УДК 004.414

HALAIKO D.O.,
OLIINYK Y.O.

SEARCHING TEXT SIMILARITY PARALLEL METHOD

АНОТАЦІЯ. Ця стаття на тему паралелізації алгоритму для пошуку схожості. В статті описано популярні послідовні алгоритми, методи паралелізації, тестові результати на великих масивах даних, діаграма діяльності паралельного алгоритму. Стаття містить переваги та недоліки ML, паралельного та послідовного алгоритму.

КЛЮЧОВІ СЛОВА: обробка природної мови, алгоритм схожості, паралельний, прискорення послідовного алгоритму.

ABSTRACT. This article is about parallelization of similarity search algorithm. In article is described popular sequential algorithms, ways to parallel, test results on big range of data, activity diagram of parallel algorithm. Article compares advantageous and disadvantageous of ML, parallel and sequential algorithms.

KEYWORDS: natural text processing, similarity algorithm, parallel, speed-up of sequential algorithm.

1. Introduction

Nowadays humankind uses a lot of gadgets, that give us access to information, but in this enormous snowball, the true author or even distinct information can't be distinguished. This leads us to great problems with supporting the original author of information and a lack of scientific breakthroughs. What have been described before isn't new. From ancient times it had been called "plagiarius" (means thief), now plagiarism.[1].

This problem is known as calculating the per cent of similarity between texts. can be part of a much bigger system. For example, by having an engine that distinguishes similarities between texts and having different sources of information, the author can be determined of the message. A real-life example can be searching for an "influencer" in social networks who can change public opinion.

The task of detecting borrowings can go beyond universities and be used by search engines, as well as systems for detecting propaganda, similar information, or retrieving the chain of distribution of information.

In this article, was implemented a single-thread algorithm and achieve speedup using parallelism.

2. Benefits of classical algorithmic solving over neural network

In our case, was used the classical algorithm approach, but this is not only one opportunity. Hikaru T.C. has created RNN to identify plagiarism [2]. The main disadvantages are "bias-variance tradeoff" and "overfitting" [3]. Other disadvantages lie in the great demand for device hardware, you need to have a great amount of CPU/GPU and RAM. Last, but not least is accuracy, classic algorithm can show better results compared to RNN.

3. Analyzing existing solution

In our article, was used a hybrid algorithm, which is a modification of several popular algorithms on the Internet.

Big variety of publications and books provide similar algorithms. One author in an article [4] provides the algorithm shown in figure 1. Despite some optimization that was done in our implementation, the core of the algorithm is fully operational and is close to production.

One more great example of plagiarism search is provided in another article [3] presented in figure 2. In this algorithm, the author suggests doing extra steps: lemmatize words and remove stop-words. However, the author of the second variant doesn't use cosin-similarity.

Algorithm purposed in article will be slightly different from the second. First, will be compared files not only with a local database of files but also with the Internet. One more step will be finding a symbol replacement. Symbol replacement is an intentional change of English letters to similar in Russian or Greek letters to confuse the checking system and hide plagiarism (similarity). The final version is shown in figure 3.

In the first step, will be converted the text to lowercase. Then will be replaced symbols that don't form a word (digits, mathematical, signs) with space. Next, text is split on words using space, tab and new lines symbols as delimiters. Next, "symbol replacement" is replaced with the correct symbols. Then, text is lemmatized to replace "doing" with "do". After that, stop words are removed. A stop word is a word that can be omitted, and the sentence wouldn't lose any sense. For example: "to", "a", "the", "an", etc. Retrieved words are compared with files and the Internet and the result is writen to a file.

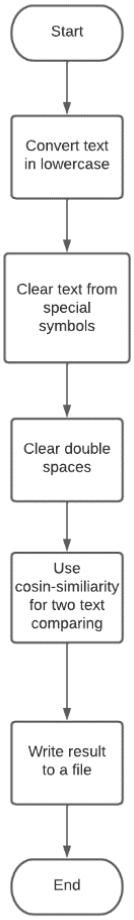

Figure 1 Algorithm of plagiarism search described in the article[4]

1. Development of sequential algorithm and analyzing the performance

A sequential algorithm was developed using the schema in figure 3. To show the correctness of the written program, unit tests has been written, that cover the main programming use cases. The finding of the plagiarism part have been tested. Tests were conducted on text frames bigger than 100 words each. Also, manual tests were conducted, that show written software can be trusted. All tests (including performance) have been run on Intel Core i7-8569U (4 cores).

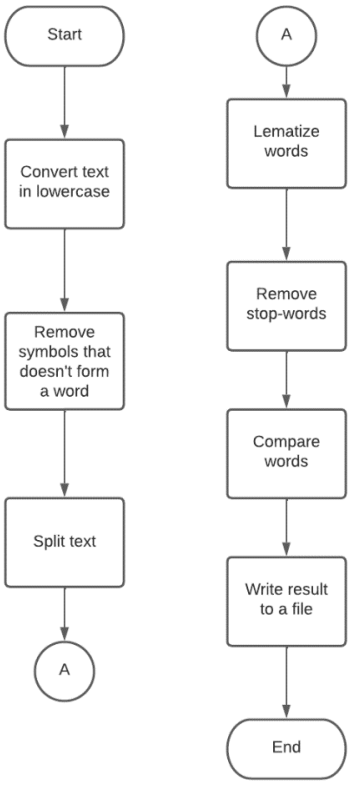

Figure 2 Algorithm of plagiarism search described in the article[3]

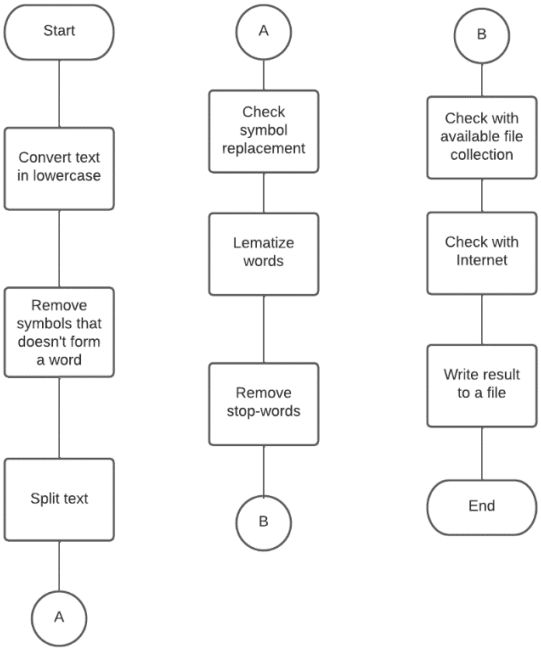

Figure 3 Purposed sequential algorithm for plagiarism detection

To measure the performance of programming software, performance testing have been split into 2 parts: check with local files and Internet check.

During the test, the text size of the first text is equal to 87 341 words. The results of the experiments see in table 1.

Table 1. Execution results of local files plagiarism detection

Words count	Time average(seconds)
3892	5,05
78445	8,58
337849	23,27
487757	30,51
500329	31,13
553557	33,63
555038	33,94
556049	34,07
646627	38,12
688707	40,46
768345	44,43
805327	46,22
823766	47,45
831904	47,89
835434	48,05
840835	48,56
850048	49,05
854314	49,22
958662	55,39
1013391	57,11
1014345	57,09
1026145	57,74
1196921	67,29
1249289	69,65
1256392	70,72
1289107	71,97
1294452	72,42
1305937	74,03
1369344	76,96

In each experiment, several iterations was conducted to reduce stochasticity. In figure 4, you can see the build chart dependency of time from words in check.

Check with the Internet is a bit trivial. From the text, the first, last and middle sentences have been chosen. From these 3 sentences, requests were created to the search engine. Each request produces 10 links to similar materials, so around 30 links for 3 sentences were gotten. Execution results are provided in table 2. The average sequential execution with the Internet is 59,97.

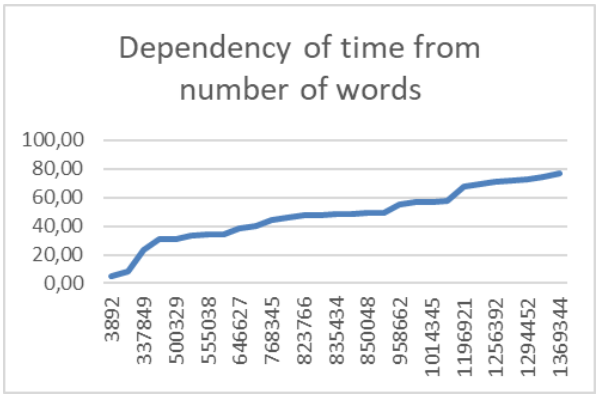

Figure 4 Chart dependency of time from words in check of plagiarism detection with local files

Make notice, that check with the Internet can be configured to choose more sentences for check with search engines, but It'll slightly increase the time of algorithm execution.

Table 2. Execution results in plagiarism detection with Internet

Iteration number	Execution time(seconds)
1	62,25
2	56,05
3	54,56
4	72,74
5	54,25

To sum up, performance depends on the input amount of data. From figure 4 is conducted, that time will increase linearly.

1. **Libraries and tools used to parallelize the algorithm**

In this work, the language Python3 was used. The main advantage of using Python is the great support of NLP (natural language processing) libraries.

It's worth mentioning, Python has been developed as a solution for writing scripts, but now it contains a lot of tools for multithreading and big data processing.

For this work, tools in standard Python3 were used: multiprocessing [5] and threading [6]. The "**Threading**" library contains methods for thread manipulation (create, use). "**Multiprocessing**" contains tools for "pool" creation and "map" operation on several threads.

2. A parallel similarity search method

Activity diagram of the parallel algorithm you can see in figure 5 and figure 6. Fork/Join block on schema display creation threads and merge them with the main thread. Using the algorithm provided on schema, was gained an increase of not less than 5%.

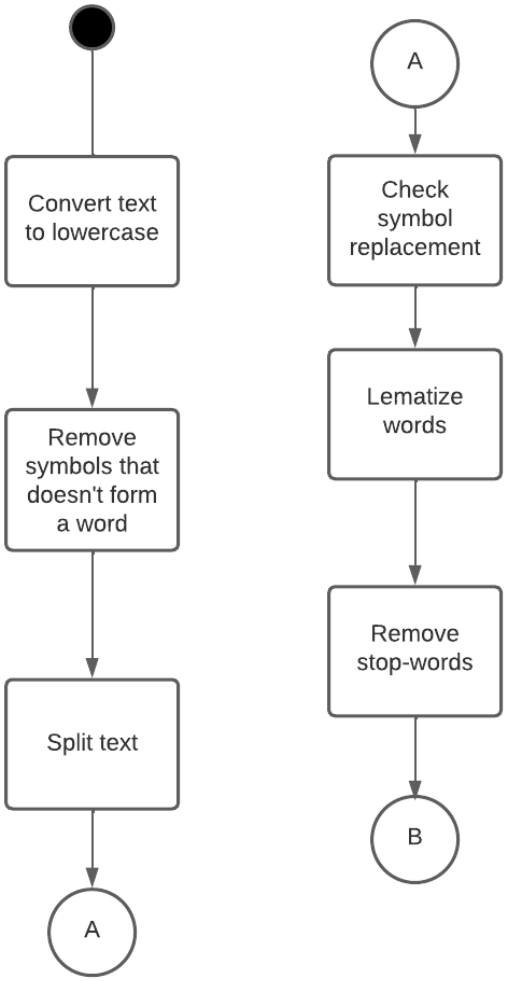

Figure 5 Parallel algorithm activity diagram

3. The result of parallel method calculations

After several tests (About testing methodology read in the chapter), the results were retrieved presented in table 3. For small word counts (less than 300k words) sequential algorithm is faster, but for big counts of words – the parallel algorithm is. Moreover, from 800k words parallel algorithm is faster by more than 2 times as shown in figure 7.

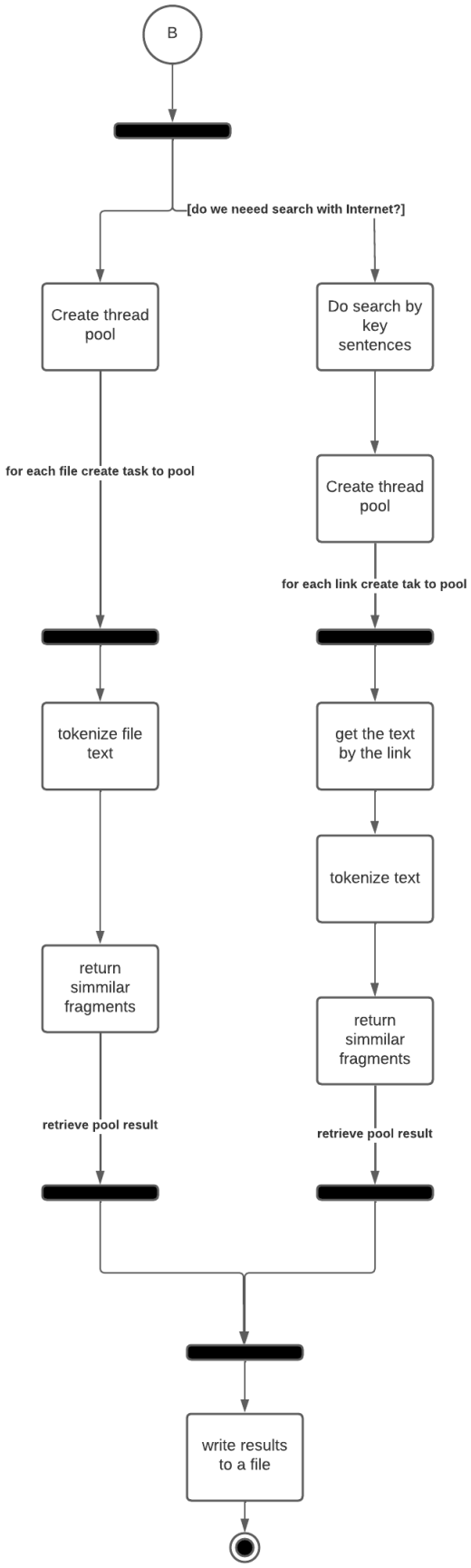

Figure 6 Parallel algorithm activity diagram (continue)

Table 3. Execution results of local and parallel files plagiarism detection

Words count	Sequential algorithm	Parallel algorithm

3892	5,05	7,07
78445	8,58	10,30
337849	23,27	21,64
487757	30,51	21,69
500329	31,13	21,81
553557	33,63	22,14
555038	33,94	22,21
556049	34,07	22,11
646627	38,12	22,52
688707	40,46	22,74
768345	44,43	23,43
805327	46,22	23,89
823766	47,45	24,10
831904	47,89	24,16
835434	48,05	24,40
840835	48,56	24,56
850048	49,05	24,60
854314	49,22	24,67
958662	55,39	25,25
1013391	57,11	25,82
1014345	57,09	26,05
1026145	57,74	26,10
1196921	67,29	27,01
1249289	69,65	28,06
1256392	70,72	27,99
1289107	71,97	28,37
1294452	72,42	28,37
1305937	74,03	28,96
1369344	76,96	29,43

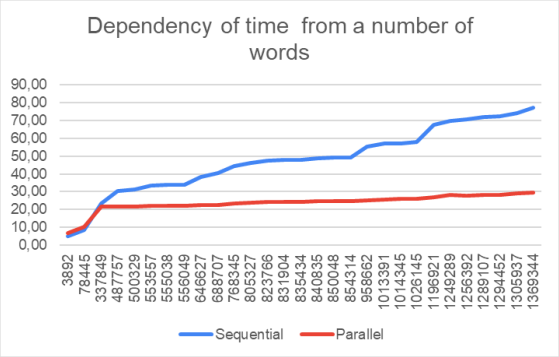

Figure 7 Dependence of execution time on the number of elements of the verification algorithm with local files

Let's calculate **speedup**:

$$Speedup = \frac{\sum t_{sequential\ algorithm\ time}}{\sum t_{parallel\ algorithm\ execution\ time}} = \\ = 2{,}016 \ = \ 201{,}62\%$$

From the chart in figure 8, **speed-up** is more than 1 if words count more than 300k words.

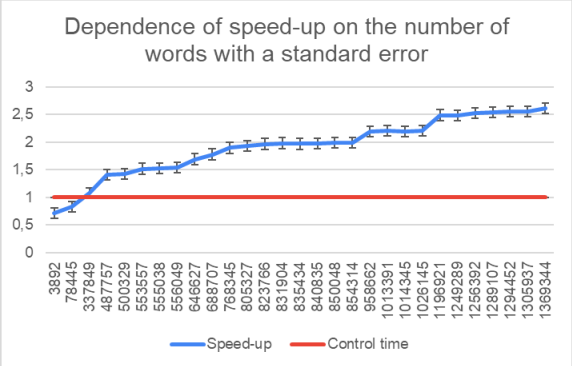

Figure 8 Dependence of speed-up on the number of words with a standard error

Moreover, speed up in check with the Internet is more than 3 times(table 4).

Table 4. Results of Comparing the Execution Time of Serial and Parallel Checking Algorithms with the Internet

	Sequential algorithm	Parallel algorithm	Speed-up
Average	59,97	17,66	3,39631966

Conclusion

A method for searching similarities is underestimated topic, that needs further discovery. Above the main sequential algorithms were discussed, that are used nowadays and purposed our own. The sequential algorithm has been parallelized and achieved at least 2 times speed-up. To reduce stochasticity in results, several series of experiments were conducted.

Software was tested on big-size data, more than 1,4 million words. An interesting fact is that for an amount less than 300k words sequential algorithm is better than a parallel. It happens because it takes time to create threads or pools.

All in all, the development of the parallel algorithm is much harder (debugging, testing, understanding), but the speed-up, that was gained, worse it.

List of literature

1. Bailey, Jonathan. 2011. "The World's First "Plagiarism" Case - Plagiarism Today." Plagiarismtoday. October 4, 2011. https://www.plagiarismtoday.com/2011/10/04/the-world%E2%80%99s-first-plagiarism-case/.
2. Hikaru, Thomas. 2021. "Simple Plagiarism Detection in Python | by Thomas Hikaru Clark | Towards Data Science." Towardsdatascience. January 25, 2021. https://towardsdatascience.com/simple-plagiarism-detection-in-python-2314ac3aee88.
3. Ahmad, Imran. 2020. 40 Algorithms Every Programmer Should Know : Hone Your Problem-Solving Skills by Learning Different Algorithms and Their Implementation in Python. Birmingham Etc.: Packt.
4. JAZYAH, YAHIA. 2018. "Microsoft Word - 006-0018(2018).doc." Iaras. October 20, 2018. https://www.iaras.org/iaras/filedownloads/ijc/2018/006-0018(2018).pdf.
5. Python Foundation. 2022. "multiprocessing — Process-based parallelism — Python 3.10.8 documentation." Docs. October 24, 2022. https://docs.python.org/3/library/multiprocessing.html.
6. Foundation, Python. 2022. "threading — Thread-based parallelism — Python 3.10.8 documentation." Docs. October 24, 2022. https://docs.python.org/3/library/threading.html.