Ranking Feature for Classifier-based Instance Matching

Khai Nguyen[1,2,3] and Ryutaro Ichise[1,2]

[1] The Graduate University for Advanced Studies, Japan
[2] National Institute of Informatics, Japan
{nhkhai,ichise}@nii.ac.jp
[3] University of Science, VNU-HCMC, Vietnam
✉ nhkhai@fit.hcmus.edu.vn

Abstract. Instance matching is the problem of finding the instances that describe the same object. It can be viewed as a classification problem, where a pair of two instances is predicted as match or non-match. A common limitation of existing classifier-based matching systems is the absence of instance pairs ranking. We propose using a ranking feature to enhance the classifier in instance matching. Experiments on real datasets confirm the significant improvement when applying our method.

Keywords: instance matching, classification, ranking.

1 Introduction

Instance matching detects the instances describing the same object in two different repositories, R_S and R_T. This task can be considered as a classification problem, in which each example represents a feature vector consisting of the correlation indicators (e.g., literal similarities) of two instances [3, 6, 7]. For training data, each example is also associated with a class, which is either matched (i.e., positive) or non-matched (i.e., negative). The matching task on a new example is to predict its actual class.

In instance matching, an important technique is blocking, which groups the potentially matched instances into the same block [1]. For example, a simple method is to group the instances sharing a number of tokens. By avoiding the huge $|R_S \times R_T|$ pairwise comparisons, blocking reduces the complexity of the matching process.

The ranking is important because of the different ambiguity between the blocks. For example, the block of 'Smith' is much larger than 'Obama' and thus, is more ambiguous. Discriminating the matched and non-matched for such blocks should be based on their local characteristics. In other words, it is better to use different discrimination strategies for different blocks instead of a single strategy for all blocks. A ranking algorithm (e.g., stable matching [4, 5], bipartite graph matching [2], and max-delta [2]) is among possible solutions because it considers only the most confident pairs in each block as positive.

Traditional classifiers discriminate the positive and the negative based on a global boundary drawn from all training examples collected from all blocks. The disadvantage of this approach is the local characteristics of each block is ignored. Therefore, it is ineffective because the blocks are naturally heterogeneous in terms of ambiguity.

We propose to reflect the ranking value of an example (i.e., the vector representing the correlations between two instances) as a feature. For each example, a ranking feature is computed using the example itself and all the related examples, which are drawn from the same block. This ranking feature is added to the original vector as an extra element. As a result, the ranking value is included in the final feature vector. Finally, the classifiers take the final vectors as the input.

2 The ranking feature and optimization

2.1 Feature design

The general idea of the ranking feature is to capture the preference of an example against all the related examples. Let $\mathcal{Q}$ be a collection of examples drawn from a block. The ranking feature $r(x, \mathcal{Q})$ of an example x of $\mathcal{Q}$ is defined as follows.

$$r(x, \mathcal{Q}) = \frac{1}{|\mathcal{Q}|} \sum_{t \in \mathcal{Q}} (f(x) - f(t)) \tag{1}$$

$$f(x) = \frac{1}{1 + e^{-wx}}$$

where f is the logistic function and w is the weight vector. Each element of w assigns the impact of a feature in the example x. w is optimized by a learning algorithm using the training data. An example may exist in different blocks. That is, the ranking feature of an example changes accordingly to the block of interest. Higher value of r indicates a better rank of an example in a block.

The logistic function is widely used in classification and regression because it has a good ability to normalize the input, thus, it is useful in analyses. Furthermore, logistic function is convex and easy to be optimized.

2.2 Optimization

The goal of optimization is to find the vector w based on the training data. The optimality of the ranking feature is to maximize the $r(x, \mathcal{Q})$ (Eq. 1) if x is a positive example and to minimize this value otherwise. The optimization for w also reflects this expectation.

Let $\mathcal{X}$ be a set of training examples and is divided into different groups: $\mathcal{X} = \{\mathcal{Q}_1, \mathcal{Q}_2, ...\}$. Each group $\mathcal{Q}_i$ is respective to the block i, from which the examples of $\mathcal{Q}_i$ are drawn. The optimization of w is done by minimizing the loss $L(w, \mathcal{X})$, which is defined as follows.

$$L(w, \mathcal{X}) = \sum_{\mathcal{Q}_i \in \mathcal{X}} \sum_{(x,y) \in \mathcal{Q}_i \times \mathcal{Q}_i} -(\ell(x) - \ell(y)) \times (f(x) - f(y)) + \frac{1}{2}\lambda ||w||^2 \quad (2)$$

where λ is the regularization parameter, which can be determined by using validation data. ℓ is the class indicator. $\ell(z) = 1$ if z is positive, otherwise $\ell(z) = 0$. The intuition of ℓ is to take only the preference of the examples of different classes.

3 Experiment

We use eight datasets for the experiment. Each dataset contains two repositories with different properties and a collection of matched instances. We apply the property alignment, blocking, and similarity estimation modules of cLink [5] to generate the examples. The property alignment creates the property mappings between two repositories (e.g., name and label) using an overlap metric on the values of the properties. The blocking generates the blocks of instances by using token-based comparison. Two instances are placed in the same block if they share at least one token. One pair may exist in different blocks. Each of such pairs is represented by multiple examples with different ranking features. The similarity estimation computes the feature vector for each instance pair. Each element of a vector is the result of applying a similarity metric on a property mapping. That is, multiple metrics can be applied for the same mapping. For strings, we use exact matching, TF-IDF, Levenshtein, Jaro-Winkler, and Jaccard. For numbers, we use reversed difference. For date-times and URIs, we use exact matching. The summary of the datasets are in Table 1.

We compare the performance of the classifier when using and not using the proposed ranking feature. We apply 5 folds cross-validation. We separate the training set into two parts: 80% and 20%. The former is for minimizing $L(w, \mathcal{X})$ (Eq. 2) and the latter is for optimizing λ parameter. The hyperparameters of the classifiers are not tuned. We split the examples based on the blocks so that the separated sets are independent. Table 2 reports the F1 scores of 4 classifiers: Logistic regression (LR), J48 decision tree, Random Forest (RF), and

Table 1. Summary of datasets.

ID	Name	#Pairwise	#Examples	#Blocks	#Features	#Positive
D1	DBLP-ACM	6.00×10^6	344,587	7,083	37	2,224
D2	ABT-Buy	1.18×10^6	72,185	2,057	23	1,097
D3	Amazon-GoogleProduct	4.40×10^6	100,672	8,413	27	1,300
D4	Sider-Drugbank	52.57×10^6	6,120	1,894	59	1,142
D5	Sider-Diseasome	21.76×10^6	4,227	604	31	344
D6	Sider-DailyMed	26.71×10^6	6,440	1,463	31	3,225
D7	Sider-DBpedia	11.17×10^9	931,701	6,290	58	1,449
D8	Dailymed-DBpedia	41.84×10^9	973,019	5,465	114	2,454

Table 2. F1 scores of using ranking (marked with '*') and not using ranking

ID	LR	LR*	J48	J48*	RF	RF*	SVM	SVM*
D1	0.9702	*0.9774*	0.9736	*0.9866*	0.9850	***0.9886***	0.9652	*0.9748*
D2	0.5956	*0.5980*	0.5766	*0.6028*	0.6470	***0.6736***	0.5410	*0.6024*
D3	*0.5508*	0.5224	0.5252	*0.5682*	0.5766	***0.6380***	0.4204	*0.5114*
D4	0.9480	*0.9710*	0.9564	*0.9740*	0.9704	***0.9810***	0.9486	*0.9734*
D5	*0.9148*	0.9114	0.9036	*0.9076*	0.9472	***0.9472***	0.9436	*0.9436*
D6	*0.9548*	0.9536	0.9780	*0.9854*	0.9804	***0.9902***	0.9578	*0.9586*
D7	*0.7104*	0.7046	***0.7258***	0.7232	0.6886	*0.6994*	0.7096	*0.7156*
D8	0.7934	*0.7968*	0.8258	*0.8398*	0.8562	***0.8638***	0.7930	*0.8094*

Support Vector Machine (SVM). In this table, the result of using ranking feature is marked with '*'. The italic and bold numbers indicate the best result in the context of same classifier and dataset, respectively. According to this table, Random Forest achieves the best result. Furthermore, using ranking feature enhances the performance in 26 out of 32 tests (81%). A paired t-test also reveals that the improvement is statistically significant (p=0.0012). Such result confirms the effective role of ranking factor in classifier-based instance matching.

4 Conclusion

The experiment result shows that our proposed feature is promising. For further research, we are interested in two directions. The first one is to train the models of the ranking feature and the classifier simultaneously. The second one is to model any ranking algorithm so that it can be combined with classifiers.

References

[1] Christen, P.: A survey of indexing techniques for scalable record linkage and deduplication. IEEE Trans. on Knowledge and Data Engineering 24 (9). pp. 1537–1555. (2012)
[2] Gal, A., Roitman, H., Sagi, T.: From diversity-based prediction to better ontology & schema matching. In: WWW'2016. pp. 1145–1155. (2016)
[3] Kejriwal, M., Miranker, D.P.: Semi-supervised instance matching using boosted classifiers. In: ESWC'2015. pp. 388–402. (2015)
[4] Ngomo, A.C.N., Lehmann, J., Auer, S., Höffner, K.: RAVEN - Active learning of link specifications. In: OM'2011. pp. 25–36. (2011)
[5] Nguyen, K., Ichise, R.: Linked data entity resolution system enhanced by configuration learning algorithm. IEICE Trans. on Information and Systems 99(6). pp. 1521–1530. (2016)
[6] Nguyen, K., Ichise, R., Le, H.B.: Learning approach for domain-independent linked data instance matching. In: MDS'2012. pp. 7–15. (2012)
[7] Rong, S., Niu, X., Xiang, W.E., Wang, H., Yang, Q., Yu, Y.: A machine learning approach for instance matching based on similarity metrics. In: ISWC'2012. pp. 460–475. (2012)