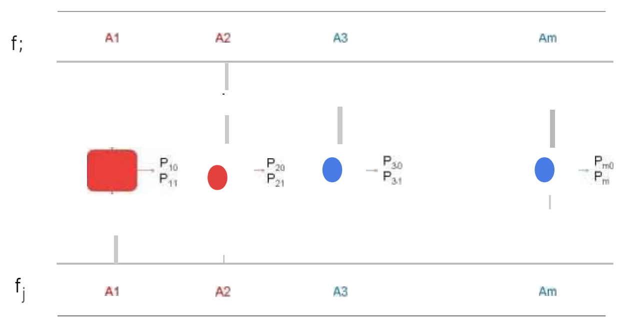

Figure 1. An illustration of our entity matching model

As you can see at Fig. 1, the A in red indicates the long textual attributes, and the A in blue indicates the numeric and noun attributes. The red square and the blue square respectively represent the semantic-level module and the string-level module, each module outputs two values po and p, which respectively represent the score of match or mismatch of the corresponding attributes.

Suppose a record has m attributes, n of which are textual, and m-n are non-textual. Note that for a fixed table, Both m and n are constants. We average the scores of the two types of attributes and assign corresponding weights W. This weight is used during the training to learn the importance of the similarity of textual attributes and non-textual attributes. We will get two scores:

$$S, = \mu(W) * \left(\frac{\sum_{i\in red} p_{ik}}{m}\right) + (1-\mu(W)) * \left(\frac{\sum_{j\in blue} p_{jk}}{n-m}\right), k = 0,1 \quad (4)$$

where μ is a sigmoid function used for limitting the weight between and 1. The So and S, are the scores of each class, a softmax fuction is used for getting the probability of belonging to each class and cross entropy is utilized as the loss function for training, then the optimization becomes to minimize (5):

$$loss = -\sum_{;=1}^{N}\left[y^{(i)}\log(P|(i») + (1-y^{(i)})\,\mathrm{loge}\, p_0^{(i)})\right] \quad (5)$$

$$p_k = \frac{e^{S_k}}{e^{S_0} + e^{S_1}}, k = 0,1$$

where y(i) is the label of the i-th training data.

IV EXPERIMENT RESULTS

In this section, we will show our experimental results. Firstly, we will introduce the datasets we used for evaluating our method. Then we will show the results of our method on F-measure and compare it with some existing methods.

A. *Datasets*

We experimented on three datasets that described in table I . They are all benchinark datasets that have been evaluated by many works using both machine learning(ML) and non-ML based method.

TABLE!. DESCRIPTION OF DATASETS

Dataset	Records	Matchs
Amazon-Google	1363-3226	1300
Abt-Buy	1081-1092	1097
DBLP-Scholar	2616-64263	5347

The first two datasets contain unstructured attributes which is long text such as product description, and more noisy. They are the target data types that our method apply to. The third dataset is structured, whose attributes is almost numeric or noun, but has a textual attribute “title” that we can compute its similarity by semantic-level module. We use it to verify the compatibility of our methods on structured data.

B. Experiment setup

Firstly, we construct the train data and test data. We obtain the non-match pairs by picking one record r, from match pairs (r_i,r_j) and randomly picking another record rk from the relation table excluding records that match the ri. The text matching part is configured as mentioned in their paper. The threshold λ in the semantic-level module is set to 0.3 finally. We use K-fold cross validation with K = 5 and match to non-match pairs ratio of 1:100. We report the average of the F-measure values obtained across all the folds.

C. *Competitor methods*

We get the evaluation data from a paper of approaches evaluation on EM[17]. We compare our method with the following methods that mentioned in that paper.

- Fellegi[1]: The method has two thresholds that can be adjusted, the lower and the upper. Record pairs whose similarity is above the upper are considered as matchs, and whose similarity is below the lower are considered as non-matchs. In that paper, the two thresholds was set equally.

- FEBRL[18]: It uses a support vector machine (SVM) to learn combination of match method.

- MARLIN[2]: In that paper, they offer two method about MARLIN, where SVM and decision tree was used to do the matching task. We only choose the one using SVM who get a better effect here for comparison.

- Magellan[19]: A EM system provides tools to help users covering the entire pipline. which is the state of the art.

TABLE II. EVALUATION OF THE RESULTS(F-MEASURE IN %)

Dataset	**Fellegi**	FEBRL	MARLIN	Mazetlan	Mine
Amazon-GoogIe	53.8	60.1	59.9	87.6	80.6
Abt-Buv	44.5	71.3	70.8	83.8	79.8
DBLP-Scholar	81.9	87.6	89.4	97.6	81.2

"Mine" represents our method in this paper, we can see from the experiment results that as we suppose,our method achieved good results in data with long text attributes. Because of the string distance metric used in string-level module is flawed, our method has a certain gap with state-of-the-art. But we have also made some progress compared to traditional methods. In relatively more structured data, we achieved acceptable results. This shows that our method can be applied to various data scenarios to some extent.

V. CONCLUSION

In this paper, we propose a new method for entity matching. Combining the existing text matching model, we can better complete the entity matching task by using the semantic level and the string level similarity calculation for different types of attribute values.Experiment results show that our method is effective. However, our experiment also found some problems. Our string-level similarity measure is flawed. It does not reflect the string-level similarity of the attribute very well. In addition, our current model only divides the attributes into two classes, semantic level and string level, giving two weight parameters. But this may ignore the role of individual attribute in the same class, so we will also try to assign different weight to each attribute to learn the importance of different attributes for distinguishing entity pairs. How to normalize these weights is a problem we have to consider.

REFERENCES

[1] Fellegi I P, Sunter A B. A Theory for Record Linkage[J]. Publications of the American Statistical Association, 1969, 64(328):1183-1210.

[2] Bilenko M, Mooney R 1. Adaptive duplicate detection using leamable string similarity measures[C]11 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2003:39-48.

[3] Singh R, Meduri V V, Elmagarmid A, et al. Synthesizing entity matching rules by examples[J]. Proceedings of the VLDB Endowment, 2017,11 (2): 189-202.

[4] Wang J, Li G, Yu J X, et al. Entity matching: how similar is similar[J]. Proceedings of the Vldb Endowment, 2011, 4(10):622-633.

[5] Elmagarmid, Ahmed K, Ipeirotis, Panagiotis G, Verykios, Vassilios S. Duplicate Record Detection: A Survey[J]. IEEE Transactions on Knowledge and Data Engineering, 2006, 19(1):1-16.

[6] Ebraheem M, Thirumuruganathan S, Joty S, et al. DeepER -- Deep Entity Resolution[J]. 2017.

[7] Li H, Xu J. Semantic Matching in Search[J]. Foundations & Trends in Information Retrieval, 2014, 7(5):343-469.

[8] Socher R, Huang E H, Pennin J, et al. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection[C]11Advances in neural information processing systems. 2011: 801-809.

[9] Pang L, Lan Y, Guo J, et al. Text Matching as Image Recognition[J]. 2016.

[10] Newcombe H B, Kennedy J M, Axford S J, et al. Automatic Linkage of Vital Records[J]. Science, 1959, 130(3381):954-959.

[11] Kopcke H, Rahm E. Frameworks for entity matching: A comparison[J]. Data & Knowledge Engineering, 2010, 69(2):197-210.

[12] Cohen W W, Richman J. Leaming to match and cluster large high-dimensional data sets for data integration[C]11 Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2002:475-480.

[13] Sarawagi S, Bhamidipaty A. Interactive deduplication using active leaming[C]11 Proc. ACM SIGKDD Conference on Knowledge Discovery and Data Mining. 2002:269-278.

[14] Gokhale C, Das S, Doan A H, et al. Corleone: hands-off crowdsourcing for entity matching[J]. 2014:601-612.

[15] Cohen W, Ravikumar P, Fienberg S. A comparison of string metrics for matching names and records[C]//Kdd workshop on data cleaning and object consolidation. 2003, 3: 73-78.

[16] Singh R, Meduri V, Elmagarmid A, et al. Generating concise entity matching rules[C]//Proceedings of the 2017 ACM International Conference on Management of Data. ACM, 2017: 1635-1638.

[17] Thor A, Rahm E. Evaluation of entity resolution approaches on real-world match problems[M]. VLDB Endowment, 2010.

[18] Christen P. Febrl: a freely available record linkage system with a graphical user interface[C]IIProceedings of the second Australasian workshop on Health data and knowledge management-Volume 80. Australian Computer Society, Inc., 2008: 17-25.

[19] Konda P, Das S, Suganthan GC P, et al. Magellan: Toward building entity matching management systems[J]. Proceedings of the VLDB Endowment, 2016, 9(12): 1197-1208.