Methods of Acceleration of Term Correlation Matrix Calculation in the Island Text Clustering Method

Yakiv Yusyn[0000-0001-6971-3808], Tetiana Zabolotnia [0000-0001-8570-7571]

Faculty of Applied Mathematics, NTUU "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Peremohy ave., 37, Ukraine
[1]yusin.yakiv@gmail.com
[2]tetiana.zabolotnia@gmail.com

Abstract. This paper considers the task of accelerating the text clustering by the island clustering method. Based on the consideration of the basic implementation of this method, the step of calculating the term correlation matrix is highlighted for further acceleration. This step has quadratic complexity from the number of terms since it considers each combination of them in pairs. In the framework of this study, it is proposed to use parallel computing and memoization as methods to accelerate the calculation of the term correlation matrix. The experiment was carried out using a specially developed deterministic algorithm for generating test data, which is also described. Based on the results of the data obtained analysis, it is shown that the use of the proposed methods accelerates the calculation of the matrix by 8-10 times on a virtual machine with 4 physical cores and 8 logical ones.

Keywords: text clustering, memoization, parallelism.

1 Introduction

The clustering of natural language textual data is widely used both as one of the variants of automatic systematization and as a proper analysis tool [1]. Clustering of a natural-language text corpus allows to: understand its structure; highlight the most atypical texts in this specific corpus that will not belong to any cluster; reduce the size of the text corpus before its further processing or storing, discarding the most similar texts, and more [2].

Thus, due to the constant increase in the volume of text corpora and complication of their structure, formulation of new and improvement of existing methods of automatic (unsupervised) clustering of text documents becomes an urgent task [3].

One of the existing effective methods of text clustering, which provides clarity to the process of obtaining clusters for humans, is the method of island clustering [4].

The basis of this method is the implementation of a two-step procedure: the first stage provides the clustering of terms (hereinafter, “term” is used in general information retrieval meaning – any word/expression) that documents consist of; the second stage – construction of document clusters, based on the clusters of terms received in the first stage. Below is a more detailed list of steps of this method [4]:

Stage I. Clustering of terms:

1. pre-processing of texts from the input collection of documents: removal of stop words, lemmatization, etc.;
2. selection from the texts the set of terms which they consist of;
3. if necessary, filtering the resulting set of terms (for example, in situations where the initial centroids of the clusters are known or the resulting set is too large);
4. construction of a graph of terms correlation;
5. pre-processing of the graph and obtaining its approximation (optional);
6. clustering of the obtained approximation of the graph;

Stage II. Construction of document clusters:

7. splitting documents into clusters based on received term clusters.

One of the biggest cost steps is to build a graph of terms correlation. This procedure lays in calculation of the graph distances matrix (for this method of clustering the measure of terms correlation serves as a distance, so the names «distance matrix» and «correlation matrix» are identical for this research), which requires a pairwise consideration of all terms and leads to the quadratic complexity of the calculation process. Theoretically, the original clustering procedure for the obtained approximation of the graph is also quadratic (since it considers all edges of the obtained graph), but when applying the effective procedures of this very approximation, the complexity of this step is almost linear in practice.

Regardless of the mentioned above, as will be shown in Section 2 of this paper, none of the papers dealing with the island clustering method considers the acceleration of the term correlation matrix calculation. From this, we can conclude that the development of methods to accelerate the calculation of the term correlation matrix is a pressing issue, the successful solution of which will increase the speed of implementation of island clustering of texts as a whole.

Thus, the main objective of the research is to accelerate the calculation of the term correlation matrix in the method of island clustering of texts. The object of the study is the process of automatic clustering of natural language text data. The subject of the study is the methods of calculation acceleration and applying them in the context of the island text clustering method. According to the stated objective, the following tasks were set and solved:

1. research of existing methods to accelerate the calculations;
2. reasonable choice of methods to accelerate the calculations for their use in the method of island clustering of texts when calculating the term correlation matrix;
3. software implementation of the calculation of the term correlation matrix using the selected methods of accelerating the calculations;
4. analysis of the efficiency of the proposed means by the criterion of the speed of calculation.

2 Related Works

The island clustering method was first described in [4]. One objective of this study was to develop a method, the computational complexity of which has no more than a log-linear dependence on the number of texts. The island clustering method meets this requirement since the calculation time of the correlation matrix linearly depends on the number of documents. In the section devoted to the experiment, it was described that clustering took about 9 minutes on a pre-indexed subset (consisting of 17545 documents, longer than 100 characters) of the standard Reuters-21578 collection.

[5] is devoted to the improvement of the island clustering method, describing the method of essential clustering. The new method did not set any new requirements for clustering performance, leaving the requirement only for the dependence of complexity on the number of texts. Since the main objective of this work was to increase accuracy, there is no information about clustering performance in the work.

However, in practice, more important is the dependence of performance on the volume of the text corpus, which is more natural to consider as the number of terms. Described related papers did not describe any techniques for improving this calculation.

3 An Existing Algorithm for Calculating the Term Correlation Matrix in the Method of Island Clustering of Texts

The term correlation matrix is a symmetric matrix containing, as elements, the correlation value between the corresponding pairs of terms $(i, j), i \neq j$ [5]. Let N_{terms} be the number of unique terms (the dimension of the matrix), then to fill in the correlation matrix, it is necessary to calculate the correlation value between $N_{terms}(N_{terms} - 1)$ pairs of terms. The correlation value between the terms i and j is determined as follows. Let n be the total number of terms in all documents, n_i – be the number of terms in the documents that meet the term i, N_j – the total number of term j occurrences in all documents, and N_{ij} – the number of term j occurrences in documents containing the term i. Then the probability p_{ij} that in documents containing term i, N_{ij} is found or more of terms j, can be used as a basis for calculating the measure of the correlation of terms i and j. This probability can be calculated by the formula of the binomial distribution (1) [5]. The lower the probability obtained, the more terms are correlated with each other.

$$p_{ij} = P_B\left(N_{ij}, N_j, \frac{n_i}{n}\right) = \left(\frac{n_i}{n}\right)^{N_{ij}} \left(1 - \frac{n_i}{n}\right)^{N_j - N_{ij}} \binom{N_j}{N_{ij}} \qquad (1)$$

Since the probability of p_{ij} is not symmetric, in practice $b_{ij} = \max(p_{ij}, p_{ji})$ is used as a measure of the correlation of terms.

As a result, we can build matrix C based on the obtained values and this matrix will look like $C = \begin{matrix} \ddots & b_{1,2} & \cdots & b_{1,Nterms} \\ & \ddots & b_{2,3} & \vdots \\ & & \ddots & b_{Nterms-1,Nterms} \\ & & & \ddots \end{matrix}$. Such a matrix can be effectively stored in memory in the form of a vector with length $N_{terms}(N_{terms} - 1)/2$.

Thus, the basic algorithm for calculating the term correlation matrix C consists of the following steps [5]:

1. in the cycle with counter i from 1 to N_{terms} with step 1 perform pp.2-4;
2. in the cycle with counter j from $i + 1$ to N_{terms} with step 1 perform pp.3-4;
3. find p_{ij}, p_{ji} and calculate b_{ij};
4. assign $c_{ij} = b_{ij}$ and $c_{ji} = b_{ij}$.

This algorithm is a test of the statistical hypothesis of pairwise independence of the presence of terms in documents. The consequence of this is the fact that for randomly generated texts this algorithm will not find any significant relationship between the terms, and therefore, no clusters will be received at the output.

In practice, matrix C also can be additionally filtered to remove insignificant relations between terms (e.g. it can be done by use absolute threshold for term correlation [5]) – p.5 in the island clustering algorithm. Such a procedure will increase the quality of obtained term clusters.

4 Methods of Accelerating the Calculations That Can Be Used in Calculating the Term Correlation Matrix

4.1 Parallel Implementation of Cycles of the Correlation Matrix Calculation

In the algorithm of the term correlation matrix calculation, it is obvious that each of the $N_{terms}(N_{terms} - 1)/2$ iterations is independent of the others. Thus, instead of sequentially executing iterations of cycles, their parallel execution is possible.

Let T_{34} be the time of steps 3-4 of the algorithm (calculating the probabilities, finding the maximum and assigning it to the elements of the matrix). Then T_{serial} – the duration of the sequential implementation of the algorithm will be $T_{34}N_{terms}(N_{terms} - 1)/2$.

Let all the iterations be divided into P parts of approximately equal size, which will be processed by P independent threads in parallel. Then $T_{parallel}$ – the minimum duration of running a parallel algorithm implementation, which is $T_{34}N_{terms}(N_{terms} - 1)/2P$ can theoretically be achieved. [6]

Thus, comparing T_{serial} and $T_{parallel}$ we can conclude that it is theoretically possible to achieve an acceleration of the matrix calculation by P times (with a theoretical maximum in $T_{parallel} = T_{34}$ at $P = N_{terms}(N_{terms} - 1)/2$). However, the use of parallel implementation of the algorithm in practice requires additional data sharing and flow management costs, so that the practical acceleration will be less than P.

4.2 Memoization of Interim Results

Memoization is one of the methods of optimizing calculations, which consists of storing the results of a function execution in a lookup table to prevent repeated calculations [7]. When using this technique, before performing the memoized function, it is checked whether this function has already been performed for the passed parameter set. If the lookup table already saves the result of this function execution for a given set of parameters – it is used without performing calculations; otherwise, the calculation is performed, and the given result is recorded in the table [7]. In this way, memoization differs from the pure use of lookup tables in a way that when used, the lookup table is not pre-calculated but filled out if relevant.

In practice, memoization can only be applied to pure features that do not create side effects (such as file entries, database reads, etc.). Also, memoization cannot be applied if the calculation results have only a certain validity period, after which they must be re-calculated (in this case, more sophisticated techniques such as caching should be used).

The additional cost of memoization is only the need for additional memory to store the lookup table (its size depends on the set of valid values of the memoized function parameters).

In the case of calculating the term correlation matrix, memoization can be applied at the third point of the algorithm when calculating the parameters n_i and N_j from formula (1) of the calculation of p_{ij}. This will only require $2N_{terms}$ of additional memory, and if you use hash tables as lookup tables for memoization, the difficulty of getting the calculated result will be only $O(1)$.

5 Experiment and Results

5.1 Test Data Generation Algorithm

To test the proposed methods of accelerating the calculation of the term correlation matrix, in the framework of this research an own algorithm for generating a textual data corpus was developed.

The developed algorithm has the following properties:

1. accepts only N_{terms} from the input parameters – the number of unique terms in the corpus texts;
2. is fully deterministic – the same result will be obtained for startups with the same input parameter value;
3. the generated corpus of texts contains pairs of terms with different degrees of correlation – fully correlated, partially, not at all correlated.

Due to the deterministic nature of this algorithm, it is possible to compare the obtained experimental data (the rate of calculation of the term correlation matrix) for the same matrix dimension with the use of different combinations of the proposed methods.

The developed algorithm for generating a textual data corpus (with a term correlation matrix of a given dimension) consists of the following steps:

1. based on the input N_{terms} value, calculate the number of texts in the corpus being generated, as well as the boundaries of the range of texts in which the next term i will occur. Formulas (2) and (3) are used to calculate these two parameters;

$$N_{docs} = \left\lfloor \sqrt[4]{N_{terms}} \right\rfloor \tag{2}$$

$$range_{docs} = \left\lfloor \frac{N_{docs}}{5} \right\rfloor + 1 \tag{3}$$

2. for each i from 0 to N_{terms}:
   (a) obtain the row value of the next term by converting i to hexadecimal;
   (b) calculate the indexes of the texts in which the next term will occur according to formula (4). In case if any index of the resulting set is negative or exceeds the index of the last text – this index is reduced to valid values;

$$i_{range} = \left((i \bmod N_{docs}) - range_{docs} \ldots (i \bmod N_{docs}) + range_{docs}\right) \tag{4}$$

   (c) calculate the number of occurrences of the term i to the texts found in p.b:
      (i) the total number of occurrences of the term i to the corpus is calculated by the formula (5);

$$N_i = N_{docs}(i \bmod N_{docs} + 1) \tag{5}$$

      (ii) the number of occurrences of term i to the central text of the range is $N_i/2$;
      (iii) the rest of the $N_i/2$ occurrences are distributed equally among the other texts of the range;
   (d) the term i according to the number of occurrences calculated is added to the texts in the i_{range} range.

5.2 Features of Implementation of the Software Used for Testing

The software based on C# and the .NET Core 2.2 platform has been developed for experimental studies of the proposed methods to accelerate the calculation of the term correlation matrix. The source code for the developed software is fully open and available at https://github.com/yakivyusin/IslandClusteringAcceleration.

The measurement of the performance of the calculation of the term correlation matrix at different dimensions of the matrix and with different combinations of acceleration methods used was performed using the BenchmarkDotNet library (v.0.11.5) [8].

The launch of the developed software was performed in Google Cloud Platform virtual machine [9]: machine type – n1-highcpu-8 [10], the operating system installed – Windows Server 2012 R2 Datacenter, .NET runtime – .NET Core 2.2.6 (CoreCLR 4.6.27817.03, CoreFX 4.6.27818.02), 64bit RyuJIT.

5.3 Results Obtained

Testing the speed of calculating the term correlation matrix was performed for 8 dimensions: 49, 169, 245, 371, 441, 569, 659, and 785 terms. According to the developed algorithm for generating a textual data corpus, this set of dimensions provided the generation of corpora with sizes from 2 to 5 texts.

For each dimension of the test set, the term correlation matrix was calculated in twelve different ways: with sequential execution of cycles, with parallel execution of an external cycle, with parallel execution of both cycles, with memoization of calculation n_i, with memoization of calculation N_j, with memoization of calculation of both parameters.

The test results are shown in Table 1, in which the following notations are used to denote a matrix calculation variant:

- S – sequential execution of cycles;
- P – parallel execution of the external cycle only;
- P2 – parallel execution of both cycles;
- - – memoization of parameter calculation is absent (at the second position in the variant code corresponds to the parameter n_i at the third position – parameter N_j);
- + – memoization of parameter calculation is present (at the second position in the variant code corresponds to the parameter n_i, at the third position – parameter N_j).

Table 1. The obtained performance of methods for accelerating the calculation of the term correlation matrix

Method	Mean	StdDev	Mean	StdDev
	$N_{terms} = 49$		$N_{terms} = 169$	
S / - / -	35.4752 ms	0.1121 ms	2.6196 s	0.0062 s
S / + / -	33.2845 ms	0.0395 ms	2.4361 s	0.0011 s
S / - / +	19.1021 ms	0.0168 ms	1.3123 s	0.0018 s
S / + / +	16.8484 ms	0.0250 ms	1.1841 s	0.0008 s
P / - / -	10.8383 ms	0.0694 ms	749.1520 ms	50.0884 ms
P / + / -	10.0251 ms	0.0845 ms	696.1854 ms	40.3249 ms
P / - / +	6.0347 ms	0.0404 ms	384.8331 ms	22.9376 ms
P / + / +	5.2268 ms	0.0482 ms	335.5853 ms	16.9692 ms
P2 / - / -	8.5507 ms	0.0475 ms	597.0329 ms	3.2351 ms
P2 / + / -	8.0345 ms	0.0485 ms	548.0303 ms	1.8181 ms
P2 / - / +	4.7860 ms	0.0139 ms	296.4303 ms	0.6176 ms
P2 / + / +	4.2407 ms	0.0169 ms	286.2303 ms	0.9918 ms
	$N_{terms} = 245$		$N_{terms} = 371$	
S / - / -	7.7801 s	0.0018 s	55.1376 s	0.0198 s
S / + / -	7.4509 s	0.0052 s	52.9286 s	0.0456 s
S / - / +	3.9939 s	0.0018 s	27.6869 s	0.0282 s
S / + / +	3.6144 s	0.0021 s	25.4090 s	0.0081 s
P / - / -	1.9809 s	0.0565 s	13.1661 s	0.3204 s
P / + / -	1.9109 s	0.0720 s	13.0529 s	0.2354 s

P / - / +	1.0398 s	0.0474 s	6.7133 s	0.0920 s
P / + / +	903.7262 ms	30.9186 ms	6.0140 s	0.1030 s
P2 / - / -	1.7839 s	0.0055 s	12.5339 s	0.0165 s
P2 / + / -	1.6661 s	0.0092 s	12.4280 s	0.0126 s
P2 / - / +	894.3405 ms	1.6272 ms	6.4234 s	0.0168 s
P2 / + / +	839.8861 ms	5.0445 ms	5.5927 s	0.0100 s
	$N_{terms} = 441$		$N_{terms} = 569$	
S / - / -	93.5463 s	0.0535 s	198.8434 s	0.0268 s
S / + / -	88.9259 s	0.0641 s	191.0914 s	0.1423 s
S / - / +	46.4341 s	0.0200 s	100.9218 s	0.0948 s
S / + / +	42.5489 s	0.0446 s	91.5685 s	0.0610 s
P / - / -	21.9313 s	0.4058 s	46.7494 s	0.5385 s
P / + / -	21.0615 s	0.4508 s	44.4518 s	0.6213 s
P / - / +	10.7926 s	0.1662 s	24.0653 s	0.2072 s
P / + / +	10.1699 s	0.1097 s	21.7441 s	0.2087 s
P2 / - / -	21.0443 s	0.0344 s	45.2012 s	0.0235 s
P2 / + / -	21.1862 s	0.0157 s	42.6193 s	0.1020 s
P2 / - / +	10.6506 s	0.0158 s	22.3946 s	0.2510 s
P2 / + / +	9.2976 s	0.0409 s	20.6989 s	0.1456 s
	$N_{terms} = 659$		$N_{terms} = 785$	
S / - / -	413.1504 s	1.2729 s	697.6169 s	0.5776 s
S / + / -	394.7832 s	1.1438 s	672.0708 s	0.7522 s
S / - / +	212.8414 s	0.0583 s	358.1073 s	9.3059 s
S / + / +	197.5828 s	0.8492 s	336.9688 s	0.1039 s
P / - / -	90.4252 s	0.4118 s	157.0082 s	1.5449 s
P / + / -	90.1054 s	0.9602 s	156.6425 s	1.1942 s
P / - / +	50.4148 s	0.3149 s	84.6741 s	0.3796 s
P / + / +	45.4625 s	0.3596 s	78.0188 s	0.6076 s
P2 / - / -	91.8562 s	0.0361 s	161.2538 s	0.0429 s
P2 / + / -	87.9799 s	0.0304 s	159.0484 s	0.0618 s
P2 / - / +	48.3707 s	0.0264 s	79.9656 s	0.0314 s
P2 / + / +	43.5551 s	0.0280 s	75.6100 s	0.0295 s

The present results have high accuracy with a low variance – the ratio of the standard deviation to mean is in the range from 0.01% to 6.7% (standard error is also within the range from 0.02% to 0.92%). All measurements (except one) where this ratio exceeds the mean value of the dataset are related to parallel implementations. This may be caused by the fact that parallel implementations are more sensitive to random changes in the workload of the operating system with multitasking, while sequential implementations just utilize one core and work on it. Also, the process of garbage collection might influence the variance of results without memoization, but on the experimental dataset this process is almost deterministic due to dataset volume.

The charts of the obtained results for matrices with dimensions of 659 and 785 terms are shown in Fig. 1 and Fig. 2 respectively. Interactive charts for all test data are created using the Highcharts library [11, 12] and are available at https://yakivyusin.github.io/IslandClusteringAcceleration/plots.html.

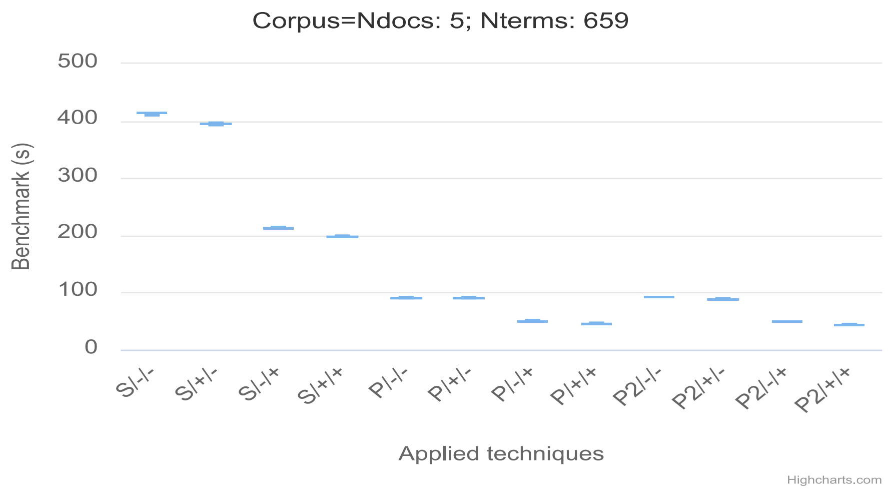

Fig. 1. The results of testing the performance of the calculation of the term correlation matrix with the dimension of 659. Achieved performance improvement is in the range between 1.05 and 9.49 times depends on used methods.

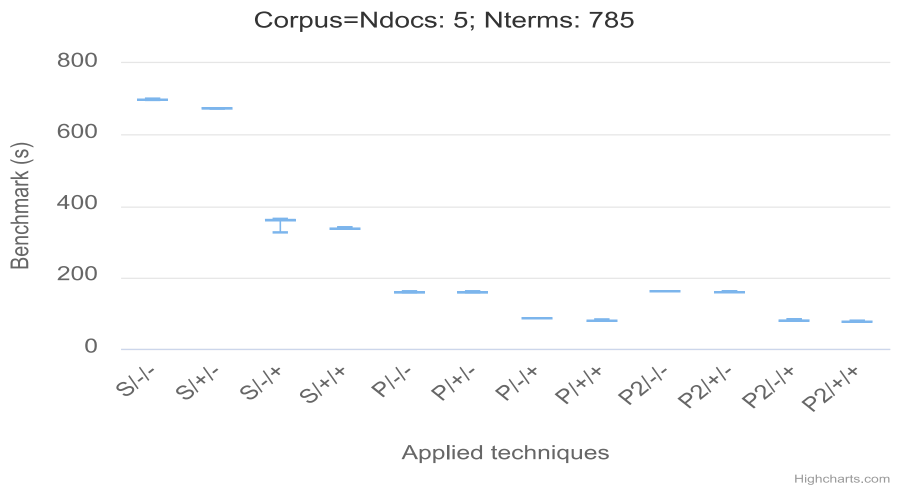

Fig. 2. The results of testing the performance of the calculation of the term correlation matrix with the dimension of 785. Achieved performance improvement is in the range between 1.04 and 9.22 times depends on used methods.

From the obtained results, it can be concluded that the use of all the proposed methods to accelerate the calculation of the term correlation matrix, increases the performance by 8-10 times in total, provided there are 8 virtual cores (depending on the dimension of the matrix and the number of texts; shown in Fig. 3), compared to the «naive» implementation of calculations.

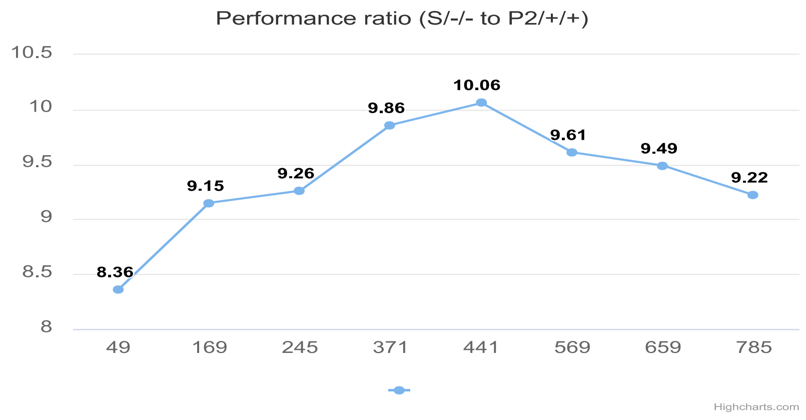

Fig. 3. Obtained results of accelerating the calculation of the term correlation matrix (the ratio of the time of «naive» calculation to the time of calculation using all the proposed methods)

As we can see, the main objective of the present paper is achieved –all proposed methods improve correlation matrix calculation performance in any given combination of used methods. The combination of all proposed methods (enabled parallelism and memoization of both calculation parameters) allows getting acceleration more than 8 times on machine with 4 physical cores / 8 virtual cores. Also, from the results obtained, the following dependencies can be emphasized:

1. clean use of parallelism leads to acceleration improvement more than 4 times. Such a result is because the test virtual machine has only 4 physical cores with 8 virtual cores;
2. memoization of N_j parameter has a much greater effect on acceleration than memoization of n_i parameter (average 2 times against average 1.05). We relate it to the fact that the calculation of N_j is a more complex process and requires read collection of all corpus terms;
3. variants of parallelism implementation with one cycle and two almost do not differ from each other (P2 variant on average is faster on 4-6%). The internal features of .NET Core parallelism implementation might influence this result and also that even in case of used one parallel cycle the number of chunks exceeds free cores count.

6 Conclusion

In the present paper, we have described methods of acceleration of term correlation matrix calculation in the island text clustering method. Proposed methods are based on two techniques: parallel execution of iterations and memoization of interim results (coefficients in correlation calculation formula). According to experimental results, the performance of calculation has been improved more than 8 times on 8 virtual core machine within the range of different text corpora (test data was generated by a described algorithm which was developed especially for this paper). The obtained result exceeds a theoretical limit of parallel execution (which is equal to the number of cores) due to memoization using.

Future work on this subject may be in the following areas:

1. experimental testing on different machines (with different number of cores) to better determine the dependence between improvement ratio and the degree of parallelism;
2. porting an implementation to a programming language without a garbage collector to reduce the number of factors that affect the results.

References

1. Reddy S., Kinnicutt P., Lee R.: Text Document Clustering: The Application of Cluster Analysis to Textual Document. In: Arabnia H., Deligiannidis L., Yang M. (eds.) INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE AND COMPUTATIONAL INTELLIGENCE, pp. 1174–1179. IEEE Computer Society, USA (2016).
2. Allahyari, M., Pouriyeh, S.A., Assefi, M., Safaei, S., Trippe, E.D., Gutierrez, J.B., Kochut, K.J.: A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques. ArXiv, abs/1707.02919 (2017).
3. Berry W.: Survey of text mining: clustering, classification, and retrieval. Springer, New York (2002).
4. Shmulevich M., Kiselev M., Pivovarov V.: The method of clustering texts, taking into account the joint occurrence of key terms, and its application to the analysis of the thematic structure of the news flow, as well as its dynamics. In: INTERNET MATHEMATICS, pp. 412–435. Yandex, Moscow (2005).
5. Shmulevich M.: The method of automatic clustering of texts, based on the extraction of object names from the texts and the subsequent construction of graphs for the joint occurrence of key terms. Ph.D. Thesis, MFTI, Moscow, Russia (2009).
6. Amdahl, G.M.: Validity of the single-processor approach to achieving large scale computing capabilities. In: AFIPS CONFERENCE PROCEEDINGS vol. 30 (Atlantic City, N.J., Apr. 18-20), pp. 483–485. AFIPS Press, Reston, Va. (1967).
7. Michie, D.: Memo Functions and Machine Learning. Nature 218, 19–22 (1968).
8. Overview | BenchmarkDotNet, https://benchmarkdotnet.org/articles/overview.html, last accessed 2019/10/27.
9. Cloud Computing Services | Google Cloud, https://cloud.google.com/, last accessed 2019/10/27.
10. Machine types | Compute Engine Documentation | Google Cloud, https://cloud.google.com/compute/docs/machine-types, last accessed 2019/10/27.
11. Interactive JavaScript charts for your webpage | Highcharts, https://www.highcharts.com/, last accessed 2019/10/27.
12. Box plot | Highcharts, https://www.highcharts.com/demo/box-plot, last accessed 2019/10/27.