Technical Report on "Description-Similarity Rules: Towards Flexible Feature Engineering for Entity Matching"

Yafeng Tang[1], Zheng Liang[1], Hongzhi Wang[1,✉], Xiaoou Ding[1], Tianyu Mu[1], Huan Hu[2]
[1] School of Computer Science, Harbin Institute of Technology, Harbin, China
[2] Huawei Cloud Computing Technologies Co., Ltd, Hangzhou, China
tangyf@stu.hit.edu.cn,{lz20, wangzh, dingxiaoou_hit,mutianyu}@hit.edu.cn, huhuan18@huawei.com

I. Full Proof of all Proposition

We start by proving the probability of error for the CSAR algorithm. Note that this is a corollary of theorem 1 in [23]. We give this proof in supplementary material to make our paper self-contained.

Corollary 1. *The probability of error of CSAR algorithm satisfies the following.*

$$e_n \leq 2K^2 exp(-\frac{n-K}{2\overline{log}K * H})$$

where $H = max_{i \in \{1,..,K\}} i * (|\mu_i - \theta|)^{-2}$, $\overline{log}K = \frac{1}{2} + \sum_{i=2}^{K} \frac{1}{i}$

Proof. Consider the event ξ defined by

$$\xi = \{j \in \{1,...,K\}, |\frac{1}{n_k}\sum_{s=1}^{n_k} X_s - f_j| \leq \frac{1}{2}\Delta_{K+1-k}\} \tag{1}$$

By Hoeffding's Inequality and an union bound, the probability of the complementary event $\overline{\xi}$ can be bounded as follows

$$\begin{aligned} P(\overline{\xi}) &\leq \sum_{j=1}^{K}\sum_{k=1}^{K-1} P(|\frac{1}{n_k}\sum_{s=1}^{n_k} X_s - f_j| \leq \frac{1}{2}\Delta_{K+1-k}) \\ &\leq \sum_{j=1}^{K}\sum_{k=1}^{K-1} 2exp(2n_k(\Delta_{K+1-k})/2)^2) \\ &\leq 2K^2 exp(-\frac{n-K}{2\overline{log}K * H}) \end{aligned} \tag{2}$$

where the last inequality comes from the fact that

$$\begin{aligned} & n_k(\Delta_{K+1-k})^2 \\ \geq & \frac{n-K}{\overline{log}(K)(K+1-H)(\Delta_{K+1-k})^{-2}} \\ & \geq \frac{n-K}{\overline{log}(K) * H} \end{aligned} \tag{3}$$

Thus, it suffices to show that on the event ξ, the algorithm does not make any error. We prove this by induction on k. Let $k \geq 1$. Assume the algorithm makes no error in all previous $k-1$ stages, that is no bad arm $\mu_i < \theta$ has been accepted and no good arm $\mu_i \geq \theta$ has been rejected. Note that event ξ implies that at the end of stage k, all empirical means are within $\frac{1}{2}(\Delta_{K+1-k})^{-2}$ of the respective true means.

Let $A_k = \{a_1, ..., a_{K+1-k}\}$ be the set of active arms during phase k. We order the a_i's such that $\mu_{a_1} > \mu_{a_2} > ... > \mu_{a_{K+1-k}}$. Denote $m' = m(k)$ for the number of arms that are left to find in phase k. The assumption that no error occurs in the first $k-1$ stages implies that

$$a_1, a_2, ..., a_{m'} \in \{1, ..., m\} \tag{4}$$

and

$$a_{m'+1}, ..., a_{K+1-k} \in \{m+1, ..., K\} \tag{5}$$

If an error is made at stage k, it can be one of the following two types:

1. The algorithm accepts a_j at stage k for some $k \geq m'+1$
2. The algorithm rejects a_j at stage k for some $j \leq m'$

Denote $\sigma = \sigma_k$ for the bijection (from $\{1, ..., K+1-k\}$ to A_k) such that $\overline{\mu}_{\sigma(1),n_k} \geq \overline{\mu}_{\sigma(2),n_k} \geq ... \geq \overline{\mu}_{\sigma(K+1-k),n_k}$. Suppose Type 1 error occurs. Then $a_j = \sigma(1)$ since if algorithm accepts, it must accept the empirical best arm. Furthermore we also have

$$\overline{\mu}_{a_j,n_k} - \theta \geq \theta - \overline{\mu}_{\sigma(K+1-k),n_k} \tag{6}$$

since otherwise the algorithm would rather reject arm $\sigma(K+1-k)$. The condition $a_j = \sigma(1)$ and the event ξ implies that

$$\begin{aligned} \overline{\mu}_{a_j,n_k} &\geq \overline{\mu}_{a_j,n_k}, \\ \mu_{a_j} + \frac{1}{2}(\Delta_{K+1-k}) &\geq \mu_{a_1} - \frac{1}{2}(\Delta_{K+1-k}), \\ (\Delta_{K+1-k}) &\geq \mu_{a_1} - \mu_{a_j} \geq \mu_{a_1} - \theta \end{aligned} \tag{7}$$

We then look at the condition (16). In the event of ξ, for all $i \leq m'$ we have

✉ Hongzhi Wang is the corresponding author.

$$
\begin{aligned}
\overline{\mu}_{a_j,n_k} &\geq \mu_{a_j} - \frac{1}{2}\Delta_{(K+1-k)} \\
&\geq \mu_{a_{m'}} - \frac{1}{2}\Delta_{(K+1-k)} \\
&\geq \theta - \frac{1}{2}\Delta_{(K+1-k)}
\end{aligned} \tag{8}
$$

On the other hand, $\overline{\mu}_{\sigma(K+1-k),n_k} \leq \overline{\mu}_{a_{K+1-k},n_k} \leq \overline{\mu}_{a_{K+1-k},n_k} + \frac{1}{2}\Delta_{(K+1-k)}$. Therefore, using those two observations and (16) we deduce

$$
\begin{aligned}
(\mu_{a_j} + \frac{1}{2}\Delta_{(K+1-k)}) - \theta \geq \theta - (\mu_{a_{K+1-k}} + \frac{1}{2}\Delta_{(K+1-k)}), \\
\Delta_{(K+1-k)} \geq 2\theta - \mu_{a_j} - \mu_{a_{K+1-k}} > \theta - \mu_{a_{K+1-k}}
\end{aligned} \tag{9}
$$

Thus so far we proved that if there is a Type 1 error, then

$$\Delta_{(K+1-k)} > max(\mu_{a_1} - \theta, \theta - \mu_{a_{K+1-k}}) \tag{10}$$

But at stage k, only k-1 arms have been accepted or rejected, thus $\Delta_{(K+1-k)} \leq max(\mu_{a_1} - \theta, \theta - \mu_{a_{K+1-k}})$. By contradiction, we conclude that Type 1 error does not occur.

Suppose Type 2 error occurs. The reasoning is symmetric to Type 1. This completes the induction and consequently the proof of the theorem.

□

Now we give a full version of the proof for proposition 1.

Proposition 1. *The probability of error of P-CSAR satisfies:*

$$e_N \leq 2aK^2 exp(-\frac{n - aK}{2a\overline{log}K * H(a)}) \tag{11}$$

where $H = max_{i\in\{1,..,K\}} i * (|\mu_i - \theta|)^{-2}$, $H(a) = max_{a\in MAB} H(a)$.

Proof. Consider the events ξ_d for each pre-trained MAB defined by

$$
\begin{aligned}
\xi_{d_1} = \{j \in \{1, ..., K\}, \\
|\frac{1}{n_k}\sum_{s=1}^{n_k} X_{s,d_1} - f_{j,d_1}| \leq \frac{1}{2}\Delta_{K+1-k}\} \\
\xi_{d_2} = \{j \in \{1, ..., K\}, \\
|\frac{1}{n_k}\sum_{s=1}^{n_k} X_{s,d_2} - f_{j,d_2}| \leq \frac{1}{2}\Delta_{K+1-k}\} \\
... \\
\xi_{d_{|D|}} = \{j \in \{1, ..., K\}, \\
|\frac{1}{n_k}\sum_{s=1}^{n_k} X_{s,d_{|D|}} - f_{j,d_{|D|}}| \leq \frac{1}{2}\Delta_{K+1-k}\}
\end{aligned} \tag{12}
$$

Also, consider the event ξ defined by

$$\xi = \{j \in \{1, ..., K\}, |\frac{1}{n_k}\sum_{s=1}^{n_k} X_s - f_j| \leq \frac{1}{2}\Delta_{K+1-k}\} \tag{13}$$

where f_j is defined as:

$$
\begin{aligned}
f_j = \frac{\sum_{sim_j \in t, A' \in D} cos < \vec{D}, \vec{D'} > *p(t, A')}{\sum_{sim_j \in t, A' \in D} cos < \vec{D}, \vec{D'} >} \\
= \frac{\sum_d cos < \vec{D}, \vec{d} > *p(t, d)}{\sum_d cos < \vec{D}, \vec{d} >} \\
= \frac{\sum_d cos < \vec{D}, \vec{d} > *f_{j,d}}{\sum_d cos < \vec{D}, \vec{d} >}
\end{aligned} \tag{14}
$$

Setting $w_d = cos < \vec{D}, \vec{d} >$, we can rewrite (13) as:

$$
\begin{aligned}
\xi = \{1 \leq j \leq K, \\
|\frac{1}{n_k}\sum_{s=1}^{n_k} \frac{\sum_d w_d * X_{s,d}}{\sum_{d\in D} w_d} - \frac{\sum_d w_d * f_{j,d}}{\sum_{d\in D} w_d}| \leq \frac{1}{2}\Delta_{K+1-k}\} \\
= \{1 \leq j \leq K, \\
|\frac{1}{n_k}\sum_{s=1}^{n_k}\sum_d w_d * X_{s,d} - \sum_d w_d * f_{j,d}| \leq \frac{1}{2}\sum_{d\in D} w_d\Delta_{K+1-k}\}
\end{aligned} \tag{15}
$$

Suppose (12) is true. Using the absolute value inequality, for any $1 \leq j \leq K$, we have:

$$
\begin{aligned}
|\frac{1}{n_k}\sum_{s=1}^{n_k}\sum_d w_d * X_{s,d} - \sum_d w_d * f_{j,d}| \\
= |\sum_d (\frac{1}{n_k}\sum_{s=1}^{n_k} w_d * X_{s,d} - w_d * f_{j,d})| \\
\leq \sum_d w_d * |\frac{1}{n_k}\sum_{s=1}^{n_k} X_{s,d} - f_{j,d}| \\
\leq \frac{1}{2}\sum_{d\in D} w_d\Delta_{K+1-k}
\end{aligned} \tag{16}
$$

This means when $\xi_{d_1} \cap ... \cap \xi_{d_{|D|}}$ is true, ξ has to hold true regardless of w_d. Its contrapositive implies that when $\overline{\xi}$ is true, $\overline{\xi_{d_1}} \cup ... \cup \overline{\xi_{d_{|D|}}}$ has to hold true regardless of w_d. By full probability law, we have

$$
\begin{aligned}
P(\overline{\xi_{d_1}} \cup ... \cup \overline{\xi_{d_{|D|}}}) = P(\overline{\xi_{d_1}} \cup ... \cup \overline{\xi_{d_{|D|}}}|\xi) * P(\xi) + \\
P(\overline{\xi_{d_1}} \cup ... \cup \overline{\xi_{d_{|D|}}}|\overline{\xi}) * P(\overline{\xi}) \\
\geq P(\overline{\xi_{d_1}} \cup ... \cup \overline{\xi_{d_{|D|}}}|\overline{\xi}) * P(\overline{\xi}) = P(\overline{\xi})
\end{aligned} \tag{17}
$$

In algorithm 2, for each $MAB_i \in \{MAB_1, ..., MAB_n\}$ we have

$$P(\overline{\xi_d}) \leq 2K^2 exp(-\frac{\frac{n}{a} - K}{2\overline{log}K * H(i)}) \tag{18}$$

By union bound, we come to the conclusion that

$$P(\bar{\xi}) \leq P(\overline{\xi_{d_1}} \cup ... \cup \overline{\xi_{d_{|D|}}}) \leq \sum_{i=1}^{a} 2K^2 exp(-\frac{\frac{n}{a}-K}{2\overline{log}K * H(i)})$$
$$\leq 2aK^2 exp(-\frac{\frac{n}{a}-K}{2\overline{log}K * H(a)}) \quad (19)$$

Thus, it suffices to show that, by probability of $2aK^2 exp(-\frac{\frac{n}{a}-K}{2\overline{log}K * H(a)})$, the algorithm does not make any error.

□

Proposition 2. *If DSR $d_1 > \delta_1, d_2 > \delta_2, ..., d_k > \delta_k \to sim$ holds, for $\delta_1 < \delta'_1, \delta_2 < \delta'_2, ..., \delta_k < \delta'_k$, DSR $d_1 > \delta'_1, d_2 > \delta'_2, ..., d_k > \delta'_k \to sim$ must hold.*

Proof. We prove this by demonstrating the equivalence of the two below DSR sets.

$DSR_1 = \{d_1 > \delta_1, d_2 > \delta_2, ..., d_k > \delta_k \to sim\}$

$DSR_2 = \{d_1 > \delta_1, d_2 > \delta_2, ..., d_k > \delta_k \to sim, d_1 > \delta'_1, d_2 > \delta'_2, ..., d_k > \delta'_k \to sim\}$

Given an attribute pair a_1, a_2 of an EM task(e.g. name for dblp-acm), its description is a k-dimensional point $(d_1, d_2, ..., d_k)$ which falls in one of the following categories:

1. $(d_1, d_2, ..., d_k) \in DSR_2$
2. $(d_1, d_2, ..., d_k) \in DSR_1 \setminus DSR_2$
3. $(d_1, d_2, ..., d_k) \notin DSR_1$

For category 1 and 2, both DSR_1 and DSR_2 will recommend sim.

For category 3, both DSR_1 and DSR_2 will not recommend sim.

To sum up, DSR_1 and DSR_2 are equivalent when used for an EM task. Therefore, if DSR $d_1 > \delta_1, d_2 > \delta_2, ..., d_k > \delta_k \to sim$ holds, for all $\delta_1 < \delta'_1, \delta_2 < \delta'_2, ..., \delta_k < \delta'_k$, whether DSR $d_1 > \delta'_1, d_2 > \delta'_2, ..., d_k > \delta'_k \to sim$ holds makes no difference in practice. So in our rule mining algorithm, we suppose they all hold and prune them due the equivalence proved above.

□

Proposition 3. *If the support of the r-th largest number p in C is $S(p) = \frac{evi(p)}{obs(p)}$, the following two propositions hold:*

(1) If $S(p) < sup$, $rank(\delta^*) \notin R(p) = [r - l(-), r + r(-)]$

(2) If $S(p) \geq sup$, $rank(\delta^*) \notin R(p) = [r - l(+), \infty)$

where we denote

$$l(-) = \lfloor \frac{sup * obs(p) - evi(p)}{1 - sup} \rfloor$$
$$r(-) = \lfloor obs(p) - \frac{evi(p)}{sup} \rfloor$$
$$l(+) = \lfloor \frac{evi(p)}{sup} - obs(p) \rfloor$$

to slightly shorten the notation.

Proof. The proposition is the result of solving the below inequalities.

$$S(\delta^*) = \frac{evi(p)}{obs(p) - l(-)} \geq sup$$
$$S(\delta^*) = \frac{evi(p) + r(-)}{obs(p) + r(-)} \geq sup \quad (20)$$
$$S(\delta^*) = \frac{evi(p) + l(+)}{obs(p) + l(+)} \leq sup$$

□

Proposition 4. *If the size of the ADS collection is N, the size of possible candidate set is $O(N^k)$.*

Proof. We prove this by the below observation.

Given a ADS collection of size $\{ads_i = (d_1^i, ..., d_k^i), 1 \leq i \leq N\}$, a random projection $\pi(i) = j, 1 \leq i, j \leq N$, there always exist a 0-1 assignment to make $(d_1^{\pi}(i), ..., d_k^{\pi}(i))$ a satisfactory point for DSR mining.

The observation is easy to satisfy: just assign all the points within the DSR range of point $(d_1^{\pi}(i), ..., d_k^{\pi}(i))$ to 1. And when this assignment is impossible for points close to O', it is obvious only linear candidates can be pruned, so the possible candidate set is $O(N^k)$ □

Proposition 5. *If DSR $d_1 > \delta_1, d_2 > \delta_2, ..., d_{k_1} > \delta_{k_1} \to sim$ holds, the DSR $d_1 > \delta_1, d_2 > \delta_2, ..., d_{k_1} > \delta_{k_1}, ..., d_{k_2} > \delta_{k_2} \to sim$ must hold.*

Proof. Like proposition 2, we prove this by demonstrating the equivalence of the two below DSR sets.

$DSR_1 = \{d_1 > \delta_1, d_2 > \delta_2, ..., d_{k_1} > \delta_{k_1} \to sim\}$

$DSR_2 = \{d_1 > \delta_1, d_2 > \delta_2, ..., d_{k_1} > \delta_{k_1} \to sim, d_1 > \delta_1, d_2 > \delta_2, ..., d_{k_1} > \delta_{k_1}, ..., d_{k_2} > \delta_{k_2} \to sim\}$

Given an attribute pair a_1, a_2 of an EM task(e.g. name for dblp-acm), its description is a k-dimensional point $(d_1, d_2, ..., d_{k_2})$ which falls in one of the following categories:

1. $(d_1, d_2, ..., d_{k_2}) \in DSR_2$
2. $(d_1, d_2, ..., d_{k_2}) \in DSR_1 \setminus DSR_2$
3. $(d_1, d_2, ..., d_{k_2}) \notin DSR_1$

For category 1 and 2, both DSR_1 and DSR_2 will recommend sim.

For category 3, both DSR_1 and DSR_2 will not recommend sim.

To sum up, DSR_1 and DSR_2 are equivalent when used for an EM task. That is to say, if DSR $d_1 > \delta_1, d_2 > \delta_2, ..., d_{k_1} > \delta_{k_1} \to sim$ holds, for all $\delta_{k_1+1}, ..., \delta_{k_2}$, whether DSR $d_1 > \delta_1, d_2 > \delta_2, ..., d_{k_1} > \delta_{k_1}, ..., d_{k_2} > \delta_{k_2} \to sim$ holds makes no difference in practice. So in our rule mining algorithm, we suppose they all holds and are pruned due the equivalence proved above. □

Proposition 6. *The time complexity of Algorithm 4 is $O(N^k)$.*

Proof. $k = 2$ is a simple case which we conceptually elaborated above that its complexity is $O(N^2)$. So we focus on the case with $k > 2$. Stratified-Sort() takes $O(NlogN)$ time for each dimension, therefore $O(kNlogN)$ in total.

Dimensional-Pruning() takes $O(1)$ time for each existing rule in $k-1$ dimensions. Suppose that Pruned-IES is performed for $k-1$ dimensions. Traversing all minimal hyper-cubes(with no data point inside) in the inner- and inter- strata order, we can guarantee that all the $F(.)$ in the Equation except for $\sum_{i_1=1}^{k} \cdots \sum_{i_k=k}^{k} F(\vec{a_{i_1}} + \vec{a_{i_2}} + ... + \vec{a_{i_k}})$ is already known, and since the Equation has $O(2^k)$ terms, line 10 takes $O(2^k * (N-1)^k) = O(N^k)$ time. Using the recurrence relation, we can come to the conclusion that Pruned-IES takes $O(N^k)$ time. □

Finally, we discuss the inherent complexity of Problem 3 in the worst case. Note that this complexity is a loose bound, since it is only the complexity of printing the solution.

Proposition 7. *The inherent complexity of Problem 3 is $O(N^{k-1})$ in the worst case.*

Proof. We prove this by giving an instance. Suppose that there are $N-1$ zeros and only a single one pareto dominated by all zeros, and sup is the smallest positive. There should be $C_{N-1}^{k} = O(N^k)$ existing satisfactory solution and therefore $O(N^k - 1)$ Pareto solution. Figure 5(d) in our original paper is an example of $N = 5, k = 2$. The inherent time complexity is the cost of printing them, which is $O(N^{k-1})$. □

II. Design Choice of Descriptions

In this section, we conducted an analysis to examine the relationship between attribute description and similarities. We present four statistical measures for attribute descriptions, including: (1)average length d_1: `length` (2)average word num d_2: `AWN`, (3)entropy d_3 :`entropy` , and (4) Index of Coincidence d_4: `IoC` [24]. Among these descriptions, d_1 and d_2 describes the feature of the entire string, while d_3 and d_4 describes the frequency of characters. These four descriptions have covered mainstream similarity functions [1].

Based on similarity function libraries established on [7] and mentioned in AutoML-EM [2], we sample similarities from each category and show them in Figure 1. We classify current attribute into four data types, and recommend four kinds of mainstream similarity functions separately. We shows the relationship between our *descriptions* and similarities, and give an example based on the name of basketball players as the attribute. The input attribute pairs are presented as s_1 and s_2.

Levenshtein Distance. Levenshtein Distance [8] tends to increase with the `length` of the strings, since the longer strings offer more possibilities for editing operations. Levenshtein distance is designed based on dynamic programming and its dynamic transition equation is as follows:

$$D[i,j] = \min \begin{cases} D[i-1,j]+1 \\ D[i,j-1]+1 \\ D[i-1,j-1]+\text{cost}(s_1[i],s_2[j]) \end{cases}$$

where the function $cost(s_1[i], s_2[j])$ represents the editing cost to transform character s_1 into character s_2, which relies on `length` to traverse the entire string.

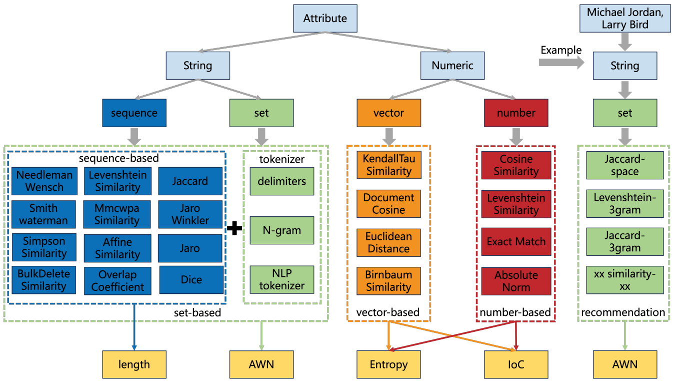

Fig. 1: A tree diagram showing how *descriptions* are correlated to the similarity functions for different data types. We also give an attribute example on the right.

Mmcwpa Similarity. Modified minCost with prefix and approximate(mmcwpa) similarity [25] is the regularized version of Levenshtein similarity. It is also associate with `length` when calculating its dynamic transition equation.

Affine Similarity. Affine Similarity [10] assigns scores to the following four operations between characters in Levenshtein [8]: the match/mismatch, insert, and delete. By comparing these scores, affine selects the path that maximizes the score as the optimization target to calculate the final similarity. The affine similarity is shown in Equation 21.

$$affine(s_1, s_2) = \frac{Levenshtein(match, mismatch, insert, delete)}{max(d_1(s1), d_1(s2))} \tag{21}$$

Clearly, the denominator in Equation 21 is our description `length` of attribute pairs.

Needleman-Wensch and Smith-Waterman As sequence-based similarities, both of them has the same definition on the operations scoring as affine similarity. The goal of Needleman-Wensch [11] is to maximize characters in the same position. Needleman-Wensch requires traversing the entire string in order to make the two strings equal in `length` after several operations. However, Smith-Waterman algorithm [12] does not require reaching the end of either sequence and allows for faster backtracking. Therefore, similarly to Levenshtein and affine, both of them are sensitive to `length`.

BulkDelete Similarity. BulkDelete similarity [14] is based on LCS distance [3] and the `length` of attribute pairs, which is defined as follows.

$$\text{BulkDelete}(s_1, s_2) = \frac{LCS(s_1, s_2)}{min(d_1(s_1), d_1(s_2))}$$

BulkDelete similarity is widely used especially when two strings with many similar substrings but are not entirely identical.

Jaccard. Jaccard similarity [15] is an individual-based similarity, which is defined as the size of the intersection of two sets divided by the size of their union. Given attribute pairs (s_1, s_2), their Jaccard similarity can be calculated as follows.

$$\text{Jaccard}(s_1, s_2) = \frac{|s_1 \cap s_2|}{|s_1 \cup s_2|}$$

Dice. Dice is a sequence-based similarity function. Dice [16] performs better than jaccard when dealing with binary data or focusing on the presence or absence of elements in sets. Given attribute pairs (s_1, s_2), dice is calculated as follows:

$$\text{Dice}(s_1, s_2) = \frac{2 \times |s_1 \cap s_2|}{d_1(s_1) + d_1(s_2)}$$

As we can see, the denominator in Dice calculates the size of the set. Similar to the conclusion about Levenshtein, there is a relationship between dice and either `length` or `AWN`, depending on whether the tokenizer is used.

Simpson. As an extension of jaccard, Simpson [17] is calculated as the ratio of the intersection size to the size of the smaller set. Simpson similarity can be defined as follows.

$$\text{Simpson Similarity}(s_1, s_2) = \frac{|s_1 \cap s_2|}{min(d_1(s_1), d_1(s_2))}$$

As we can see, the denominator of simpson calculates the minimum value of (s_1, s_2) length, hence it is related to the description `length`.

Overlap-Coefficient. By applying tokenizer N on Simpson similarity, we have Overlap-coefficient [4],, a set-based similarity, where `AWN` is used to calculate the minimal set size as follows.

$$overlap - coefficient = \frac{N(s_1) \cap N(s_2)}{\min(d_2(N(s_1)), d_2(N(s_2)))}$$

Jaro As an sequence-based similarity, Jaro [5] defines a tolerance range R as the half of $\max(d_1(s1), d_2(s2))$. Then, the Jaro similarity can be defined as follows.

$$Jaro(s_1, s_2) = \frac{1}{3}\left(\frac{m}{d_1(s_1)} + \frac{m}{d_1(s_2)} + \frac{m-n}{m}\right)$$

where m denotes the number of matched characters inside R, and n denotes that of identical characters at the different positions. Obviously, `length` has an impact on the Jaro similarity.

Jaro Winkler. As an extension of the Jaro, Jaro-Winkler [18] considers `length` and the position of matching characters, making it suitable for comparing strings of different lengths and identifying similarities even in the presence of minor discrepancies or variations. The Jaro-Winkler similarity is calculated as follows:

$$JaroWinkler(s_1, s_2) = J(s_1, s_2) + \frac{lp(1 - J(s_1, s_2))}{3}$$

where $J(s_1, s_2)$ represents the Jaro similarity between the strings (s_1, s_2). lp is a parameter that can increase the similarity for strings with a common prefix. Typically, lp is set to 0.1.

Please note that the above similarity functions can be adapted to data type *sets* by adding delimiters, for example, *Levenshtein-3gram* similarity. In this case, `AWN` becomes more relevant than `length`. In our experiment, we provide five different tokenizer including:space, 3-gram and index-split, nlp-split, base-split, which are recommended in HANLP [19].

Document-cosine. Document-cosine [6] is related to vector-based similarities for calculating numerical attributes [9]. In practical terms, we can invert attribute pairs (s_1, s_2) into `entropy` vectors($d_3(s_1), d_3(s_2)$), and calculate Document-cosine similarity as follows.

$$Doc - Cosine = \frac{d_3(str_1) \cdot d_3(str_2)}{|d_3(str_1)||d_3(str_2)|}$$

where d_3 is the `entropy` vector for the given attribute. For more details, please refer our Equation (3) in the origin paper.

Birnbaum Similarity. Birnbaum similarity [13] is based on the co-occurrence matrix. Given a co-occurrence matrix M, where M_{ij} represents the frequency of character j co-occurring with attribute value i. R is the global schema. Birnbaum similarity can be defined as follows.

$$\text{Birnbaum}(s_1, s_2) = \frac{\sum_{i \in R, j \in s_1 \cup s_2} \min(M_{ij}, M'_{ij})}{\sum_{i \in R, j \in s_1 \cup s_2} \max(M_{ij}, M'_{ij})}$$

where M_i and M'_i present the co-occurrence matrix of s_1 and s_2. It is worth noting that `IoC` can be derived from matrix A. For example, $IoC(s_1) = 1 - \sum_{i \in R, j \in s_1} M_{ij}^2$

Euclidean Distance. Euclidean Distance is a vector-based similarity. With the help of `entropy`, we can describe strings in vector form. Given the attribute A_1, A_2 and its corresponding vocabulary, euclidean distance is defined as follows.

$$\text{Euclidean distance} = \sqrt{\sum_{i=1}^{n}(d_3(s_{2,i}) - d_3(s_{1,i}))^2}$$

Exact Match. Exact Match is used to determine whether two strings are completely identical, returning a boolean variable. It first checks if the length of the input strings are equal, which is related to `length`.

KendallTau. KendallTau [20] is a sequence-based and vector-based similarity, which maintains a vocabulary V for all characters in s1 and s2. KendallTau assigns $d_3(s)$ to each input string s. Each dimension of $d_3(s)$ corresponds to the frequency of a character in V. KendallTau counts the occurrences of each character in the two input strings separately to get Kendall vectors v_1 and v_2. KendallTau can be defined as follows.

$$\text{KendallTau}(s_1, s_2) = \frac{2}{n(n-1)} \sum_{n}^{i<j} sgn(v_{1i} - v_{1j}) sgn(v_{2i} - v_{2j})$$

where sgn is defined as

$$sgn(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0 \end{cases}$$

KendallTau generates vectors is based on our description `entropy`, which can be used to measure the similarity of permutation order between time series.

It should be noted that both `IoC` and `entropy` are sensitive in capturing the impact of character frequency changes on similarity functions. Therefore, most similarity functions related to entropy are also, to some extent, related to IoC.

References

[1] Santiago Ontañón: An Overview of Distance and Similarity Functions for Struc- tured Data. CoRR abs/2002.07420 (2020)

[2] Pei Wang, Weiling Zheng, Jiannan Wang, Jian Pei: Automating Entity Matching Model Development. ICDE 2021: 1296-1307

[3] V. Chvátal and D. Sankoff, "Longest common subsequences of two random sequences," Journal of Applied Probability, vol. 12, no. 2, pp. 306–315, 1975. doi:10.2307/3212444

[4] Vijaymeena M K , Kavitha K , .A Survey on Similarity Measures in Text Min- ing[J].Machine Learning & Applications An International Journal, 2016, 3(1):19- 28.DOI:10.5121/mlaij.2016.3103.

[5] Matthew A. Jaro (1989) Advances in Record-Linkage Methodology as Applied to Matching the 1985 Census of Tampa, Florida, Journal of the American Statistical Association, 84:406, 414-420, DOI: 10.1080/01621459.1989.10478785

[6] Dan Tian, Mingchao Li, Yang Shen, Shuai Han: Intelligent mining of safety hazard information from construction documents using semantic similarity and information entropy. Eng. Appl. Artif. Intell. 119: 105742 (2023)

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

[8] Schulz, Klaus U. and Stoyan Mihov. "Fast string correction with Levenshtein automata." International Journal on Document Analysis and Recognition 5 (2002): 67-85.

[9] Topsoe, F.: Some inequalities for information divergence and related measures of discrimination. IEEE Trans. Inf. Theor. 46(4), 1602–1609 (2000)

[10] Snapper, Ernst . "Metric affine geometry. " Metric Affine Geometry 430.11(1971):1-111.

[11] Saul B. Needleman, Christian D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, Journal of Molecular Biology,Volume 48, Issue 3,1970,Pages 443-453

[12] Shiwei Wei, Yuping Wang, Yiu-ming Cheung, A Branch Elimination-based Efficient Algorithm for Large-scale Multiple Longest Common Subsequence Problem, IEEE Transactions on Knowledge and Data Engineering, (1-1), (2021).

[13] Amores A, Force A, Yan YL, Joly L, Amemiya C, Fritz A, Ho R, Langeland J, Prince V, Wang YL, Westerfield M, Ekker M, Postlethwait JM. 1998. Zebrafish hox clusters and vertebrate genome evolution. Science 282: 1711–1714.

[14] Ballantine, J. P., & Jerbert, A. R. (1952). Distance from a Line, or Plane, to a Poin. The American Mathematical Monthly, 59(4), 242–243. https://doi.org/10.2307/2306514

[15] LEVANDOWSKY, M., WINTER, D. Distance between Sets. Nature 234, 34–35 (1971). https://doi.org/10.1038/234034a0

[16] Reuben R Shamir, Yuval Duchin, Jinyoung Kim, Guillermo Sapiro, Noam Harel bioRxiv 306977; doi: https://doi.org/10.1101/306977

[17] Vijaymeena, M. K.; Kavitha, K. (March 2016). "A Survey on Similarity Measures in Text Mining" (PDF). Machine Learning and Applications. 3 (1): 19–28. doi:10.5121/mlaij.2016.3103.

[18] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. 2003. A comparison of string distance metrics for name-matching tasks. In Proceedings of the 2003 International Conference on Information Integration on the Web (IIWEB'03). AAAI Press, 73–78.

[19] https://www.hanlp.com/semantics/dashboard/index

[20] Fagin, R.; Kumar, R.; Sivakumar, D. (2003). Comparing top k lists. SIAM Journal on Discrete Mathematics. 17 (1): 134–160

[21] R. W. Hamming, "Error detecting and error correcting codes," in The Bell System Technical Journal, vol. 29, no. 2, pp. 147-160, April 1950, doi: 10.1002/j.1538-7305.1950.tb00463.x.

[22] Moses S. Charikar. 2002. Similarity estimation techniques from rounding algorithms. In Proceedings of the thiry-fourth annual ACM symposium on Theory of computing (STOC '02). Association for Computing Machinery, New York, NY, USA, 380–388. https://doi.org/10.1145/509907.509965

[23] Multiple Identifications in Multi-Armed Bandits Séebastian Bubeck, Tengyao Wang, Nitin Viswanathan Proceedings of the 30th International Conference on Machine Learning, PMLR 28(1):258-265, 2013.

[24] Radu Cornel Guiasu, Silviu Guiasu: Conditional and Weighted Measures of Ecological Diversity. Int. J. Uncertain. Fuzziness Knowl. Based Syst. 11(3): 283-300 (2003)

[25] Tresoldi, Tiago. (2016). Newer method of string comparison: the Modified Moving Contracting Window Pattern Algorithm.