Research the Stability of Decision Trees Using Distances on Graphs

N.D. Moskin[I], K.A. Kulakov[I], A.A. Rogov[I], R.V. Abramov[II]

[I] Petrozavodsk State University, Petrozavodsk, Russia
[II] ITMO University, Saint Petersburg, Russia

Abstract. The article deals with the problem of stability of classifiers based on decision trees for the problem of text attribution. Such a task arises, for example, in the study of the authorship of articles from the pre-revolutionary journals "Time" (1861–1863), "Epoch" (1864–1865) and the weekly "Citizen" (1873–1874). The texts were divided into separate parts of different sizes using the sliding window method, then the frequency of n-grams (encoded sequences of parts of speech) in each fragment was determined. Further, these indicators were used to build various classifiers. The resulting decision trees were compared with each other using the tree edit distance. For this purpose, a procedure for processing, comparing and visualizing graphs was implemented in the SMALT software package. As a result of experiments using different weights for editing operations, patterns were revealed between the parameters for constructing text fragments and the decision trees obtained on their basis.

Keywords: *text attribution, n-gram, decision tree, graph matching, tree edit distance, software complex "SMALT".*

DOI: 10.14357/20790279230111

Introduction

The main way to check the quality of a classifier is that it is checked against a control sample. If two classifiers showed similar results, it is not clear which one to choose. One option is to create a new control sample and conduct a new study. As a result, we get that the choice of a classifier significantly depends on the results on the control sample. When constructing and using a classifier in practice, much attention has to be paid to the quality (representativeness) of the training and control samples. It is very difficult. One of the existing ways out of this situation is to build an ensemble of classifiers. Sometimes it happens that each ensemble classifier is built on its own training sample. Its elements may contain fewer features than in the original data. The final classifier will be a function of the ensemble. Voting is often used as such a function. The element belongs to the class to which it belongs to most of the classifiers from the ensemble. Significant disadvantages of this method are the inability to justify the decision to classify and the complexity.

To eliminate these disadvantages, this paper proposes first splitting the ensemble of classifiers into clusters according to the degree of similarity, and then choosing the most typical (stable) classifier. Note that the training sample for this classifier will be representative. When training a classifier for a similar task, it will be possible to get by with one classifier, using a similarly constructed training set.

Experiments on the construction and comparison of classifiers were carried out on the material of the pre-revolutionary magazines "Time" (1861–1863), "Epoch" (1864–1865) and the weekly "Citizen" (1873–1874). It's a known fact that F. M. Dostoevsky was their editor. It means that he could have made his own edits to the texts of articles of other authors [1]. The features that distinguish one author from another are the frequency of the occurrence of certain n-grams (encoded sequences of parts of the speech).

1. Decision tree constructing using different samples

In the NLP domain it is essential to have as much data as possible. In order to fulfill those pre-requirements in the circumstances of restricted in a size corpus a few techniques might be considered. Usage of a sliding window is one of those.

Sliding window breaks one text into several chunks which are used further as data (fig. 1). The main idea is to enrich training set through partially duplicating already existing corpus. The following logic might be applied:

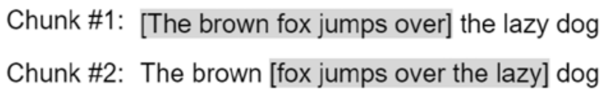
- N and S are chosen where N – sliding window size and S – skip window size.
- K chunks of the original text are created with a distance between start points of adjacent chunks equal to S.

You can calculate K (rounded to the nearest integer) using the following formula:

$$K = \frac{T-N}{S} + 1,$$

where T – size of a text that is being transformed. It is possible to vary both N and S. Basically they can be treated as hyperparameters. Varying N you can obtain more chunks with smaller size or less chunks with larger size.

Adjusting S governs amount of data as well as an intersection rate between two adjacent chunks. For example, a choice of N=1000 and S=100 will cause chunks #1 and #2 to have 900 words in common, chunks #1 and #3 – 800 words in common, etc. It is vital to choose this number high enough to save richness of the data although constructing a larger corpus.


Chunk #1: [The brown fox jumps over] the lazy dog

Chunk #2: The brown [fox jumps over the lazy] dog

Fig. 1. Parameters are N=5 and S=2. Sliding window results with 2 chunks.

The original training-test split was 80%–20%, thus 80% of data is used for training and the other 20% is used for testing algorithm and calculate metrics. Parts of speech are used as features and the purpose of a model is to predict an author of a given text using given features.

The described procedure was applied to a training part of a dataset to enlarge the existing corpus. After a decision tree [2] with default parameters from a Scikit-Learn [3] library is trained with built-in methods using the obtained training and test sets.

The default Decision Tree algorithm in Scikit-Learn is CART [4]. The model is represented as a binary tree with a certain criteria in every node. Prediction is made by going from the root of the tree to one of its leaf. The root is picked by meeting a condition that is represented in the node. A right path is picked for a true statement and a left one is for the false. For this task we use part of speech as data that represents an author, thus we have part of speech criteria in nodes. For example: a bigram "noun-noun" is met in the text more than 49 times?

The tree is built in a following manner:

1. Choose a node of the tree.
2. Calculate information gain for every feature and its threshold that splits data into two child nodes.
3. Choose a feature and its threshold for the node with a maximum purity.
4. Repeat until a stopping criteria is not reached.

The most popular impurity measure for CART algorithm is Gini index [5], which is defined as following:

$$I_{Gini} = 1 - \sum_{i=1}^{j} p_i^2,$$

where j is the total number of classes, p_i is the distribution of the i class in the node. Information gain is calculated as follows:

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right}),$$

where D_p, D_{left}, D_{right} – parent, left and right nodes respectively, N_i – number of data samples in node i.

2. Metrics on the set of trees

To compare trees, you can use the methods and algorithms developed within the graph matching direction [6; 7]. On a set of graphs a distance is set, which allows us to estimate how much these or other structures are "similar" to each other. One such distance is a measure based on editing operations (operations of insertion, deletion and replacement of vertices and edges in a graph) [8].

The function of error-correcting graph matching from $G_1=(V_1, E_1, \alpha_1, \beta_1)$ to $G_2=(V_2, E_2, \alpha_2, \beta_2)$ is called a bijective function $f: V_1' \to V_2'$, where $V_1' \subseteq V_1$ and $V_2' \subseteq V_2$. Cost of f define by the following formula:

$$c(f) = \sum_{u \in V_1'} c_{ns}(u) + \sum_{u \in V_1 - V_1'} c_{nd}(u) +$$

$$+ \sum_{u \in V_2 - V_2'} c_{ni}(u) + \sum_{e \in E_s} c_{es}(e) + \sum_{e \in E_d} c_{ed}(e) + \sum_{e \in E_i} c_{ei}(e),$$

where E_s, E_d and E_i are the sets of edges that are replaced, removed and inserted, respectively, and also:

- $c_{ns}(u)$ – the cost of replacing the vertex $u \in V_1'$ with $f(u) \in V_2'$;
- $c_{nd}(u)$ – the cost of removing a vertex $u \in V_1 - V_1'$ from G_1;
- $c_{ni}(u)$ – the cost of inserting a vertex $u \in V_2 - V_2'$ in G_2;
- $c_{es}(e)$ – the cost of replacing the edge e;
- $c_{ed}(e)$ – the cost of removing the edge e;
- $c_{ei}(e)$ – the cost of inserting the edge e.

At the same time, specific values c_{ns}, …, c_{ei} are selected depending on the specifics of the task being solved. Graph edit distance $d(G_1, G_2)$ is the cost of the optimal function f from G_1 to G_2:

$$d_1(G_1, G_2) = \min_f \left\{ c(f) : G_1 \xrightarrow{f} G_2 \right\}.$$

In other words, the graph edit distance $d_1(G_1, G_2)$ is the minimum total cost of editing operations that transform a graph G_1 into a graph G_2. Note that this distance can be used to compare arbitrary graphs, but there are measures designed specifically for trees. These include, for example, a distance based on tree alignment [9; 10]. To align two trees T_1 and T_2 (fig. 2), you need to insert empty vertices 0 into them so that the resulting trees T_1' and T_2' have the same structure (fig. 3).

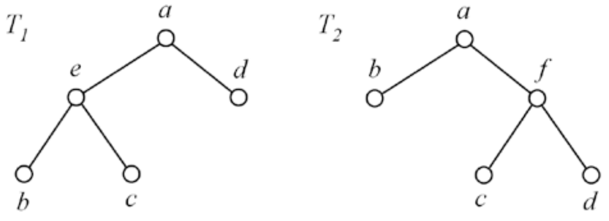


Fig. 2. Trees T_1 and T_2.

In this case, the labels of the corresponding vertices of the obtained trees T_1' and T_2' can be different. After that, we overlay trees T_1' and T_2' on each other and for each pair of vertices we calculate the measure of their difference μ_i. The sum $d_2(T_1,T_2) = \sum_{i=1}^{n} \mu_i$ will determine the distance between the trees if it is the minimum for all possible alignments of the given trees.

Next, consider four metrics for comparing trees, proposed in the work [11]. Let T_1 and T_2 be two trees with vertex set V_1 and V_2, respectively. A bijection $\phi : H_1 \to H_2$ where $H_1 \subseteq V_1$ and $H_2 \subseteq V_2$ is called an isomorphism of subtrees between T_1 and T_2 if ϕ preserves the adjacency relations between vertices and the connectivity of the compared subgraphs.

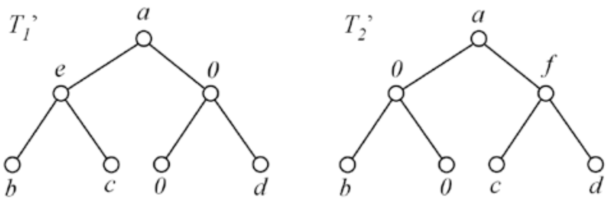


Fig. 3. Trees T'_1 and T'_2 resulting from alignment

Suppose that σ is some measure of similarity between the vertices of the compared trees T_1 and T_2, which, for example, can be based on the values of the attributes of these vertices. Then we define a measure of similarity $W_\sigma(\phi)$ between trees based on the isomorphism of subtrees ϕ and the measure σ:

$$W_\sigma(\phi) = \sum_{u \in H_1} \sigma(u, \phi(u)).$$

A subtree isomorphism ϕ_{12} is called a maximal subtree isomorphism between T_1 and T_2, if $W_\sigma(\phi)$ takes the greatest value among all possible subtree isomorphisms ϕ. For any two trees T_1 and T_2 define the following measures (here $|T|$ denotes the number of vertices in the tree T):

– $d_3(T_1, T_2) = \max(|T_1|, |T_2|) - W(\phi_{12})$;

– $d_4(T_1, T_2) = |T_1| + |T_2| - 2W(\phi_{12})$;

– $d_5(T_1, T_2) = 1 - \dfrac{W(\phi_{12})}{\max(|T_1|, |T_2|)}$;

– $d_6(T_1, T_2) = 1 - \dfrac{W(\phi_{12})}{|T_1| + |T_2| - W(\phi_{12})}$.

The proof of the fulfillment of the metric properties for can be found in [12]. Another measure of dissimilarity on a set of graphs, based on the biotopic distance of sets, was proposed in [13]. In particular, the author notes that this measure can be used to compare decision trees.

3. Implementation of processing and comparison of decision trees in SMALT

As part of the testing of algorithms and presentation of research results, the implementation of processing and comparison of decision trees in the SMALT information system was carried out. The information system "Statistical methods of literary text analysis" (SMALT) is intended for processing texts in pre-revolutionary graphics and conducting statistical research on texts. The system is focused on processing texts from the magazines "Time", "Epoch" and "Citizen" edited by F. M. Dostoevsky.

As part of the implementation, functions for working with the decision tree graph and functions for pairwise comparison of decision tree graphs were added. To implement the work with the decision tree graph, the "controller-model" pattern was used. The controller accepted the request, determined the required action, loaded the required model, and formatted the result. The following operations were implemented in the controller: loading a decision tree, editing decision tree metadata, viewing a decision tree, deleting a decision tree, viewing a list of loaded decision trees. Due to the large number of decision trees (for example, several decision trees with different steps can be obtained within one text comparison operation), it was customary to consider all downloaded decision trees private, i.e. not available to unregistered users. The user can add other people's private decision trees to the list of favorites or remove from it. Also, a user with explorer privileges can make decision trees public or remove publicity. These operations will allow you to limit the list of available trees in the pairwise comparison function to your own, selected or public graphs.

The model was responsible for interacting with the database. Decision tree graph data is organized into five tables: graph metadata, graph-related texts, graph vertices, graph edges, and a list of favorite graphs. One of the labor-intensive tasks was the implementation of parsing a text file with a decision tree graph and presenting the data in the database. The complexity of the task was due to the specific file format, the large amount of additional ignored data, and the wide variety of views. For example, the top of the graph should contain a list of parts of speech in the name and weight, but the format allowed for no restrictions on the name and no weight.

For graphical representation of a graph, the graphviz utility and the GraPHP library (https://github.com/graphp/graphviz) are used. As vertices, n-grams with combinations of parts of speech are used (for example, "Verb Pronoun"). If a graph vertex has a weight, then it is additionally displayed in the vertex label. If the weight of a vertex is greater than zero, then it is highlighted in red. By default, the image with the graph is displayed in a reduced version, but it can be opened in an additional window for a more detailed study.

Pairwise comparison of graphs is implemented based on the algorithm from [14]. The algorithm employs a dynamic programming approach and runs in polynomial time. The user can choose one of two comparison methods: based on vertex weights or based on vertex names and weights. To adjust the algorithms, the user can set the following parameters: Cost of adding a vertex, Cost of removing a vertex, Cost of changing a close vertex by weight, Level of proximity of vertices by weight, Cost of changing a far vertex by weight, Cost of changing a vertex with a partial match of parts of speech, Cost of changing a vertex with mismatch of parts of speech. For example, replacing a node (Adverb Verb, weight 0.405) with a node (Union Numeral, weight 0.012) with default settings (proximity level = 0.25, cost of replacing the far node by weight and with mismatch of parts of speech = 2, other costs = 1) will be equal to 4 (2 for the replacement of far vertices by weight + 2 for the replacement with mismatched parts of speech).

As a result of comparing graphs, the user gets the distance between the decision trees. The user can also view the final weight table.

4. Analysis of the regularities between the parameters of F. M. Dostoevsky's text sampling and decision trees

Let's study the regularities between the parameters of constructing a sample of texts (the size of the fragment, as well as the step of the sliding window) and the distances between the decision trees. The complete list of reference texts by F. M. Dostoevsky was taken as a sample [15].

Each article was divided into parts with a length of either x_1=1000 words or x_2=750 words. The limited number of such fragments forces us to resort to data expansion techniques. A sliding window was used as a similar technique: W={100, 200, 300, 400, 500, 750}. Then decision trees were built. Pairwise comparison of graphs was performed using a metric based on editing operations, the calculation of which was implemented based on the algorithm from [14]. Note that graph edit distance is currently one of the most popular similarity measures on a set of graphs [16].

Let's define the following weights of editing operations for G_1 and G_2:

– The cost of adding a vertex $u \in V_2 - V_2'$ to the graph G_2, which is denoted by $c_{ni}(u)$;
– The cost of removing a vertex $u \in V_1 - V_1'$ from the graph G_1, which is denoted by $c_{nd}(u)$;
– The operation of replacing two vertices $u \in V_1'$ with $f(u) \in V_2'$. Then the cost of replacement $c_{ns}(u)$ is the sum of the following components:
  - cost of changing close vertex by weight $c_{ns}^1(u)$ (at the same time, the level of proximity of vertices by weight is set $c_{ns}^2(u)$). That is, if the modulus of the difference between the Gini indices is less than $c_{ns}^2(u)$, then the vertex is considered "close", and the cost of the operation is equal to $c_{ns}^1(u)$;
  - cost of changing far vertex by weight $c_{ns}^3(u)$. That is, if the modulus of the difference between the Gini indices is greater than $c_{ns}^2(u)$, then the vertex is considered "far", and the cost of the operation is equal to $c_{ns}^3(u)$. Note that if the Gini indices are the same, then nothing is added to $c_{ns}(u)$;
  - cost of replacing a vertex with a partial match of parts of speech $c_{ns}^4(u)$, i.e. in bigrams of two pairs of parts of speech, one coincides;
  - the cost of replacing a vertex with a mismatch of parts of speech $c_{ns}^5(u)$, i.e. there are no common parts of speech in bigrams. Note that if the n-grams are the same, then nothing is added to $c_{ns}(u)$.

As an example, consider the calculation of the cost $c_{ns}(u)$ of a vertex replacement operation for three cases with the set parameters $c^1_{ns}(u) = 1$, $c^2_{ns}(u) = 0.25$, $c^3_{ns}(u) = 2$, $c^4_{ns}(u) = 1$, $c^5_{ns}(u) = 2$ (as a result, the total cost of the vertex replacement operation can be in the range from 0 to 4 inclusive):

– Let's compare the vertices with the bigrams "Adjective-Noun" (*gini*=0.348) and "Adjective-Noun" (*gini*=0.398), respectively. Since the n-grams are the same, and the difference in *gini* is 0.05, i.e. less than $c^2_{ns}(u) = 0.25$, then the replacement cost is $c_{ns}(u) = 0+1=1$.
– Let's compare the vertices with bigrams "Conjunction-Pronoun" (*gini*=0.262) and "Particle-Pronoun" (*gini*=0.358). Since n-grams coincide in one part of speech, and the difference in *gini* is 0.096, i.e. less than $c^2_{ns}(u) = 0.25$, then the replacement cost is $c_{ns}(u) = 1+1=2$.
– Let's compare the vertices with bigrams "Adjective-Participle" (*gini*=0.032) and "Modal word-Adverb" (*gini*=0.444). Since the n-grams do not completely coincide, and the difference in *gini* is 0.412, i.e. more than $c^2_{ns}(u) = 0.25$, then the replacement cost is $c_{ns}(u) = 2+2=4$.

Table 1 shows an example of a distance matrix obtained with the set parameters $c_{ni}(u) = 1$, $c_{nd}(u) = 1$, $c^1_{ns}(u) = 1, c^2_{ns}(u) = 0.25, c^3_{ns}(u) = 2, c^4_{ns}(u) = 1, c^5_{ns}(u) = 2$, where the depth of the tree was not limited. The fragment size was chosen to be 750 or 1000, and the size of the sliding window was 100, 200, 300, 400, 500, and 750 words. The result of cluster analysis is shown in fig. 4.

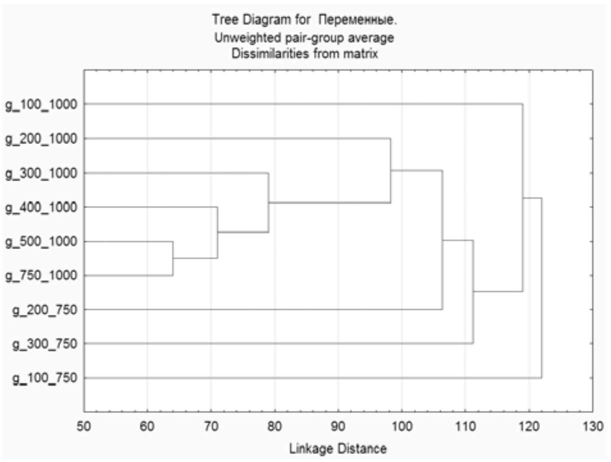


Fig. 4. An example of a dendrogram for a distance matrix from table 1

To represent a set of close graphs as one that would contain basic information about all structures,

Table 1.
An example of a distance matrix between decision trees.

F		1000 words						750 words		
	W	100	200	300	400	500	750	100	200	300
1000 words	100	0	129	118	116	109	102	138	128	131
	200	129	0	107	99	92	95	129	114	119
	300	118	107	0	85	78	74	119	112	112
	400	116	99	85	0	72	70	113	106	106
	500	109	92	78	72	0	64	109	100	103
	750	102	95	74	70	64	0	108	100	103
750 words	100	138	129	119	113	109	108	0	129	131
	200	128	114	112	106	100	100	129	0	124
	300	131	119	112	106	103	103	131	124	0

Table 2.
Median graphs for nine experiments

Experiment	$c_{ni}(u)$	$c_{nd}(u)$	$c^1_{ns}(u)$	$c^2_{ns}(u)$	$c^3_{ns}(u)$	$c^4_{ns}(u)$	$c^5_{ns}(u)$	$\min\limits_{G\in Z} D$	$\hat{G}$
the depth of the decision tree was not limited									
1	1	1	1	0,25	2	1	2	716	g_750_1000
2	2	2	1	0,25	2	1	2	1106	g_500_1000
3	1	1	1	0,25	2	2	4	716	g_750_1000
decision tree depth limited to 5									
4	1	1	1	0,25	2	1	2	565	g_750_1000
5	2	2	1	0,25	2	1	2	849	g_500_1000
6	1	1	1	0,25	2	2	4	568	g_750_1000
decision tree depth limited to 4									
7	1	1	1	0,25	2	1	2	357	g_500_1000
8	2	2	1	0,25	2	1	2	459	g_500_1000
9	1	1	1	0,25	2	2	4	357	g_500_1000

consider the concept of a median graph [17]. Let the distance $d(G_i, G_j)$ be given on the set $Z=\{G_1, G_2, \ldots, G_m\}$. Then we call the median graph on the set Z:

$$\hat{G} = \arg\min_{G\in Z} D = \arg\min_{G\in Z} \sum_{i=1}^{m} d(G, G_i),$$

where $D = \sum_{i=1}^{m} d(G,G_i)$ – the sum of the distances from the selected graph G to all other graphs from the set Z. Note that the median graph search problem is NP-complete. Table 2 describes the parameters for conducting nine experiments to calculate the median graph (experiments with other parameters showed comparable results). As the calculations showed, *g*_750_1000 became such a graph four times (i.e. x_1=1000, y_6=750), in the remaining five cases – *g*_500_1000 (i.e. x_1=1000, y_5=500).

Conclusion

In this article on the material of the pre-revolutionary magazines "Time" (1861–1863), "Epoch" (1864–1865) and the weekly "Citizen" (1873–1874), the problem of the stability of classifiers, which are built to determine the authorship of texts, is considered. The features were the frequency of occurrence of certain n-grams (encoded sequences of parts of speech). The decision trees obtained as a result of applying the sliding window method were compared with each other using the tree edit distance (it is currently one of the most popular similarity measures on a set of graphs). To analyze the results obtained in the SMALT information system (http://smalt.karelia.ru/), tools for storage, processing and comparing trees were implemented. For graphical representation of graphs the graphviz utility and the GraPHP library were used. The most stable decision trees (median graphs) were identified for several text collections using different weights for editing operations.

References

1. *Abramov R. V., Kulakov K. A., Lebedev A. A., Moskin N. D., Rogov A. A.* 2021. Research of features of Dostoevsky's publicistic style by using n-grams based on the materials of the "Time" and "Epoch" magazines. Vestnik of Saint Petersburg University. Applied Mathematics. Computer Science. Control Processes. Saint Petersburg, 17(4): 389–396.
2. *Safavian S. R., Landgrebe D.* 1991. A survey of decision tree classifier methodology. IEEE transactions on systems, man, and cybernetics, 21(3): 660–674.
3. *Pedregosa F., et al.* 2011. Scikit-learn: Machine learning in python. The Journal of Machine Learning Research, 12: 2825–2830.
4. *Lewis R. J.* 2000. An introduction to classification and regression tree (CART) analysis. Annual meeting of the society for academic emergency medicine in San Francisco, California. Citeseer 14.
5. *Coppersmith D., Hong S. J., Hosking J. R. M.* 1999. Partitioning nominal attributes in decision trees. Data Mining and Knowledge Discovery, 3(2): 197–217.
6. *Conte D., Foggia P., Sansone C., Vento M.* 2004. Thirty years of graph matching in pattern recognition. International Journal of Pattern Recognition and Artificial Intelligence, 18(3): 265–298.
7. *Jiang X., Bunke H.* 2008. Graph matching. Case-Based Reasoning on Images and Signals. Vol. 73 of Studies in Computational Intelligence. Springer, 149–173.
8. *Riesen K.* 2015. Structural Pattern Recognition with Graph Edit Distance: Approximation Algorithms and Applications. Advances in Computer Vision and Pattern Recognition. Springer, Heidelberg. 158 p.
9. *Bille P.* 2003. Tree edit distance, alignment distance and inclusion. IT University of Copenhagen. Technical Report Series, TR-2003-23.
10. *Jiang T., Wang L., Zhang K.* 1995. Alignment of trees – an alternative to tree edit. Theoretical Computer Science, 143(1): 137–148.
11. *Torsello A., Hidovic D., Pelillo M.* 2004. Four metrics for efficiently comparing attributed trees. Proc. ICPR'04 – 17th International Conference on Pattern Recognition, 2: 467–470.
12. *Torsello A., Hidovic D., Pelillo M.* 2005. Polynomial time metrics for attributed trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, 27(7): 1087–1099.
13. *Kuznetsov A. V.* 2017. Mera neskhodstva na mnozhestve grafov i ee prilozheniya [A measure of dissimilarity on a set of graphs and its applications]. Vestnik Voronezhskogo gosudarstvennogo universiteta. Seriya: sistemnyj analiz i informacionnye tekhnologii [Bulletin of the Voronezh State University. Series: system analysis and information technology], 1: 125–131.
14. *Isert C.* 1999. The editing distance between trees. Ferienakademie Bäume: Algorithmik und Kombinatorik. Sarntal, Italy.
15. *Rogov A. A., Abramov R. V., Buchneva D. D., Zakharova O. V., Kulakov K. A., Lebedev A. A., Moskin N. D., Otlivanchik A. V., Savinov E. D., Sidorov Y. V.* 2021. Problema atribucii v zhurnalah "Vremya", "Epoha" i ezhenedel'nike "Grazhdanin" [The problem of attribution in the maga-

zines "Time", "Epoch" and the weekly "Citizen"]. Petrozavodsk: Islands, 391 p.

16. *Wu L., Chen Y., Shen K., Guo X., Gao H., Li S., Pei J., Long B.* 2021. Graph Neural Networks for Natural Language Processing: A Survey. ArXiv abs/2106.06090.

17. *Hlaoui A., Wang S.* 2003. A New Median Graph Algorithm. Graph Based Representations in Pattern Recognition (GbRPR 2003). Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, 2726: 225–234.

Moskin N.D. PhD in Technics, Associate Professor, Petrozavodsk State University, 33 Lenin str., Petrozavodsk, 185910, Russia. E-mail: moskin@petrsu.ru (correspondent author)

Kulakov K.A. PhD in Physics and Mathematics, Associate Professor, Petrozavodsk State University, 33 Lenin str., Petrozavodsk, 185910, Russia. E-mail: kulakov@cs.karelia.ru

Rogov A.A. Dr. Sci. in Technics, Professor, Petrozavodsk State University, 33 Lenin str., Petrozavodsk, 185910, Russia. E-mail: rogov@petrsu.ru

Abramov R.V. ITMO University, 49 Kronverksky Pr., bldg. A, Saint Petersburg, 197101, Russia. E-mail: monset008@gmail.com