The Conception of Strings Similarity in Software Engineering

Sergey Frenkel(✉) and Victor Zakharov

Federal Research Center "Computer Science and Control" Russian Academy of Sciences, Moscow, Russia
fsergei51@gmail.com, VZakharov@ipiran.ru

Abstract. Many tasks of modern software engineering, such as malware detection, attack recognition, Web Caching and Prefetching, etc., are based on the concept of distance between various data sets, e.g. between the strings of symbols. Such distances should express "similarity" between various data, e.g., the degree of similarity (or dissimilarity) between a suspected program and benign software.

In our previous works, some inequalities have been obtained that describe upper and lower bounds on Normalized Edit Distance (NED) values in terms of the Jaccard distance.

In this paper, based on this result we suggest and study the Averaged Normalized Edit Distance (ANED) as a new similarity metric which can be useful in classification-via-clustering problems. We show that ANED has well-interpreted properties, on the base of which it is possible to define a metric subspace on the strings space. The ANED based approximation can be used for various areas of data clustering, but in this paper we demonstrate the experiments showing the relevance of our approach to malware clustering for their detection issues. Traces used in our experiments come from the KVM hypervisor Runtime Execution Introspection and Profiling (REIP) system based on Virtual Machine Introspection (VMI) techniques to profile hooked Windows API calls.

Keywords: Similarity and distance · Software design for security · Malware detection

1 Introduction

The problem of assessing the similarity of data sets for classification purposes one way or another appears in various problems of informatics, such as web Cognitive Load analysis [1], web caching and Prefetching [2], web context analysis, malware detection, attacks recognition [3], etc. For the most part the similarity estimation is based on the concept of distance between the data sets, which expresses the degree of similarity (or dissimilarity), e.g. similarity of a suspected program to benign software [3]. Presently various similarity metrics were suggested and used in data classification through-clustering tasks [4], one of the most discussed is Jaccard distance J_D (or easily connected with J_D Jaccard similarity Index (JI) $J_S = 1 - J_D$) measure. Another metric is the Edit Distance (ED), namely,

A. Kalenkova et al. (Eds.): TMPA 2019, CCIS 1288, pp. 56–67, 2021.
https://doi.org/10.1007/978-3-030-71472-7_4

the minimal number of edit operations (delete, insert and substitute of a single symbol) required to convert one sequence (string) to the other [4, 13]. In order to normalize the ED to interval (0,1), the Normalized Edit Distance (NED) is often used [5, 6, 13].

The computational complexity of the NED is high, e.g., in comparison with the commonly used Jaccard distance. Moreover, when computing Jaccard measure, one can employ several approximation techniques, such as Locally-Sensitive hashing with MinHash [4], to dramatically speedup the clustering, classification and identification, what is absent for NED. Nevertheless, despite these difficulties, now ED/NED is seen as a highly desirable measure of similarity (distance) in such areas as optical character recognition, text processing, computational biology, cryptography etc., as it, from the point of view of the Machine Learning community, is a more acceptable measure for such complex structured information [7]. Note that ED-or-NED are often used not only as a characteristic of proximity, but as the cost of automatically converting one line to another for automatic language recognition/classification [8].

The analysis of the literature shows that although the researchers try to consider semantics in the tasks of assessing the similarity of texts [9], often in the tasks of programs developments, in particular, in the tasks of malware detection, the "similarity" regarding the difference of the data (e.g., in the benign-malicious behavior estimation) on an intuitive level only is considered, and accordingly, detection is carried out on the basis of numerical characteristics that have no obvious links with the property of being similar [9].

In the paper, we analyze some informal (and some formal as well) inferences from using of Jaccard-based similarity measure suggested in [10] which is based on JI approximation and reflects some properties of NED. We show, that under some insignificant updating suggested measure, it receives some property, enabling to reflect to some degree semantics of the compared string forms. The reason of why we can consider a relation between such different measures as Jaccard and NED is that the scope of our analytical results is so-called the representing strings, which is a result of original (raw) textual data shingling [4], taking into account that there is a solid evidence that these similarity estimation results can be applied to raw strings that have representation by n-gram with low repetitions (more explanations see in Sect. 3).

In general, the area of this paper refers to the range of such tasks for assessing and using similarity measures for which the effectiveness of using edit distance can be justified. Thus, what we do in this paper is to analyze to extent to which the applied approximation allows one to reflect the semantical differences of two strings.

Briefly, the contribution of this paper in comparison with [10, 11] is:

- it is shown that the average value obtained by averaging over the interval of possible NED values has specific properties that are different from both JD and NED, namely, the possibility to take into account the explicit dependence on the difference in sizes of the compared sets, the possibility of using the property of triangle inequality for clustering different subsets of strings,
- the relationship between the values of the true values, and the approximate values of JD, NED and their approximate estimates is shown, which can be useful when choosing threshold values for the conditions for assigning to clusters,

– we show, in fact, that well-known similarity measures, formally calculated without using any conditions for the semantics proximity of the objects being compared [9], can be transformed into measures more sensitive to semantic differences, with computational complexity similar to Jaccard metrics, with ability to reflect (a greater extent) the semantics differences of the compared data.

The rest of the paper is organized as follows. Section 2 is an analysis of the most popular metrics from the point of view of their ability to reflect the semantical differences of the data compared. Note that talking about "semantics" in this paper we mean simply the applied properties and goals of the compared data, without using any formal definitions of semantics (see, e.g. [12]). For example, if we deal with malware detection problem, we must think about how a similarity measure used for discrimination between malicious and benign behavior and results of their execution.

Section 3 explains and analyzes the measure of similarity suggested in [10]. A way to overcome the triangular inequality violation of traditional NED is shown. The technical aspects of the similarity measures computation are considered in Sect. 4. The results are discussed in the Sect. 5 and in Conclusion.

2 Data Similarity Conception

Before solving the problem of approximating a NED, let us consider what basic requirements the similarity estimates should meet. First of all, it must be based on a well-defined mathematically notion distance in a space. Formally, distance ("distance in a space") is a function D with nonnegative real values defined on the Cartesian product $X \times X$ such that $D: X \times X \to R^+$. It is called a distance metric on X if for every $x, y, z \in X$:

$D(x, y) = 0$ iff_$x = y$ (the identity axiom);
$D(x, y) + D(y, z) \geq D(x, z)$ (the triangle inequality);
$D(x, y) = D(y, x)$ (the symmetry axiom).

A set X, which is provided with a metric, is called a metric space.

The similarity S(x, y) metrics considered as an inversion to the distance notion which must follow these rule, but be greater, the smaller the differences between the objects x, y, $S(x; x) > S(x; y)$, $x \neq y$, in particular.

Let us consider the case, when the data semantics imposes that the data to be some strings of ordered symbols. The traces of API system calls are an example.

From the practical viewpoint, the similarity definition problem is a combination of two subproblems: (i) what kind of similarity metric is most relevant to the compared data, and (ii) given a query string Q, how to use a similarity metric to search with suitable complexity (cost), in order to find all strings in a data set whose distances with Q is no more than a given threshold.

Let us consider two the most popular measures of similarity.

Jaccard Distance. The Jaccard distance J_D (or Similarity Index $J_S = 1 - J_D$) is often used for strings similarity estimation despite its intended use for simple (not multi-!) sets.

The Jaccard distance is $J_D(x, y) = |x \Delta y|/|x \cup y|$, where Δ denoted the symmetric difference between two sets x, y (that is x and y are considered as a unordered set of symbols from a given alphabet). Correspondingly, Jaccard similarity metrics $J_S(x, y) = 1 - J_D(x, y) = |x \cap y|/|x \cup y|$ is defind.

This metric can be interpreted as probability that random mapping by a hash-function h_i (different mappings for different i) do not repeat accidental collisions, that is probability $Pr(h_i(x_i = y_i)) = J_S(x, y) + (1 - J_s)/2^k$ that a random permutation of the subsets (substrings, in particular) produces the same values, k is the number of bits mapped by the hash-function hi. The meaning of this consideration is that the probability that the Minhash function [4] for two sets equals the J_S of those sets, therefore there is a clear interpretation of similarity. It is important that J_S is a true metric in the space of sets with such distance, as the triangle inequality holds. This is why it may be effectively used in clustering algorithms. Moreover, in spite of obvious violation of the string's semantic, the using of Jaccard similarity is rather successful for clusterization of traces for malicious code detection [9]. The appropriate result in these applications is possible if the main difference between malicious and benign codes is the composition of system calls and their parameters.

However, in many cases, in view of the triviality of connection (noted above) between the similarity of J_S (or J_D) and the structure and semantics of program behavior data displayed in traces, an incorrect detection of the consequences of attacks is possible. For example, in the very topical problem of detecting Replacement Attacks [3], J_S of two traces can incorrectly reflect the change of control graph (representing dependencies between the system calls in the traces of the program execution [11]) because it takes into account the difference in the number of systems calls only but not the sequence of their interactions, since a significant change in the structure of the traces under the influence of attacks can only slightly change the value of J_S.

Most frequently, the text files are shingled into q-grams (sequences of q tokens/terms from the text) [4, 14], see an example in the Sect. 3, therefore, the distances/similarities are considered relatively to q-grams. That is Jaccard index on shingle sets S(d1), S(d2): $J_S(d1, d2) = |S(d1) \cap S(d2)|/|S(d1) \cup S(d2)|$ is used, where d1, d2 are the texts compared.

Edit Distance. Edit (Levenshtein) distance (ED) [13] takes into account (to a certain extent) the structure of compared symbol strings. It takes into account the location in a trace where the characters do not match (number of "insertion" edit operations), the location where the symbols of one string are missing in the other (number of "deletes" edit operations), and reflects more correctly, for example, the fact that with a given class of attacks, a slight change in the types of system calls (and, accordingly, a slight change in J_S) leads to a significant change in ED due to a change in the trace of structure. Thanks to this, it, for example, increases stemming from the input strings being "repetitive", which means that many of their substrings are approximately identical, while J_D may be insensitive to such specific features of the structure.

This string similarity metric captures both similarities in the overall structure of the two sentences being compared as well as some similarity between different word forms [15].

But in general, ordering of objects in the strings compared (e.g., the ordering of web objects in web caching and prefetching prediction task [2]) is not explicitly reflected in

the ED. Further, although ED computes the distance for string of different lengths, the degree of influence of differences in these lengths on the value of ED is not reflected in any way in calculation models. Besides, the above-mentioned ability to interpret J_S as the probability of the hash coincidence of the two sections of the two compared traces, giving the possibility of a clear interpretation of their "similarity", is impossible for the ED.

So-called Normalized Edit Distance (NED) can enhance to some degree these aspects of ED using.

Normalized Edit Distance as Similarity Metric. Normalized Edit Distance of two strings x, y is [6]:

$$\mathrm{NED} = \mathrm{ED}(x, y)/\max(|x|, |y|) \tag{1}$$

and $\mathrm{SimNED}(x, y) = 1 - \mathrm{ED}(x, y)/\max(|x|, |y|)$, where SimNED(x, y) is Normalized ED Similarity.

That is, a perfect match will have SimNED(x, y) of 1.0, and completely dissimilar strings will be assigned a value of 0.0.

As it can be seen from (1), NED can be interpreted as a probability, that number of the transformation of the maximal (of two) strings (as well as minimal string to maximal one) requires ED(x, y) edit operations.

However, strictly speaking, there is no the effective hashing algorithm to allow an interpretation of the probability as the probability of hash values (like for the J_S) [14], and the computation of ED and NED is time-consuming $O(n^2 \mathrm{loglogn}/\log^2 n)$, while there are efficient hashing based linear algorithms for approximating Jaccard distance for large data sets. But, what is important, the specific of ED computation (complexity increasing) is that in contrast to Jaccard (or Hamming) distance/similarity, when string comparison consists only in comparing string characters (without regard to their position, as in Jaccard, or standing at the same places in the strings, as for Hamming distance), it is necessary to consider the alignment operations, as associated with the requirement of minimal number of editing operations. This, in turn, ensures that a much larger specificity of the structure of strings is taken into account in terms of their similarity.

There is also the problem that in contrast to the J_D, the normalized edit distance NED does not satisfy to the triangle inequality, what may prevent computation-effective clustering of the maliciousness (or benign, depending on algorithm of machine learning based detection). Although so-called Generalized Edit (Levenshtein) Distance (GLD) was suggested [16], for which the triangle inequality is fulfilled, however, its calculation requires the selection of weights for the cost of performing editing operations, which can significantly increase the computational cost, which is significantly higher compared to J_D.

3 Jaccard Distance-Based NED Approximation

Let us assume that we know a Jaccard similarity (or distance) between strings x and y, which are considered as two sets of symbols corresponding their plain texts. How

could we approximate normalized edit distance NED(x, y)? An obvious hurdle in the technique suggested, namely, using Jaccard as the basis for the approximation of NED is that these metrics are based on different mathematical concepts. Jaccard is defined over (unordered) sets, in which each different element appears only once, despite that it may occur many times in different parts of the set (document, in particular). Edit Distance is defined over strings and depends on the order of the symbols in the underlying strings.

In order to overcome the difficulties associated with this discrepancy, we confine the argument to certain types of sets and strings, both derived from the original documents (plain texts of the traces, in the case considered); the documents in question go through a shingling process (which collects all the substrings of certain length of appearing in the document), which is the first necessary stage in most of modern methods of similarity estimation [4, 9, 10]. The outcome of the shingling process is sets of n-grams (without repetitions), which will be used for computing Jaccard similarity (distance). Then, we create representing strings of the sets by sorting and concatenating their elements according to, say, lexicographic ordering, as described in Sect. 3.9.2 in [4]. As a result, we get strings of n-grams (string over the alphabet of the n-grams) that are sorted and has no repetitions.

For example, 3-gram of a fragment of API system calls trace, CreateFile, is.

`Cre rea eat ate teF e Fi Fil ile.`

These representing strings will be used for NED estimation. As it was shown in [10], the difference between NED on pairs of original texts (strings) and NED of their n-grams representation is decreased very fast as a function of the n-gram size, which proves the possibility to use the representing strings instead of the original texts.

Our experiment results showed that it is possible to choose n-grams (3-gram and more, we used up to 13-gram) that yield better than 7% average difference between the NED over the original documents, and the NED over the representing strings.

Thus, it justifies our choice to concentrate in analyzing the representing string as we do in the sequel. We note that in general, one may sample a given data set and tune the length of the n-grams for the given data set, taking into account the correspondence between the original document and representing strings, and then to proceeding with the clustering of the representing strings. Thus, such consideration allows to consider any data set as a string, and correspondingly, to define the problem of Jaccard based expression of NED.

This result is understandable as the more n-grams size the more symbols must be inserted/deleted/substituted on the same way as it requires ED computation algorithm for the plain text.

In [10] we received inequalities for the NED in terms of Jaccard metric that impose upper and lower bounds on the NED values:

$$1 - \alpha \leq \mathrm{NED}(x, y) \leq (1 + \alpha)(J_D(X, Y)/(2 - J_D(X, Y)))$$

X, Y means the set of symbols, contained in the strings x, y (recall that we deal with representing strings {x, y} obtained from original (raw) strings, that is J_D is distance between corresponding n-grams (Sect. 2), $\alpha = \min(|x|, |y|)/\max(|x|, |y|)$.

Let us average NED over the interval $[1-\alpha, (1+\alpha)J_D(x,y)/(2-J_D(x,y))]$ (assuming the uniform NED distribution within this interval). Then we received the averaged NED depicted as ANED:

$$\mathrm{ANED}(x,\ y)\ =\ (1+\alpha(J_D(X,\ Y)\ -\ 1)/(2\ -\ J_D(X,\ Y)) \tag{2}$$

Leaving for now aside the question of the accuracy and usefulness of this averaging from the point of view of using strings for classification (for example, the traces classification as malicious and benign programs), the first significant result is that we express the average NED value through the values J_D which are computed by the hashing mentioned above.

ANED term α takes into account such an important factor of the editing distance as the fraction of characters that you need to "insert\delete\replace" to convert the string x to y (or vice versa). Accordingly, from the point of view of the program execution semantics, the magnitude of the similarity metric is not simply reduced to the ratio of the number of characters coinciding in them (n-grams, in particular), as is the case in the Jacquard distance metric.

The relationships between the ratio of the pair strings length, their Jaccard distance and the ANED are represented in Fig. 1.

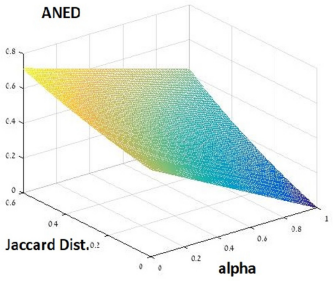

Fig. 1. Relationship between Average Normalized Edit Distance, ratio of strings pair α, and Jaccard Distance.

One of the very important Jaccard metric properties is *(d1, d2, P1, P2)-sensitivity* [15], that is:

if $J_D(x, y) \leq d1$ then Prob $[h(c) = h(y)] \geq P1$,
or, for Jaccard similarity:
$J_S(x, y) \geq d2$, then Prob$[h(x) = h(y)] \geq P2$.
where h(x), h(y) –hash-function implementing given permutation.

This property provides the ability to use the MinHashing algorithm to a good approximation of the estimate of the similarity of two sets [4]. Obviously, due to the uniformity relationship between the average NED and JD, we can find that our average Normalized editing distance (ANED) is also (d1, d2, P1, P2) -sensitive, which also indicates possibility of approximation based on LSH.

3.1 About Triangular Inequality for NED

As noted above, the triangle inequality does not hold for NED (unlike J_D and J_S), i.e. set of strings S with a given NED (also with SimNED) do not form a metric space. It means that for each strings x; y; z ∈ S, it can be: SimNED(x, y) + SimNED(y, z) ≤ SimNEDD(x, z). Taking into account the formulae (1) we can formulate the requirement that the subset {x, y, z} be a metric space with the metric SimNED(x, y):

$$1 - \mathrm{ED}(x,\ z)/\max(|x|, |z|)\ \leq\ 2 - \mathrm{ED}(x,\ y)/\max(|x|, |y|) - \mathrm{ED}(y,\ z)/\max(|y|,\ |z|) \tag{3}$$

We can rewrite the condition (3) relatively ANED (2) in an obvious way and can see that regarding ANED, a similar analysis for formula (2) shows that there are (continuous) regions {α, *JD*}, where, calculating $J_D(x, y)$, $J_D(y, z)$, $J_D(x, z)$ (and corresponding "alphas" for each pairs) we can find different subsets of triads {x, y, z} for which the triangular inequality is true. Note, that in practice the fact that for most pairs of traces there are always natural signs of their disagreement, for example, the J_S about zero (α is considerably less that 1 as well*)* allows us to exclude a significant number of cases from consideration, and increase the proportion of triads satisfying the triangle rule (see Sect. 4).

Correspondingly, it is possible to implement the effective clustering with ANED as a distance metric, say, using K-nearest Neighbour algorithm.

4 Similarity Model Validation

Now we demonstrate the rationality of our view on data set similarity estimation on an example dealing with traces of malicious programs recognition mostly represented in [10].

4.1 About Data Set

The data set is the records of the Windows API system calls of malware including a) the timestamp; b) the function name; c) all parameter values, and d) the return value.

We considered the traces subset (gathered in Taiwan National University [10]) focused on important and significant Windows API calls related to a) Files and I/O (Local file system), b) Windows System Information (Registry), c) Processes, and d) Dynamic-Link Libraries (DLLs). The order of the API system calls is perfectly preserved. There was access to two sets of malware traces and one set of benign traces are ready. The set has 272 malware samples (which fork 419 processes). According to VirusTotal, their first-seen dates were from August 2009 to October 2014. The benign data set contains about ten software (such as IE, Paint, Calc, CMD) of Windows XP and Win7's built-in software.

4.2 Similarity Metrics Measurement Issue

Similar traces can be grouped together using Locality Sensitive Hashing (LSH) in linear time with only a small increase in false negative results, hashes items into buckets several times, such that:

– similar items are hashed into the same bucket with high probability,
– items that are not similar enough are hashed into a common bucket with low probability.

Hence, there is a benefit of using a large number of buckets for maximizing the probability of collision of similar items.

4.2.1 Similar Traces Finding

In accordance with LSH technique [4] items that are mapped to the same bucket are considered as candidates for being similar. But there are no any strong methods to compute probabilities of real semantically-grounded similarity. In fact, J_S is just the probability that LSH maps two Jaccard-similar traces in the same bucket. When computing Jaccard measure, one can employ several approximation techniques, such as MinHash, to dramatically speedup the clustering, classification and identification.

The use the ANED estimation allows us to supplement the clustering technique outlined in the next section, by a scheme, where LSH provides Jaccard similar strings (traces) in the same clusters, (that allows us to check the NED for any item in the cluster, without accurate ED computation), and another technique we use is MinHashing [4] which is a compression method for sets of items that preserves the Jaccard similarity, that allows to work with much shorter same-length MinHash signatures.

5 Experimental Results of ANED-Based Approximation and Their Discussion

Figure 2 contains the main validation and explanatory data on topic of this paper. These results were obtained for 55 pairs of malware trace by LSH with Minhash [10].

For improved efficiency the text items are MinHashed into signatures, then LSH is performed on these signatures (integer vectors) using the banding technique [4].

First of all, note that ANED was computed by formula 2 not by accurate J_D values, but via its LSH-Minhash approximation. Certainly, this is more interesting from the practical point of view as LSH with Minhash allows reducing essentially J_D computation cost, that meet to the requirement to reduce NED cost computation as much as possible. It can be seen, that the behavior of ANED (regarding the pairs of compared trace s and ratios of their lengths α), computed by suggested approximation by ANED (star line), is the same the accurate NED values (solid line) in terms of increasing and decreasing values of both variables relative to the numbers of pairs and values to the ratio of their lengths. Moreover, Fig. 2 shows a rather good approximation of the NED by ANED.

Let's see how the given data allows to evaluate the fulfillment of the basic properties of NED in its approximation of ANED, and also how it allows understanding some

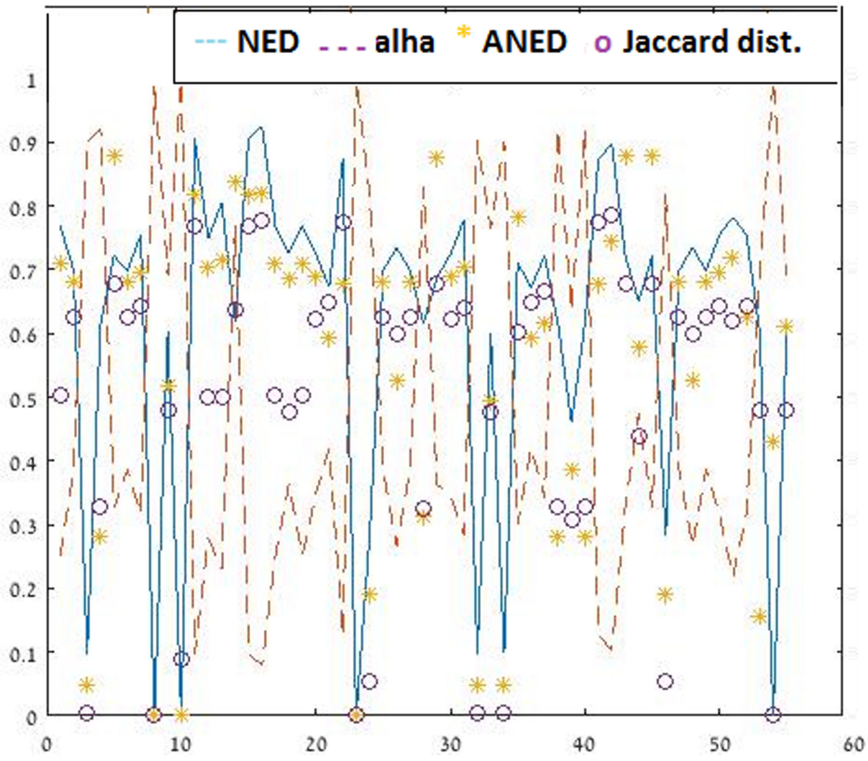

Fig. 2. J_D-based approximation of Normalized Edit Distance by ANED calculated through LSH estimations of J_D given the relationship between length of strings α.

possibilities to display certain properties of the semantics of words contained in the compared lines in ANED. Since, for strings of approximately equal length, the number of operations required to convert one string to another should be greater than the number of mismatched words (characters) in both strings (since, replacing one character with another (operation "substitution") also requires the operation "delete"), it is obvious that $J_D \leq$ NED. As we can see for Fig. 2, for the pair of traces with lengths for pairs of strings whose lengths are not dramatically different from each other (say, $\alpha > 0.8$) this relation also holds for ANED, in spite of that for ANED computation by formula (2) LSH Minhash approximations of J_D were used, not the J_D exact values. It means that the use of ANEDs corrects the situation when the use of Jaccard for sets with the same character set but organized as strings of different lengths gives a zero distance value, i.e., a complete match. At the same time, computational costs are equivalent to Jaccard computational cost.

As mentioned in Sect. 2, the main question of our study is to preserve the semantics of the similarity of the ED based measures despite the use of J_D for ANED computation. For example, API system calls "RegQueryValue" is often called sequentially in the Windows program and without taking into account the values of its arguments, it may not be possible to compare the equivalence of the traces of two different programs to

detect possible malicious behavior. In this case, the parameters may differ more than a few characters, and the Jaccard distance for their n-gram representation can be close to zero, while the NED will give significantly higher distance values, i.e. the probability that one string (trace) transforms into another with probability which is equal to NED. As a result, using ANED can provide clustering that learns from the malicious dataset without any explicit descriptions of each malware or its class. Each hash table bucket, obtained during J_D computing as it mentioned above, is selected and used as its representative, then the binary (malicious/benign) decision by comparing each trace query against all medoids m_1, m_2,.., m_k (corresponding to the buckets mentioned above). If its maximal similarity to one of the medoids exceeds a predefined threshold t chosen as $\max_{i=1,..,:k}$ (ANED), then it is classified as malicious, otherwise it is classified as benign.

Some examples of buckets with computed average value of Jaccard and NED distances (as a threshold to be include in the bucket) and corresponding NEDs of shingled and original texts:

Bucket #290, has 6 traces: $J_D = 0.320966$, NED = 0.320051, NED(Orig. Text) = 0.290116, Bucket #437, has 5 traces: $J_D = 0.575048$, NED = 0.575708, NED(Orig Text) = 0.613325.

6 Conclusion

Many approaches to the similarity of different symbolic structures estimation are based on edit Edit Distance notion. In this paper we showed that the average value obtained by averaging over the interval of possible NED values has specific properties, namely, the possibility to take into account the explicit dependence on the difference in sizes of the compared sets, the possibility of using the property of triangle inequality for clustering subset of strings, in dependence on ratio of their length and the mutual features of pair of strings, expressed by Jaccard distance. That is the pairs $(\alpha, J_D(x, y))$ can characterize some semantical important properties of the string pairs, e.g., that the similarity of two traces x, y, is less than simple fraction of coincided symbols, as it takes place in the Jaccard distance metric. It means that for the tasks for which the effectiveness of using edit distance-based similarity have been justified, the edit distance-like measure can be transformed into measure more sensitive to semantic differences, with computational complexity like to Jaccard.

Acknowledgements. Research partially supported by the Russian Foundation for Basic Research under grants RFBR 18–07-00669, 18–07-00576 and 18–29-03100.

References

1. Tracy, J.P.: Measuring cognitive load to test the usability of web sites. University of Memphis, Memphis, USA (2007)
2. Kallurkar, P., Sarangi, S.: pTask: a smart prefetching scheme for OS for intensive applications. In: 49th Annual IEEE/ACM - International Symposium on Microarchitecture (MICRO-49), 15–19 October 2016, pp.1–12 (2016)

3. Ming, J., Xin, X., Lan, P., Liu, D., Mao, B.: Replacement attacks: automatically impeding behavior-based malware. In: Malkin, T., Kolesnikov, V., Lewko, A., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 497–517. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_24
4. Leskovec, J., Rajaraman, A., Ullman, J.D.: Mining of Massive Datasets. Cambridge University Press, Cambridge (2014)
5. Vidal, E., Marzal, A., Aibar, P.: Fast computation of normalized edit distances. IEEE Trans. Pattern Anal. Mach. Intell. **17**(9), 899–902 (1995)
6. Abdulalah, N., Arslani, I.: Efficient algorithms for normalized edit distance (2000). https://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=00C048E4B2B4BD190985960DC69FED7F?doi=10.1.1.63.8070&rep=rep1&type=pdf
7. Kim, C.W.: NtMalDetect: a Machine learning approach to malware detection using native API system calls. arXiv:1802.05412v2, 19 (2018)
8. Chakraborty, I., Das, D., Goldenberg, E., Koucky, M.: Saks, V.: Approximating edit distance within constant factor in truly sub-quadratic time. arXiv:1810.03664 (2018)
9. Jang, J., Brumley, D., Venkataraman, B.S.: BitShred: feature hashing malware for scalable triage and semantic analysis. In: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, 17–21 October 2011, pp. 309–320 (2011)
10. Dolev, S., Ghanayim, M., Binun, A., Frenkel, S., Sun, Y.S.: Relationship of Jaccard and edit distance in malware clustering and online identification. In: Proceedings of NCA2017, pp. 369–373 (2017)
11. Frenkel, , Zakharov, V.: Brief Announcement: Graph-based and probabilistic discrete models used in detection of malicious attacks. In: Dinur, I., Dolev, S., Lodha, S. (eds.) CSCML 2018. LNCS, vol. 10879, pp. 184–187. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-94147-9_15
12. Dennal, A., Benslimane, S.M.: A New measure of the calculation of semantic distance between ontology concepts. Int. J. Inf. Technol. Comput. Sci. **7**, 48–56 (2015)
13. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions and reversals. In: Soviet Physics Doklady, pp. 10–707 (1966)
14. Das, S., Pakray, P., Gelbukh, A.: Identifying semantic similarity using Levenshtein ratio. In: Proceedings of 10th International Workshop on Semantic Evaluation SemEval-2016, 1 January 2016, pp. 702–705 (2016)
15. Yuana, P., Wang, H., Chea, J., Ren, S., Xu, H.: Dechang approximate string similarity join using hashing techniques under Edit Distance constraints. J. Softw. **10**(9), 2721–2730 (2014)
16. Yujian, L., Bo, L.: A Normalized Levenshtein distance metric. IEEE Trans. Pattern Anal. Mach. Intell. **6**(29), 1091–1095 (2007)