Available online at www.ijpe-online.com

International Journal of Performability Engineering

Vol. 13, No. 4, July 2017, pp. 427-436
DOI: 10.23940/ijpe.17.04.p10.427436

Similarity Entropy-Based Self-Adaptive String Outlier Detection Method

Ou Ye*, Zhanli Li

School of Computer Science and Technology, Xi'an University of Science and Technology
Xi'an, 710054, Shannxi, China

Abstract

Although a large variety of outlier detection techniques have been developed, the algorithms pay less attention to the impact of structure factor on semantics for string data, and the threshold is difficult to be given automatically with unknown distribution law of string data, so the accuracy of string outlier detection is difficult to be ensured. This paper presents a similarity entropy-based self-adaptive string outlier detection method to address this issue. Firstly, semantic similarity is calculated by matrix computation based on word matching, and structure similarity is calculated by considering the structure factors. On this basis, string data is mapped into similarity cells, and they are detected to identify outlier data by using similarity distance. In order to reduce the sensitivity problem of threshold, the similarity entropy histogram is constructed to determine the dynamic threshold. The simulation experiments are conducted to prove the feasibility and rationality of this method, and the results show that this method can reduce sensitivity problem of threshold and ensure accuracy.

Keywords: outlier detection; semantic similarity; structure similarity; similarity entropy; similarity distance

(Submitted on February 27, 2017; Revised on April 2, 2017; Accepted on May 17, 2017)

1. Introduction

At present, client information is very important for any enterprise. It can be used to contact clients and extend the enterprise's business. In client information, there are some string data, such as client name, address, and enterprise information. However, when the amount of client information increases, string outlier data also increases. Because they have good hiding features in information, it is difficult to be found directly. Outlier detection technologies can be used to address this issue.

Currently, outlier detection technologies have been used in finance, meteorological forecast, data cleansing and other fields [1]. The research achievements mainly focus on two respects: optimization algorithms [2] and applications [3]. In general, there are four kinds of key technologies for outlier detection, which are statistical-based methods [4,5], density-based methods [6], distance-based methods [7] and cluster-based methods [8]. Because string data in client information have fewer attributes and do not abide by specific probability distribution, the distance-based methods can be used to detect the string outliers.

Despite many previous works on outlier detection based on distance, few of them can work well in string outlier detection, due to the following challenges:

* Corresponding author. Tel.: +86-13572437594.
E-mail address: 785669070@qq.com.

- Because the existing distance-based outlier detection methods do not consider the impact of structure on semantics for string data as much, some data that has different structures but similar semantics are likely to be considered as the outliers.
- Single distance measurements can't ensure the accuracy of string outlier detection.
- The given threshold has the obvious influence on outlier detection results with unknown distribution law of string data.

Our motivation is to propose a similarity entropy-based self-adaptive string outlier detection method. By calculating the semantic and structure similarities to reduce the impact of structure factor, and using histogram statistics of similarity entropy to determine the threshold, we address the sensitivity problem of threshold. In summary, our contributions mainly lie in three aspects:

- Considering the impact of structure on semantic, we propose the new semantic and structure similarities measurement mechanisms for string data. In the similarity calculation, matrix computation based on word matching is used to calculate the semantic similarity and reduce redundant matching. It is word-level matching rather than character-level matching.
- We detect the string outlier data in similarity cell by using the semantic and structure similarities to ensure the accuracy of outlier detection.
- On the basis of similarity entropy, by using the similarity entropy histogram to determine the dynamic threshold, we reduce the sensitivity problem of threshold.

The rest of this paper is organized as follows: in section 2, the related works are described briefly; in section 3, the similarity entropy-based self-adaptive string outlier detection method is proposed; in section 4, the simulation results are discussed; finally, we discuss conclusions and future works in section 5.

2. Related Work

Nested-Loop algorithm (NL) and cell-based algorithm (CB) are common distance-based algorithm. Among them, the distance of each pair of objects between two data blocks is calculated to detect the outliers in Nested-Loop algorithm, but the time complexity is high. In order to reduce the time complexity of Nested-Loop algorithm, the outliers are detected by calculating the distances between data and counting the number in cell-based algorithm. But only when parameter $k \leqslant 4$, the execution time of algorithm is less than NL algorithm and it is very insensitive for parameter. Partition-based algorithm detects the outliers by calculating DK(p) that is the distance between data p and k^{th} near neighbors. It can address the problem of cell-based algorithm, but single distance measurement can't ensure the accuracy of outlier detection, as shown in [9]. Zhen presents outlier detection algorithm based on double distance (DTKA) to detect outliers by using the double distance, but the semantics of string data is not considered in this algorithm [10]. Fan presents likelihood of semantic outlier algorithm (LSO) to construct the semantic relationship between data first, and then the median of semantic distribution is treated as threshold to detect the outliers [11]. However, because the structure factor of data can affect the semantics, some data are likely to be considered as the outliers because of grammar feature and structure, such as abbreviation or missing words in string data. Additionally, the given threshold in above algorithms has the obvious influence on result.

3. The similarity entropy-based self-adaptive string outlier detection method

3.1. Problem Definition

By considering different distance-based methods for outlier detection, we define the problem of string outlier detection as follows. Suppose we are provided with a training set or database $\boldsymbol{D}=\{X_1, X_2, \ldots, X_N\}$, where is the i^{th} string data, and N is the number of training samples. Moreover, suppose we also have a test set $\boldsymbol{D}^*=\{y_0, y_1, \ldots, y_M\}$, where y_i is the i^{th} string test sample, M is the number of test samples. Our task is to design a function to determine whether y_i in $\boldsymbol{D}^*$ is normal data or string outlier. That is:

$$f : y_i \mapsto \{normal, outlier\} \tag{1}$$

The distance is the critical parameter to judge whether the test sample is normal or outlier in distance-based methods for outlier detection. Therefore, distance-based methods can be denoted in (2), which compare the current testing sample with all the training data. In (2), where Dist($\cdot$) is a pairwise distance and τ_i is a threshold about X_i.

$$f(Dist)=\begin{cases} normal & \forall x_i, Dist(y_i, x_i) \geq \tau_i \\ outlier & otherwise \end{cases} \tag{2}$$

For string data, the pairwise distance can reflect the similarity. Therefore, we can modify (2) to (3),

$$f=\begin{cases} normal & \forall x_i, SimDist(y_i, x_i) \leq \tau_i \\ outlier & otherwise \end{cases} \tag{3}$$

where $SimDist(y_i, x_i)$ is the pairwise similarity distance, it can be quantized using (4):

$$\begin{aligned} SimDist(y_i, x_i) &= Dist(Sim(y_i, x_i)) \\ &= Dist(SimS(y_i, x_i), SimC(y_i, x_i)) \end{aligned} \tag{4}$$

where $SimS(\,)$ denotes the semantic similarity, and $SimC(\,)$ denotes the structure similarity. By using (1)-(4), the string outlier data can be detected. Here, the impact of structure factor on semantics is considered.

3.2. Similarity Measurement

In this section, considering the semantic, structure and weight factors of string data, we propose the new semantic and structure similarities measurement mechanisms. In order to measure the semantics and reduce the redundant matching of pairwise data, we adopt matrix computation based on word matching to calculate the semantic similarity. It is the word-level matching. The process of matrix computation based on word matching is descripted by two steps. Firstly, the i^{th} training sample X_i ($X_i \in \boldsymbol{D}$) and any test sample y_i ($y_i \in \boldsymbol{D^*}$) need to be mapped to two vectors: $V:(w_1, w_2, \cdots, w_n)$ and $V^*:(w'_1, w'_2, \cdots, w'_m)$, where w_n and w'_m are n^{th} and m^{th} word in X_i and y_i. In order to speed up to search X_i in training set $\boldsymbol{D}$, we depend on the keyword of y_i and edit distance algorithm to indetify X_i. That is shown in equation (5), where $Edist(\,)$ is the edit distance; kwd is the most important keyword in sample y_i; $St(\,)$ is the candidate set, which is obtained by using kwd to retrieval $\boldsymbol{D}$. In the candidate set, all training data contain the keyword of y_i. We can calculate the pairwise distance between every data in candidate set and y_i by using edit distance algorithm, the training data with minimal pairwise distance is X_i. If candidate set is null, $SimS(y_i, X_i)=0$, and $SimC(y_i, X_i)=0$.

$$X_i = \arg\min(Edist(St(y_i.kwd))) \tag{5}$$

In order to overcome the shortcoming of edit distance algorithm, we adopt matrix computation based on word matching to calculate the semantic similarity in equation (6).

$$\begin{aligned} SimS(y_i, x_i) &= \left(C \bullet \begin{pmatrix} 1 \\ 1 \\ \cdots \\ 1 \end{pmatrix}_m \right)^T \bullet \chi^T \\ &= \left(V^T \times V^* \bullet I \right)^T \bullet \chi^T \\ &= \left((w_1, w_2, \cdots, w_n)^T \times (w_1', w_2', \cdots, w_m') \bullet I \right)^T \bullet \chi^T \end{aligned} \tag{6}$$

where $\boldsymbol{C}=\{ c_{11}, \cdots, c_{ij,} \cdots, c_{nm,}\}$, and $c_{ij} \in \{0,1\}$. $\boldsymbol{C}$ can be obtained by cross multiplication between two vectors, it is shown in (7) and (8). And $\boldsymbol{\chi}=(\chi_1, \chi_2, \cdots, \chi_n)$ is the weight vector, which denotes the semantics weight of i^{th} word in data, it is shown in (9).

$$V^T \times V^* = C \tag{7}$$

$$w_i \times w_j = \min\left(\frac{SC(w_i \mid y_{i,} w_j \mid x_i)}{\sum_{i \in N, k \in [1,n]}^{n} |w_i|}, \frac{SC(w_i \mid y_{i,} w_j \mid x_i)}{\sum_{j \in M, l \in [1,M]}^{m} |w_j|} \right) \tag{8}$$

$$\sum_{i=0}^{n} \chi_i = 1 \tag{9}$$

Because the structure of string data can affect the semantic meaning, we also need to measure the structure similarity of pairwise data, as in (10):

$$SimC(y_i, x_i) = \delta \times \eta \times (1-\rho) + \rho \times \max\left(\frac{SC(y_i, x_i)}{|y_i|}, \frac{SC(y_i, x_i)}{|x_i|} \right) \tag{10}$$

where η is the switching position factor, it can reflect the impact of position change on semantics, e.g. two strings "you like Jim" and "Jim like you", because of the position change between words, their semantic meanings are different. The parameter δ is the position factor. It can reflect the impact of characters on semantics, e.g. two strings "converting point" and "converging point", because of the different individual characters, their semantic meanings are different, too. The parameter ρ is the abbreviation factor, which can reflect the change of string abbreviation on semantic meaning, it is shown in (13). The parameter η, δ and ρ can be quantized by using (11)-(13):

In (11), $MinChagN(y_i, X_i)$ is the minimal operation number of words that y_i converts to X_i. And in (12), where $|y_i|$ is the length of test sample y_i, and $|X_i|$ is the length of training sample X_i.

$$\eta = MinChagN(y_i, X_i) / |y_i| \tag{11}$$

$$\rho = \begin{cases} 1 & , \ y_i \ \ is\ abbreviation \\ 0 & , \ otherwise \end{cases} \tag{12}$$

$$\delta = \begin{cases} \min(|y_i|/|x_i|, |x_i|/|y_i|) , & |y_i| \neq |x_i| \\ 1 & , \quad otherwise \end{cases} \tag{13}$$

Algorithm 1 Similarity Calculation of String Data

Input: Training Data Samples $\boldsymbol{D}=\{X_1, X_2, ..., X_N\}$, A Test Sample $y_i \in \boldsymbol{D}^*$
Output: SimS(y_i, X_i), SimC(y_i, X_i)

1. The test sample y_i is mapped to the vector $\boldsymbol{V}$.
2. The scale parameter n =Size($\boldsymbol{V}$).
3. Construct the word weight vector $\boldsymbol{\chi}$.
4. Extract the keyword *kwd* from y_i that has the maximum weight in $\boldsymbol{\chi}$.
5. **for** each i=1...N, **do**
   Construct candidate set St(·) by using *kwd* to retrieval $\boldsymbol{D}$, and calculate the edit distance.
   end for
6. $X_i = \arg\min(Edist(St(y_i.kwd)))$
7. The training sample X_i is mapped to the vector $\boldsymbol{V}^*$.
8. Calculate SimS(y_i, X_i)=MatrixCalculation($\boldsymbol{V}$, $\boldsymbol{V}^*$, $\boldsymbol{\chi}$).
9. Calculate SimC(y_i, X_i) by (10)-(13).

Function SimS(y_i, X_i)=MatrixCalculation($\boldsymbol{V}$, $\boldsymbol{V}^*$, $\boldsymbol{\chi}$)

1. **for** every element $w_i \leftarrow \boldsymbol{V}$, i=1...n, **do**
2. **for** every element $w_j' \leftarrow \boldsymbol{V}^*$, j=1...m, **do**
3. Calculate matrix $\boldsymbol{C}$ by (7)-(8)
4. **end for**
5. **end for**
6. Using the matrix $\boldsymbol{C}$ and vector $\boldsymbol{\chi}$ to calculate SimS(y_i, X_i) by (6)

3.3. String Outlier Detection

In order to detect the string outlier data accurately by using semantic and structure similarity, we need to construct a new distance measurement mechanism, which can combine the similarity to calculate and judge the outlier data. In this subsection, we introduce the similarity cell to detect string outlier data.

From (1), (3), (4) we can find that y can be mapped to two-dimensional space by using the similarity calculation. Moreover, in the two-dimensional space, distance can be used to detect outlier data. Therefore, we construct a similarity cell. In this cell, the similarity is the important criterion of string outlier detection to be embodied in similarity cell. It is shown in Figure 1.

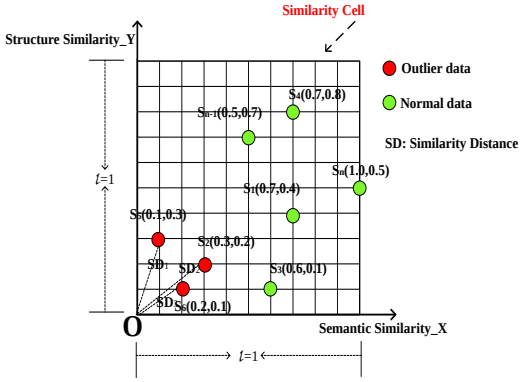

Figure 1. String outlier detection in similarity cell

The general idea of string outlier detection is illustrated in Figure 1. In the similarity cell, the origin **O** denotes the semantic similarity $SimS(y_i, X_i)$=0, and structure similarity $SimC(y_i, X_i)$=0. The smaller the similarity, the closer the test sample is to the origin. In other words, the closer to the origin, the more likely it is outlier data. Therefore, we can use distance-based algorithm to measure the distance between origin and test sample. However, normal distance measurements do not combine the semantic similarity with structure similarity. Therefore, we adopt a new similarity distance measurement mechanism to address this issue. In Figure 1, either normal or outlier data is described by using similarity distance. The normal data are denoted by green points. Their similarity distances are all more than threshold. Otherwise, the test samples are outlier data (denoted by red points).

Given a test sample $y_i \in \boldsymbol{D^*}$ and a point x^* in similarity cell, the similarity distance $SimDist(y_i, x^*)$ may vary depending on specific application or dataset. According to l_p-norm distance [12], we define a new l'_p-norm similarity distance $SimDist(y_i, x^*)$ here.

$$SimDist(y_i, x^*) = \frac{\alpha \times SSim + \beta \times CSim}{\alpha + \beta}$$
$$= \frac{[\alpha \times (y_i.SimS(y_i, x_i) - x^*.SimS'(x^*, x_j))^p + \beta \times (y_i.SimC(y_i, x_i) - x^*.SimC'(x^*, x_j))^p]^{\frac{1}{p}}}{\alpha + \beta} \quad (14)$$

where p=1 is the ℓ'_1-norm, and p=2 is the ℓ'_2-norm (we use p=2 in our case). The parameter α is the semantic factor, β is the structure factor, and $\alpha > \beta$. *SSim* is the semantic similarity distance, and *CSim* is the structure similarity distance. Suppose x^* is the origin **O**, *x.SimS'* and *x.SimC'* are 0, so (14) can be converted to (15). That is shown in below:

$$SimDist(y_i, O) = \frac{(\alpha \times y_i.SimS(y_i, x_i)^p + \beta \times y_i.SimC(y_i, x_i)^p)^{\frac{1}{p}}}{\alpha + \beta} \quad (15)$$

Combining (1)-(4) with (15), if similarity distance is more than threshold τ_i, the test sample y_i is normal; otherwise it is the outlier data.

Algorithm 2 String Outlier Detection

Input: Training Data $X_i \in \boldsymbol{D}$, Testing Data Sample $y_i \in \boldsymbol{D^*}$, Origin **O**
Output: the label of y_i

1. **for** each i=1...M, **do**
2. Calculate $SimS(y_i, X_i)$ by (6) and $SimC(y_i, X_i)$ by (10)-(13).
3. Calculate similarity distance $SimDist(y_i, \boldsymbol{O})$ by (15).

4. **end for**
5. Calculate the self-adaptive threshold τ_i.
6. **for** each i=1...M, **do**
7. Detect outlier data by using (3)

In order to address the sensitivity problem of threshold, we calculate threshold τ_i by using the similarity entropy histogram. More details will be discussed in next subsection.

3.4. Dynamic Threshold Calculation

From another perspective, string outlier detection may be transformed into cluster searching. The clustering is better between test set $\boldsymbol{D}^*$ and training set $\boldsymbol{D}$, the test sample is more likely the normal data. Otherwise, it is likely the outlier data. Information Entropy can reflect the clustering's situation in whole space. In this subsection, we define the similarity entropy as in (16).

$$\hat{H}(y_i, x_i) = -SimS(y_i, x_i)\log_2(SimS(y_i, x_i) + m) \tag{16}$$

where m is the minimum factor (m=0.001), so that to avoid the situation when $SimS(y_i, X_i)$=0.

For different test set $\boldsymbol{D}^*$ and training set $\boldsymbol{D}$, the threshold should different. The given threshold is easy to affect the accuracy of outlier detection. Therefore, we construct the similarity entropy histogram to calculate threshold τ_i.

Firstly, for the test sample set $\boldsymbol{D}^*$, we can obtain the maximum and minimum similarity entropy $\hat{H}(y_i, x_i)_{\max}$ and $\hat{H}(y_i, x_i)_{\min}$ by using (16) when semantic similarities are not the same. We suppose the number of levels in histogram is L (L=10 in our case), the similarity entropy of k^{th} level is shown as (17).

$$H_k = \hat{H}(y_i, x_i)_{\min} + p(r_k), \quad k = 0,1,\cdots,L-1 \tag{17}$$

where $p(r_k)$ denotes the interval of similarity entropy between adjacent two levels. It is defined as in (18).

$$p(r_k) = \begin{cases} \dfrac{\left(\hat{H}(y_i, x_i)_{\max} - \hat{H}(y_i, x_i)_{\min}\right) \times k}{L}, & \hat{H}(y_i, x_i)_{\max} \neq \hat{H}(y_i, x_i)_{\min} \\ 0, & otherwise \end{cases} \tag{18}$$

According to L and H_k, we can construct the similarity entropy histogram. In this histogram, there are L intervals, and the sum of similarity entropies in same interval is its value in histogram. The intervals are (H_0, H_1), (H_1, H_2), ..., (H_{L-2}, H_{L-1}).

On this basis, we can calculate the expectation $E_{\hat{H}}(L_i)$ of similarity entropies by using similarity entropy histogram. The $E_{\hat{H}}(L_i)$ is defined as in (19).

$$E_{\hat{H}}(L) = \frac{\sum_{k=0}^{L-1}\left[\frac{\sum_{t=0}^{n-1} H_k(y_i, x_i)_t}{n+m} \times (L-1-k)\right]}{L \times L} \tag{19}$$

where n is the number of similarity entropy in k^{th} level, and m is the minimum factor.

Each semantic similarity value is corresponding to a similarity entropy. If one similarity entropy is nearest to $E_{\hat{H}}(L_i)$, the semantic similarity that corresponds to this similarity entropy is threshold τ_i. It is defined as in (20).

$$\tau_i = SimS(y_i, x_i) = \max(SimS) \tag{20}$$

where $SimS$ is a set. Because of (16), the same similarity entropy may correspond to different semantic similarities. So, there may several semantic similarity values in $SimS$. The threshold τ_i is the maximum semantic similarity value in $SimS$.

$$SimS = \arg\min(\hat{H}(y_i, x_i) - E_{\hat{H}}(L)) \tag{21}$$

If there is only one test sample in $\boldsymbol{D^*}$ or $\hat{H}(y_i, x_i)_{\max} = \hat{H}(y_i, x_i)_{\min}$, p(rk) =0, and (19) can be modified to (22).

$$E_{\hat{H}}(L) = \frac{\sum_{t=0}^{n-1} H_k(y_i, x_i)_t}{n+m} \times (L-1) \tag{22}$$

4. Experiment Evaluation

In this section, the similarity entropy-based self-adaptive string outlier detection method (SESAOD method) is compared with nested-loop algorithm(NL), cell-based algorithm(CB), partition-based algorithm (PB), outlier detection based on double distance (DTKA) and Likelihood of semantic outlier algorithm (LSO), so that to verify accuracy of method. All methods are performed by C# language (OS: Windows Server 2003; CPU: Core (TM) 2 Duo CPU T6570 2.1GHZ; Memory: 2G; Platform: Microsoft VS 2008, SQL Server 2005). The string data of different scales in client information are used to detect the outlier data. There are four datasets: 4000(2998 training samples, 1002 test samples), 8008(6008 training samples, 2000 test samples), 15000(11992 training samples, 3008 test samples), 21000 (15999 training samples, 5001 test samples). By several experiments, α =0.8 and β=0.2 are given.

For the results of different experiments, the common performance metrics: precision rate (PCR) and recall rate (RCR) [13] are used to estimate the performance of SESAOD method.

- Precision Rate (PCR): The percentage of correct data that is detected in detection result.

$$\Pr ecesion = T / P = T /(T + F) \tag{23}$$

- Recall Rate (RCR): The percentage of correct data that is detected in all correct result.

$$Recall = T / R \tag{24}$$

4.1. Threshold Detection

We select 4000, 8000, 15000 and 21000 samples to detect the rationality and accuracy of threshold selection. If τ_i is given in SESAOD method, the experimental results of different threshold are shown in Figure 2. From them we can find that, when τ_i is around 0.9, it is satisfactory in both detection results and performance metrics, so the given threshold τ_i can be 0.9. However, for the samples of different scales, experimental results indicate the results may better by using the dynamic threshold. A part of results are shown in Table.1 and Table. 2.

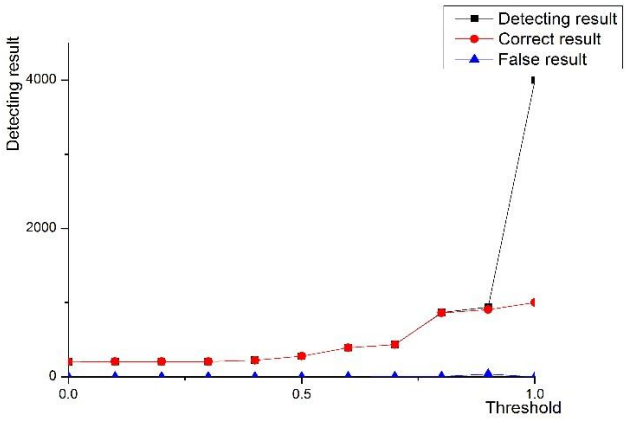

Figure 2. The results of different thresholds in 4000 samples

Table 1. A Part of Comparison Results of given and dynamic threshold for 4000 samples

τ_i	Detection Result	Correct Result	PCR	RCR
0.7	435	434	99.77%	43.31%
0.8	867	862	99.42%	86.03%
0.882	**996**	**958**	**96.18%**	**95.61%**
0.9	941	906	96.28%	90.42%

Table 2. A Part of Comparison Results of given and dynamic threshold for 8000 samples

τ_i	Detection Result	Correct Result	PCR	RCR
0.7	888	885	99.66%	44.25%
0.8	1886	1802	95.55%	90.10%
0.9	**1905**	**1860**	**97.64%**	**93.00%**

In Table 1 and Table 2, we found that when the dynamic threshold τ=0.882 and τ=0.9, the detection results and performance metrics may better than given threshold's results, and dynamic threshold is more precise. The results of 21000 samples show the same conclusion. Therefore, by using the dynamic threshold we can reduce the sensitivity problem of threshold and ensure the accuracy of outlier detection.

4.2. Outlier Detection Results

In Figure 3-Figure 5, we compare our method with existing methods to estimate the performance of SESAOD method.

In Figure 3, the result of SESAOD method is nearest to the correct result. Because of the single distance measurement, some normal data may be considered as the outlier data, the result of PB and DTKA algorithms are more than correct result. In Figure 4, the median of semantic distribution is treated as threshold for the LSO algorithm. With the change of semantic distribution, the median has a great influence on the threshold, so the precision of LSO algorithm is descending, and the change is obvious. For the SESAOD method, not only is the detection result near the correct result, but also the count of correct outlier samples; so the precision rate of SESAOD method is highest.

Only precision rate can't indicate the accuracy of method completely, the recall rate is also the performance metrics. According to the results in Figure 3 and Figure 4, we can calculate the recall rate for different scale samples, and the results are shown in Figure 5. Because the count of correct outlier samples in detection result is nearly correct, the recall rate of SESAOD method is highest.

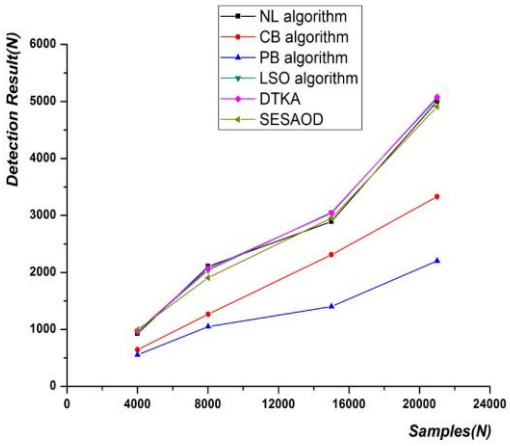

Figure 3. The detection results for the samples of different scales

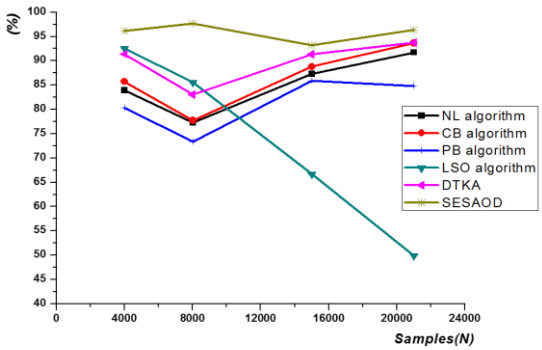

Figure 4. PCR for the samples of different scales

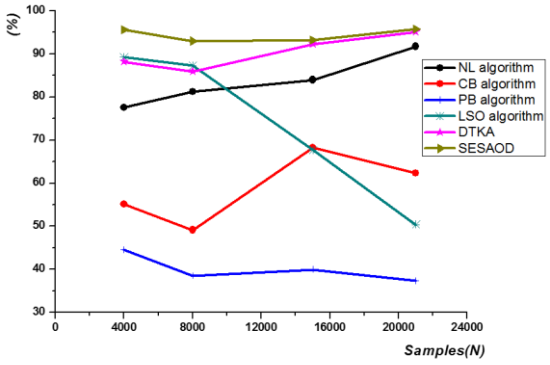

Figure 5. RCR for the samples of different scales

In Figure 6, because of the matrix computation based on the word matching in SESAOD method, it can speed up to calculate the similarity. But histogram statistic needs extra time. In general, our presented method is more efficient than other algorithms except DTKA. However, ours outperforms other algorithms as shown above.

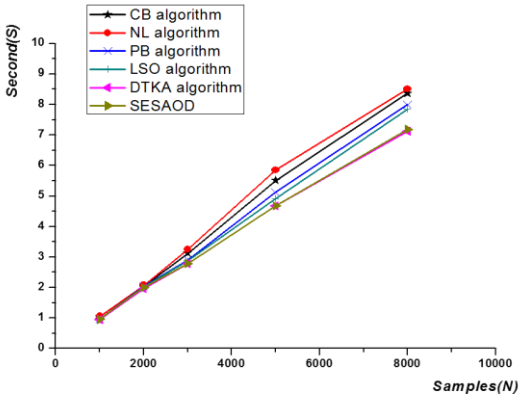

Figure 6. Speed comparison with different methods

5. Conclusion

In this paper, we propose a similarity entropy-based self-adaptive string outlier detection method to improve the accuracy. In the method, we use the matrix computation to calculate the semantic similarity, and use the similarity entropy histogram to determine the dynamic threshold, so that to reduce the sensitivity problem of threshold. The experiments show favorable results when compared with above other methods. In the future, we will use the semantic relationship to reflect in our method, so that to improve the accuracy of outlier detection.

Acknowledgements

This research was supported in part by Shannxi Provincial Department of education special scientific research project (No.16JK1505).

References

1. Nurunnabi A, West G, Belton D, "Outlier detection and robust normal-curvature estimation in mobile laser scanning 3D point cloud data", *Pattern Recognition*, vol. 48, no. 4, pp. 1404-1419, 2015.
2. Barnabe-Lortie V, Bellinger C, Japkowicz N, "Smoothing gamma ray spectra to improve outlier detection", in *Proceedings of the International Conference on Computational Intelligence for Security and Defense Applications*, pp. 1-8, 2014.
3. Pardo M C, Hobza T, "Outlier detection method in GEEs", *Biometrical Journal*, vol. 56, no. 5, pp. 838-850, 2014.
4. Hido S, Tsuboi Y, Kashima H, et al, "Statistical outlier detection using direct density ratio estimation", *Knowledge and information systems*, vol. 26, no. 2, pp. 309-336, 2011.
5. Zhang Y, Hamm N A S, Meratnia N, et al, "Statistics-based outlier detection for wireless sensor networks", *International Journal of Geographical Information Science*, vol. 26, no. 8, pp. 1373-1392, 2012

6. Marateb H R, Rojas-Martínez M, Mansourian M, et al, "Outlier detection in high-density surface electromyographic signals", *Medical & biological engineering & computing*, vol. 50, no. 1, pp. 79-89, 2012.
7. Kontaki M, Gounaris A, Papadopoulos A N, et al, "Continuous monitoring of distance-based outliers over data streams", in *Proceedings of the International Conference on Data Engineering*, pp. 135-146, 2011.
8. Cassisi C, Ferro A, Giugno R, et al, "Enhancing density-based clustering: Parameter reduction and outlier detection", *Information Systems*, vol. 38, no. 3, pp. 317-330, 2013.
9. Orair G H, Teixeira C H C, Meira Jr W, et al, "Distance-based outlier detection: consolidation and renewed bearing", in *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 1469-1480, 2010.
10. Zhen Yang, Minghui Zhang, "Research of algorithm formining outlier based on double distance application in coal mining", *Manufacturing Automation*, vol. 5, pp. 40-42, 2013.
11. Shicai Fan, "The Outlier Detection Based on Semantics", *Inner Mongolia Coal Economy*, vol. 7, pp. 19-21, 2011.
12. Yang Cong, Junsong Yuan, Yandong Tang, "Video Anomaly Search in Crowded Scenes via Spatio-Temporal Motion Context", *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 10, pp. 1590-1599, 2013.
13. Li Guo-Hui, Du Xiao-Kun, Hu Fang-Xiao, Yang Bing, Tang Xiao-Hong, "Structure Matching Method Based on Functional Dependencies", *Journal of Software*, vol. 20, no. 10, pp. 2667-2678, 2009.