# A Heuristic Approach for Configuration Learning of Supervised Instance Matching

Khai Nguyen and Ryutaro Ichise

The Graduate University for Advanced Studies, Japan
National Institute of Informatics, Japan
{nhkhai,ichise}@nii.ac.jp

**Abstract.** Instance matching is the problem of finding instances that co-describe the same topic. The performance of an instance matching system relies on the specified configuration, which is optimally designed for each particular dataset. We propose *cLearn*, a heuristic-based algorithm that automatically searches for the most appropriate matching configuration for the given repositories. *cLearn* is effective when being tested on OAEI benchmarks and outperforms previous supervised systems.

**Keywords:** supervised, instance matching, heuristic search, configuration learning.

## 1 Background

Finding co-reference instances is an important problem of knowledge discovery and data mining. It has a large application in data integration and linked data publication. Many approaches have been proposed for instance matching. Among them, supervised instance matching has been revealed as the most accurate approach [8, 1, 3]. Meanwhile, configuration-based matching [3, 4, 6, 5] attracts most studies because of its advantages in scalability and interpretation. This approach considers the matching score of instances to predict whether they are co-referent or not. The score computation and other settings of instance matching process are specified by a matching configuration, which can be optimized by a learning algorithm. Configuration learning using genetic algorithm has been a research topic of some studies [3, 6, 4]. The limitation of genetic algorithm is that it costs many iterations for reaching the convergence. We propose *cLearn* as a heuristic algorithm that is effective and more efficient. *cLearn* can be used to enhance the performance of any configuration-based instance matching system.

Fig. 1 illustrates the general architecture of supervised and configuration-based instance matching. Given two input repositories, $R_{source}$ and $R_{target}$, the property alignment generator creates the property alignments which are expected to describe the same attribute. The similarity function generator uses the property alignments to create a list of initial similarity functions. A similarity function is specified by two pieces of information: a property alignment $[p_{source}, p_{target}]$ and a similarity measure $S$ (e.g. Levenshtein, Jaro-Winkler, and TF-IDF). For two instances: $x$ and $y$, a similarity function calculates the similarity $S(value(x, p_{source}), value(y, p_{target}))$ where $value(a, p)$ extracts the attribute

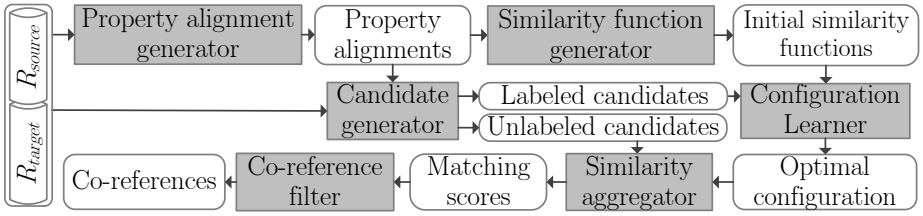


**Fig. 1.** The general architecture of configuration-based supervised entity resolution systems.

value declared by $p$ of $a$. The candidate generator reduces the huge number of pairwise alignments between instances of input repositories by selecting only pairs of potentially co-referent instances [7]. Such pairs are called candidates. In supervised instance matching, annotation step is applied to candidate set in order to obtain a labeled set. Labeled candidates is then provided for the configuration learner. The similarity aggregator computes the final matching score for the unlabeled candidates, using the similarity functions and aggregation function (e.g., linear, quadratic, and boolean) produced by the configuration learner. The final component, co-reference filter, applies some constraints (e.g. stable matching) on the matching scores to construct the final co-references [5].

## 2 The *cLearn* algorithm

A configuration specifies the combination of similarity functions $F_{sim}$, the similarity aggregator $Agg$, and the acceptance threshold $\delta$ of the co-reference filter. Given series of initial similarity functions $I_{sim}$ and similarity aggregators $I_{agg}$, and the labeled candidates, the mission of *cLearn* is to select the optimal $F_{sim}$, $Agg$, and $\delta$.

The pseudo code of *cLearn* is written in Algorithm 1[1]. Labeled candidates are divided into training set $T$ and validation set $V$ before inputting into *cLearn*. $Init$ creates a configuration by assigning $Agg$, $F_{sim}$, and $\delta$ with given values. $Match$ executes the similarity aggregator and co-reference filter in order to obtain the detected co-referent instances. $FindThreshold$ automatically assigns a value to $\delta$. This function first selects the top $|T^+|$ candidates with highest matching score, where $|T^+|$ is the number of the actual co-references (positive candidates) in $T$. Then, it assigns the lowest score of the correctly detected co-reference to $\delta$. $Evaluate$ computes the performance of entity resolution by comparing the generated results with the labeled data. In *cLearn*, $F1$ score is used as the default performance metric because $F1$ it is the main expectation of general instance matching task. The metric can be easily changed (e.g, precision and recall) to adapt with the objectives of particular task.

*cLearn* begins with the consideration of each single similarity function and then checks their combinations. This algorithm works with two underlying heuristics. The first one is the limitation of the number of single similarity functions to $K_{top}$ (line 10), which is set to 16 by default. The second one is the direct enhancement assumption (line 20). The performance of using a new combination

[1] In this pseudo code, we use dot ('.') notation to indicate the member accessor.

**Algorithm 1:** *cLearn*

**Input**: Training set $T$, validation set $V$, integer paramerter $K_{top}$
list of similarity functions $I_{sim}$, list of similarity aggregators $I_{agg}$
**Output**: Optimal configuration $C_{opt}$

$C_{agg} \leftarrow \emptyset$
**foreach** $A \in I_{agg}$ **do**
  $visited \leftarrow \emptyset$
  **foreach** $sim \in I_{sim}$ **do**
    $c \leftarrow Init(Agg \leftarrow A, F_{sim} \leftarrow sim, \delta \leftarrow 0)$
    $links \leftarrow Match(c, T)$
    $c.\delta \leftarrow FindThreshold(links, T)$
    $F1 \leftarrow Evaluate(links, T)$
    $visited \leftarrow visited \cup \{[c, F1]\}$
  $candidate \leftarrow TopHighestF1(visited, K_{top})$
  **while** $candidate \neq \emptyset$ **do**
    $next \leftarrow \emptyset$
    **foreach** $g \in candidate$ **do**
      **foreach** $h \in candidate$ **do**
        $c \leftarrow Init(Agg \leftarrow A, F_{sim} \leftarrow g.c.F_{sim} \cup h.c.F_{sim}, \delta \leftarrow 0)$
        $links \leftarrow Match(c, T)$
        $c.\delta \leftarrow FindThreshold(links, T)$
        $F1 \leftarrow Evaluate(links, T)$
        $visited \leftarrow visited \cup \{[c, F1]\}$
        **if** $g.c.F_{sim} \neq h.c.F_{sim}$ **and** $F1 \geq g.F1$ **and** $F1 \geq h.F1$ **then**
          $next \leftarrow next \cup \{[c, F1]\}$
    $candidate \leftarrow next$
  $F1 \leftarrow Evaluate(argmax_{v \in visited}(v.F1).c, V)$
  $C_{agg} \leftarrow C_{agg} \cup \{[c, F1]\}$
$[C_{opt}, F1] \leftarrow argmax_{v \in C_{agg}}(v.F1)$
**return** $C_{opt}$

must not be less than that of the components. This heuristic does not guarantee an identical result with exhaustive search. However, it is reasonable as a series of similarity functions that reduces the performance has little possibility of generating a further combination with improvement. Meanwhile, the exhaustive search is extremely expensive.

*cLearn* is implemented as part of ScSLINT framework, and its source code is available at http://ri-www.nii.ac.jp/ScSLINT.

## 3 Evaluation

We use OAEI 2010 dataset to compare ScSLINT+*cLearn* with ObjectCoref [2] and the work in [8], which uses Adaboost to train a classifier. This dataset contains five subsets, whose sizes are from medium to large. We use the same amount of training data given to each other system for *cLearn*, on each re-

**Table 1.** F1 score of the compared systems on OAEI 2010.

| Training size | System | Sider-Drugbank | Sider-Diseasome | Sider-DailyMed | Sider-DBpedia | DailyMed-DBpedia |
|---|---|---|---|---|---|---|
| 5% | ScSLINT+*cLearn* | **0.911** | **0.824** | **0.777** | 0.6414 | **0.424** |
| | Adaboost | 0.903 | 0.794 | 0.733 | 0.641 | 0.375 |
| Varied by subset | ScSLINT+*cLearn* | **0.894** | **0.829** | **0.722** | | |
| | ObjectCoref | 0.464 | 0.743 | 0.708 | | |

spective subset for the comparisons. Adaboost uses 5% candidates for training and ObjectCoref uses 20 actual co-references, which is respectively equivalent to 2.3%, 11.6% and 1.2% candidates for the first three subsets. The results on the last two subsets of ObjectCoref are missing. In order to reduce the random noise, for each subset, we repeat the test 10 times and compute the average results, which are reported in Table 1. According to this table, ScSLINT+*cLearn* consistently outperforms other systems. Compared to ObjectCoref, ScSLINT+*cLearn* remarkably improves the results. Compared to Adaboost, *cLearn* is much better on two subsets related to DailyMed, a repository by itself contains the highest number of co-references.

*cLearn* is efficient as the average numbers of configurations that *cLearn* has to check before stopping is only 246. This number is promising because it is much smaller than that of using genetic algorithm, which is reported in [3] with a recommendation of 500 configurations for each iteration. A qualitative comparison with genetic algorithm is considered as future work.

With the efficiency, effectiveness, and small training data requirement of *cLearn* on a real dataset like OAEI 2010, we believe that *cLearn* has promising application in supervised instance matching, including using active learning strategy to even reduce the annotation effort.

## References

[1] Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: 9th KDD. pp. 39–48 (2003)
[2] Hu, W., Chen, J., Cheng, G., Qu, Y.: Objectcoref & falcon-ao: results for oaei 2010. In: 5th Ontology Matching. pp. 158–165 (2010)
[3] Isele, R., Bizer, C.: Active learning of expressive linkage rules using genetic programming. Web Semantics: Science, Services and Agents on the World Wide Web 23, 2–15 (2013)
[4] Ngomo, A.C.N., Lyko, K.: Unsupervised learning of link specifications: Deterministic vs. non-deterministic. In: 8th Ontology Matching. pp. 25–36 (2013)
[5] Nguyen, K., Ichise, R., Le, B.: Interlinking linked data sources using a domain-independent system. In: 2nd JIST. LNCS, vol. 7774, pp. 113–128. Springer (2013)
[6] Nikolov, A., d'Aquin, M., Motta, E.: Unsupervised learning of link discovery configuration. In: 9th ESWC. LNCS, vol. 7295, pp. 119–133. Springer (2012)
[7] Papadakis, G., Ioannou, E., Niederée, C., Fankhauser, P.: Efficient entity resolution for large heterogeneous information spaces. In: 4th WSDM. pp. 535–544. (2011)
[8] 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: 11th ISWC. LNCS, vol. 7649, pp. 460–475. Springer (2012)