DOI 10.31891/2307-5732-2024-333-2-69
УДК [004.021:004.91]+004.415.53

YUSYN YAKIV
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
https://orcid.org/0000-0001-6971-3808
e-mail: yusyn@pzks.fpm.kpi.ua
RYBACHOK NATALIIA
National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute"
https://orcid.org/0000-0002-8133-1148
e-mail: rybachok@pzks.fpm.kpi.ua

IMPROVEMENT OF THE DETERMINISTIC METHOD OF THE TEXT DATA CORPORA GENERATION

This paper is devoted to the issue of generating corpora of text data for their use in solving software engineering problems in the context of developing information systems for natural language processing (NLP). One of the methods intended for this is the basic CorDeGen method, however, the analysis revealed certain disadvantages. Such a disadvantage is that NLP methods at the pre-processing stage can remove part of the terms generated by this method from the texts, treating them as stop words of a certain language. Removing part of the terms leads to the fact that the distribution of terms between the texts predicted by the CorDeGen method is distorted and the obtained result of processing the corpus with a certain NLP method is significantly different from the expected one.

To solve this disadvantage, a new modified CorDeGen+ method is proposed in the paper, which introduces an additional, language-dependent stage of checking each generated term for admissibility, and if necessary, replacing it with another one. At the same time, all the advantages of the basic CorDeGen method are preserved by the proposed method, as well as other possible disadvantages, except for the corrected one. The paper examines language variations of the proposed method for the four most common European languages and a variation for languages that use non-Latin letters.

The conducted experimental test showed the effectiveness of the CorDeGen+ method in terms of correcting the described disadvantage of the basic CorDeGen method. Also, this test showed that the degree of slowdown of the corpora generation process due to the introduction of an additional stage depends on the corpus size. In the case of micro-corpora (100 unique terms), the degree of slowdown reaches 39%, but as the size of the corpus increases, the degree drops sharply, and for super-large corpora (312500 unique terms) it reaches a maximum of 6.8%.

Keywords: natural language processing; software quality assurance; text data corpora; corpora generation; CorDeGen method.

ЮСИН ЯКІВ, РИБАЧОК НАТАЛІЯ
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

ПОКРАЩЕННЯ ДЕТЕРМІНОВАНОГО МЕТОДУ ГЕНЕРУВАННЯ КОРПУСІВ ТЕКСТОВИХ ДАНИХ

Дана робота присвячена проблематиці генерування корпусів текстових даних для їх використання під час вирішення задач інженерії програмного забезпечення в контексті розроблення інформаційних систем оброблення природної мови. Одним із методів, призначених для цього, є базовий метод CorDeGen, проте проведений аналіз виявив його певні недоліки. Таким недоліком є те, що методи оброблення природної мови на етапі попереднього оброблення можуть видаляти із текстів частину генерованих даним методом термів, розцінюючи їх як стоп-слова певної мови. Видалення частини термів призводить до того, що передбачений методом CorDeGen розподіл термів між текстами спотворюється і отриманий результат оброблення корпусу певним методом оброблення природної мови значно відрізняється від очікуваного.

Для вирішення даного недоліку в роботі запропоновано новий модифікований метод CorDeGen+, що вводить додатковий, мовнозалежний етап перевірки кожного генерованого терма на допустимість, і у разі необхідності – заміни його на інший. При цьому, всі переваги базового методу CorDeGen зберігаються запропонованим методом, як і інші можливі недоліки, крім виправленого. В роботі розглянуто мовні варіації запропонованого методу для чотирьох найпоширеніших європейських мов та варіацію для мов, що використовують не латинські літери.

Проведена експериментальна перевірка показала ефективність методу CorDeGen+ у частині виправлення описаного недоліку базового методу CorDeGen. Також дана перевірка показала, що ступінь уповільнення процесу генерування корпусів через введення додаткового етапу залежить від розміру корпусу. У випадку мікро-корпусів (100 унікальних термів) ступінь уповільнення сягає 39%, проте зі збільшенням розміру корпусу стрімко падає і для надвеликих корпусів (312500 унікальних термів) складає максимум 6.8%.

Ключові слова: оброблення природної мови; забезпечення якості програмного забезпечення; корпус текстових даних; генерування корпусів; метод CorDeGen.

Introduction

Information systems dedicated to solving various tasks of natural language processing have become widespread today and are used in everyday life and many areas of human professional activity. Due to the peculiarities of the field of natural language processing, solving various software engineering tasks during the development of such information systems can be complicated. One of the peculiarities of this field that complicates the development process of information systems is that they often work with corpora (or collections) of text data.

When solving software engineering tasks for natural language processing information systems, there may be a need for a large number of different corpora of text data. An example of such a task would be quality assurance – using certain testing methodologies, such as property-based testing or metamorphic testing, requires hundreds or even thousands of corpora. In such a case, it is impossible to limit testing to a few, pre-prepared and calculated corpora, so

the question of generating corpora on demand arises. Today, the problem of corpora generation methods adapted for use specifically in solving software engineering tasks is poorly studied, although the need for such methods does not diminish over time. That is why the development of new and improvement of existing methods of generating corpora of text data for their use in solving software engineering tasks is and remains an urgent task.

Analysis of recent research

Currently, there are many studies devoted to the generation of text data corpora for use in various natural language processing tasks (e.g., [1–4]), but most of them have little applicability to solving software engineering tasks. This is due to various factors, the main ones of which are the following:

- The need for a large amount of natural data for corpus generation.
- Low performance.
- The absence of the possibility of a priori determination of the result of processing the generated corpus by various methods.

In paper [5], the authors proposed a method for generating corpora of text data called CorDeGen, which was developed specifically for its use in solving software engineering tasks. The steps of the abstract CorDeGen method are shown in Fig. 1 [5].

Method 1. Abstract CorDeGen method

1: Input parameter N_{terms} (number of unique terms)
2: Calculation of the number of documents N_{docs} using the function $f(x)$
3: For each term i
4: Receive the string representation of the term
5: Calculation of the vector $\overline{tf}$, containing the number of occurrences of the term in documents, using the calculation of the function $g(x)$
6: Record to each document of the string representation of the term, based on the calculated number of occurrences

Figure 1. The abstract CorDeGen method

The description of the abstract CorDeGen method does not specify a specific method of receiving a string representation of a term and specific functions $f(x)$ or $g(x)$, because if certain conditions are met, this has little effect on the behavior of the method and the obtained results.

Based on the definition of the abstract method, the basic CorDeGen method is presented in [5], which determines the necessary elements in the manner given in formulas (1) – (3):

$$term_{string}: itoa(i, 16), \tag{1}$$

$$f(x): \left\lfloor \sqrt[4]{x} \right\rfloor, \tag{2}$$

$$g(x): tf_{ij} = \begin{cases} 0, j \notin (c_i - r \dots c_i + r) \\ \frac{1}{2r+2} N_i, j \in (c_i - r \dots c_i + r), j \neq c_i \\ \frac{2}{2r+2} N_i, j = c_i \end{cases}, \text{where}$$

$$c_i = i \bmod N_{docs},$$

$$r = \left\lfloor \frac{N_{docs}}{5} \right\rfloor + 1,$$

$$N_i = N_{docs}(i \bmod N_{docs} + 1). \tag{3}$$

In such a variant, the algorithm implementing the basic CorDeGen method has asymptotic computational complexity $O(N^{1,5})$ [5].

The advantages of the basic CorDeGen method for solving software engineering tasks are [5]: full determinism of the generation process; the minimum possible number of input parameters (only one – the size of the corpus, expressed via the number of unique terms); providing the possibility of a priori determination of the structure of the generated corpus and the result of its processing.

However, the basic CorDeGen method also has disadvantages and/or limitations. One such disadvantage is related to the use of hexadecimal numbers as terms, because most natural language text processing methods (and therefore most of its software implementations) involve preprocessing the texts, such as removing stop words for a given natural language. Accordingly, if the software does not provide for turning off such pre-processing (or choosing a "neutral" language with an empty dictionary of stop words), then part of the terms from the generated texts can be deleted. For example, if the software implementation of the natural language processing method considers any texts as written in English, then the term "a" (index 10) will be deleted because it completely matches the article. Also, decimal numbers are often included in the lists of stop words, which makes this disadvantage relevant in the case of

methods for languages based on the non-Latin alphabet too.

This disadvantage, due to the removal of part of the terms, leads to a change in the distribution of terms between the generated texts and, accordingly, makes it difficult to determine the expected result of processing the generated corpus. This work is dedicated to correcting this disadvantage.

Formulation of the goals of the article

The aim of the paper is to eliminate the disadvantage of the basic CorDeGen method of generating text corpora by developing such a modified and improved method that the text corpora generated with its help will not contain terms similar to stop words.

Presentation of the main material

CorDeGen+

A new modification of the basic CorDeGen method, named CorDeGen+, is proposed to solve the above-described disadvantage.

The main idea of this modification is that after obtaining a string representation of a term (by converting it to the hexadecimal number system), it is necessary to check whether the received representation falls into the list of stop words of a given language. If it hits, a new string representation must be obtained and checked; this step must be repeated until the resulting string representation no longer falls into the list of stop words. Accordingly, the CorDeGen+ method introduces two additional functions: $r(x)$ – the function of obtaining a new representation of the term and $v(x)$ – the function of checking whether the term is included in the list of stop words.

Thus, the abstract CorDeGen+ method consists of the steps shown in Fig. 2.

Method 2. Abstract CorDeGen+ method

1: Input parameter N_{terms} (number of unique terms)
2: Calculation of the number of documents N_{docs} using the function $f(x)$
3: For each term i
4: Receive the string representation of the term
5: While the string representation of the term is prohibited ($v(x) == $ 'DENY')
6: Receive the new string representation of the term using the function $r(x)$
7: Calculation of the vector $\overline{tf}$, containing the number of occurrences of the term in documents, using the calculation of the function $g(x)$
8: Record to each document of the string representation of the term, based on the calculated number of occurrences

Figure 2. The abstract CorDeGen+ method

Since in most cases, the additional step 6 will be performed 0 or 1 times, the computational complexity of any algorithm implementing the CorDeGen+ method remains the same compared to the basic CorDeGen method – $O(N^{1,5})$.

Obviously, the function $v(x)$ depends on the particular natural language being considered, so the entire CorDeGen+ method is language-dependent. Next, variations of the proposed CorDeGen+ method for the most common European languages will be considered: English (CorDeGen+$^{(EN)}$), German (CorDeGen+$^{(DE)}$), French (CorDeGen+$^{(FR)}$) and Italian (CorDeGen+$^{(IT)}$). The development of separate variations of the method for a specific language allows not to enter an additional input parameter with a list of stop words and also allows optimization of the algorithm for determining whether a term belongs to exclusions. As for languages not based on the Latin alphabet, for any such language, it is possible to use the same variant of the CorDeGen+ method, which will filter out only decimal numbers as stop words. This variant of the method is called CorDeGen+$^{(0\text{-}9)}$.

Unlike the function $v(x)$, the function $r(x)$ does not depend on the specific natural language chosen. The simplest way to implement this function is to use the same hexadecimal representation (thus, no new algorithms or functions are introduced, which does not complicate the implementation of the method), but for the term index with a certain shift s. That is, if the hexadecimal representation of the term index falls into the list of stop words, the proposed function $r(x)$ will return the hexadecimal representation of the index $i+s$. If the hexadecimal representation of this term index also falls into the stop word list, then the index $i+2s$ representation will be returned, and so on. Given that, according to the original CorDeGen method, the indices are in the range from 0 to N_{terms} (not inclusive), the value of N_{terms} can be used as the shift s value. Thus, the shifted indexes will never overlap with the original indexes.

To implement the function $v(x)$ in the case of CorDeGen+$^{(EN)}$, the list of stop words for the English language, given by the link [6], is taken as a basis. This list contains 733 stop words, but most of them cannot be generated by the CorDeGen method because they contain letters outside the [a-f] range. There are only seven words

in total, which consist only of letters allowed in the hexadecimal numbering system: a, b, be, f, d, c, e. Also, as mentioned above, we will additionally consider all representations containing only decimal numbers – 0, 12, 439, etc. – to be prohibited (this is also true for other language variations). Considering everything described above, the function $v(x)$ of the CorDeGen+$^{(EN)}$ implementation can be presented as follows:

- If the length of the hexadecimal representation x is 1, then it is forbidden (it either contains a decimal number or falls into the original stop word list).
- If the hexadecimal representation x is "be", then it is forbidden.
- If the hexadecimal representation x contains only decimal digits, then it is forbidden.
- Otherwise, it is allowed.

To implement the function $v(x)$ in the case of CorDeGen+$^{(DE)}$, the list of stop words for the German language, given by the link [7], is taken as a basis. Of the 620 stop words in this list, eight can be generated by the CorDeGen method: a, b, c, d, e, f, ab, da. Also considering the exclusion of decimal numbers, the function $v(x)$ of the CorDeGen+$^{(DE)}$ implementation can be given as follows:

- If the length of the hexadecimal representation x is 1, then it is forbidden.
- If the hexadecimal representation x is "ab" or "da", then it is forbidden.
- If the hexadecimal representation x contains only decimal digits, then it is forbidden.
- Otherwise, it is allowed.

To implement the function $v(x)$ in the case of CorDeGen+$^{(FR)}$, the list of stop words for the French language, given by the link [8], is taken as a basis. This list of 691 stop words includes nine prohibited by the CorDeGen+ method: a, b, c, d, e, f, ce, da, de. Considering the exclusion of decimal numbers, the function $v(x)$ of the CorDeGen+$^{(FR)}$ implementation can be given as follows:

- If the length of the hexadecimal representation x is 1, then it is forbidden.
- If the length of the hexadecimal representation x is 2 and it is "ce", "da", or "de", then it is forbidden.
- If the hexadecimal representation x contains only decimal digits, then it is forbidden.
- Otherwise, it is allowed.

To implement the function $v(x)$ in the case of CorDeGen+$^{(IT)}$, the list of stop words for the Italian language, given by the link [9], is taken as a basis. The list (632 stop words in total) for this language contains the most words that can be generated by the CorDeGen method (are valid hexadecimal numbers): a, b, c, d, e, f, ad, da, ed, fa, ecc, ebbe, fece. Due to the rather large number of stop words, the function $v(x)$ can be represented in different ways (including decimal numbers), with different ways and degrees of optimization of the check. The simplest option is as follows:

- If the length of the hexadecimal representation x is 1, then it is forbidden.
- If the length of the hexadecimal representation x is 2 and it is "ad", "da", "ed" or "fa", then it is forbidden.
- If the hexadecimal representation x is "ecc", then it is forbidden.
- If the length of the hexadecimal representation x is 4 and it is "ebbe" or "fece", then it is forbidden.
- If the hexadecimal representation x contains only decimal digits, then it is forbidden.
- Otherwise, it is allowed.

In the case of CorDeGen+$^{(0\text{-}9)}$, the function $v(x)$ has the simplest representation (in fact, this is the last check of each language function $v(x)$ presented above):

- If the hexadecimal representation x contains only decimal digits, then it is forbidden.
- Otherwise, it is allowed.

The proposed modified CorDeGen+ method preserves all the main advantages of the basic CorDeGen method: the method remains fully deterministic; in the case of using individual variations of the method for individual languages, the number of input parameters does not increase; the distribution of terms between the texts remains unchanged. In addition, the disadvantage of removing part of the terms from the generated texts is solved, which was the purpose of developing this modified method.

As for the disadvantages of the proposed method, it retains all the other disadvantages of the basic method, for example, the lack of semantics in the generated terms and texts. Compared to the basic method, the CorDeGen+ method also has its own disadvantage, namely the need to perform additional iterations. Although this does not change the asymptotic computational complexity of the algorithm, in practice its software implementations will run slower than a similar implementation of the basic method (which will be shown later).

Considering the above-described advantages and disadvantages of both methods and their resulting limits of applicability, Fig. 3 shows a decision tree for choosing between the basic CorDeGen method and the proposed CorDeGen+ method.

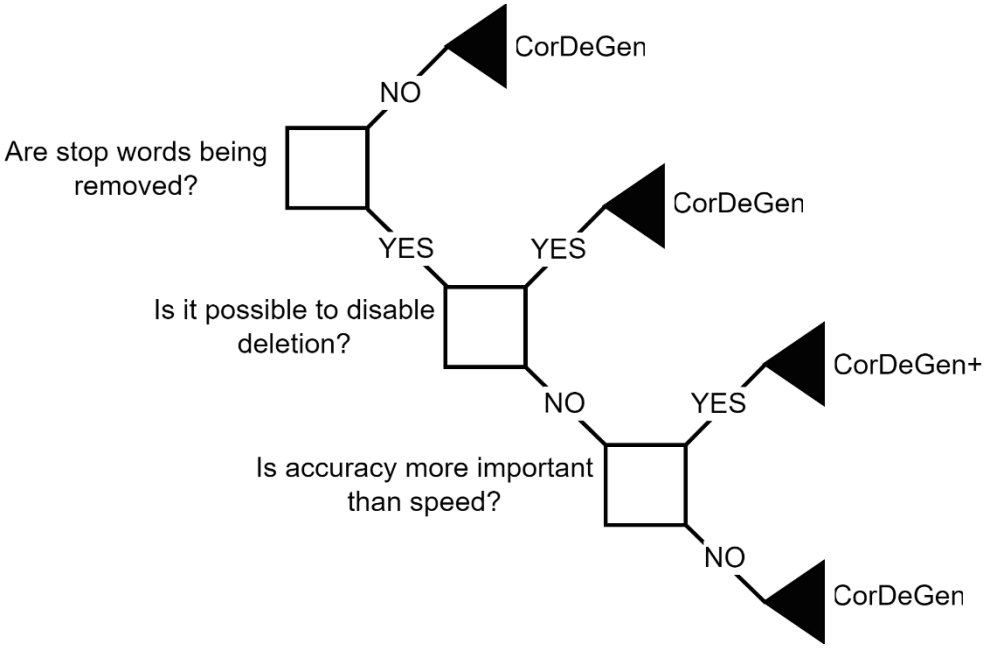

Figure 3. Decision tree for choosing between the basic and the proposed method

Reference implementation

The proposed language variations of the developed CorDeGen+ method and the basic CorDeGen method are implemented in the C# programming language on the .NET 8 platform (.NET SDK 8.0.200 / .NET Runtime 8.0.2).

The developed reference implementation consists of several software artifacts (modules) of different types, which contain different functionalities:

- A software library (CorDeGen) containing implementations of the basic method and proposed language variations of the modified CorDeGen+ method. This library can be connected to any existing or new project on the .NET platform (programming languages C# / F# / etc.) to provide corpora generation capabilities.
- Two software artifacts (CorDeGen.Tests.Unit and CorDeGen.Tests.Integration) containing unit and integration tests for the developed implementations. The developed tests are implemented based on the xUnit [10] and FsCheck [11] libraries and can be run using the basic dotnet test command.
- A software artifact (CorDeGen.Benchmark) containing performance tests of the developed implementations (see below).
- A command-line interface application (CorDeGen.CLI) that uses the CorDeGen library to provide the ability to generate corpora for the end user.

Experiments: benchmarking

In order to study the effect of slowing down the process of generating text data corpus when using variants of the proposed CorDeGen+ method, benchmarking was performed within the framework of this study, using the developed software implementation.

The BenchmarkDotNet library [12], which is standard on the .NET platform, was used to write and run benchmarks of developed implementations of language variations of the proposed CorDeGen+ method. This library greatly simplifies the benchmarking process by automatically selecting the required number of methods call repetitions, automatically performing warm-up and jitting [13]. Also, this library automatically performs statistical processing of the obtained results [13].

As values for the corpus size parameter during benchmarking, members of the geometric progression series with the parameters $b_1 = 100, q = 5, n = 6$ were used. The choice of such a progression step is because it is the closest integer value to $\sqrt[1,5]{10}$, therefore, with such a progression step, with each new value of the corpus size parameter, the generation time will increase approximately 10 times (with the theoretic asymptotic computational complexity of the method algorithm equal to $O(N^{1,5})$). As the first term of the series, the value 100 is chosen so that the sixth, last value is large enough, but such that it can also be encountered in practice.

The results of benchmarking on a physical machine (CPU: 6 physical / 12 logical cores, 2.60GHz; RAM: 16 Gb, 2667 MHz) are shown in Table 1.

As can be seen from Table 1, the length of the interquartile range for all methods and variations is small in relative terms, which indicates sufficient accuracy of the obtained results.

Fig. 4 shows the ratio of the mean corpus generation time (arithmetic) by each proposed language variation of the CorDeGen+ method to the mean corpus generation time (arithmetic) of the same size by the basic method.

Table 2

Efficiency testing results (100 – 312500 unique terms)

Method	Min	Q1	Median	Q3	Max
100 unique terms (μs)					
CorDeGen	13.251	13.281	13.304	13.355	13.409
CorDeGen+$^{(0-9)}$	17.539	17.591	17.611	17.669	17.789
CorDeGen+$^{(EN)}$	17.746	17.935	17.967	18.058	18.191
CorDeGen+$^{(DE)}$	17.911	18.157	18.393	18.415	18.511
CorDeGen+$^{(FR)}$	18.068	18.190	18.385	18.605	18.688
CorDeGen+$^{(IT)}$	18.227	18.292	18.683	18.721	18.812
500 unique terms (μs)					
CorDeGen	100.921	104.917	105.044	105.143	106.025
CorDeGen+$^{(0-9)}$	133.814	134.011	134.121	134.429	134.69
CorDeGen+$^{(EN)}$	130.421	131.264	131.438	131.564	131.973
CorDeGen+$^{(DE)}$	124.917	129.257	130.722	130.846	131.354
CorDeGen+$^{(FR)}$	131.153	132.031	132.294	132.603	132.832
CorDeGen+$^{(IT)}$	132.569	132.741	132.918	133.194	133.632
2500 unique terms (ms)					
CorDeGen	1.241	1.285	1.291	1.293	1.296
CorDeGen+$^{(0-9)}$	1.389	1.408	1.426	1.436	1.462
CorDeGen+$^{(EN)}$	1.445	1.457	1.464	1.501	1.505
CorDeGen+$^{(DE)}$	1.428	1.456	1.458	1.462	1.464
CorDeGen+$^{(FR)}$	1.483	1.550	1.554	1.556	1.565
CorDeGen+$^{(IT)}$	1.402	1.412	1.441	1.443	1.45
12500 unique terms (ms)					
CorDeGen	13.141	13.313	13.417	13.455	13.69
CorDeGen+$^{(0-9)}$	13.873	14.014	14.102	14.193	14.333
CorDeGen+$^{(EN)}$	13.693	13.788	13.824	13.982	14.139
CorDeGen+$^{(DE)}$	13.909	14.101	14.174	14.304	14.45
CorDeGen+$^{(FR)}$	13.424	13.589	13.705	13.767	13.85
CorDeGen+$^{(IT)}$	13.333	13.591	13.631	13.695	13.918
62500 unique terms (ms)					
CorDeGen	116.149	117.187	117.401	117.867	118.224
CorDeGen+$^{(0-9)}$	120.237	122.383	122.864	124.039	124.413
CorDeGen+$^{(EN)}$	119.983	122.857	123.987	125.128	127.743
CorDeGen+$^{(DE)}$	125.013	127.663	128.072	128.471	132.337
CorDeGen+$^{(FR)}$	124.161	124.924	126.113	127.246	128.994
CorDeGen+$^{(IT)}$	123	124.084	124.593	127.219	128.401
312500 unique terms (s)					
CorDeGen	1.159	1.176	1.179	1.187	1.202
CorDeGen+$^{(0-9)}$	1.227	1.235	1.247	1.25	1.27
CorDeGen+$^{(EN)}$	1.186	1.2	1.204	1.21	1.219
CorDeGen+$^{(DE)}$	1.165	1.182	1.186	1.188	1.193
CorDeGen+$^{(FR)}$	1.216	1.244	1.247	1.258	1.285
CorDeGen+$^{(IT)}$	1.245	1.264	1.265	1.267	1.269

As can be seen in Fig. 4, as the size of the corpus increases, this ratio decreases rapidly for each language variation until it is fixed at more or less the same level. For the largest tested corpus size, the ratio is in the range of 1.002 – 1.068, i.e. the use of language variations of the proposed CorDeGen+ method slows down the corpus generation process by 0.2 – 6.8% compared to the basic method.

Experiments: generation results

To demonstrate the superiority of the proposed CorDeGen+ method over the basic CorDeGen method, a similar clustering example given in [5] was used in the study.

In [5] it is shown that the result of clustering the corpus generated by the basic CorDeGen method using the k-means method with the parameter k = 2 is two clusters of the same size (if the number of documents is even) or almost the same size (if the number of documents is odd). At the same time, the numbers of the texts that fall into these two clusters form continuous ranges. However, this statement is true only when there is no preprocessing of the corpus to remove stop words.

The developed software implementations of the CorDeGen+$^{(EN)}$ language variation and the basic method were used for experimental verification and Microsoft.ML library [14] was used for clustering, as in [5].

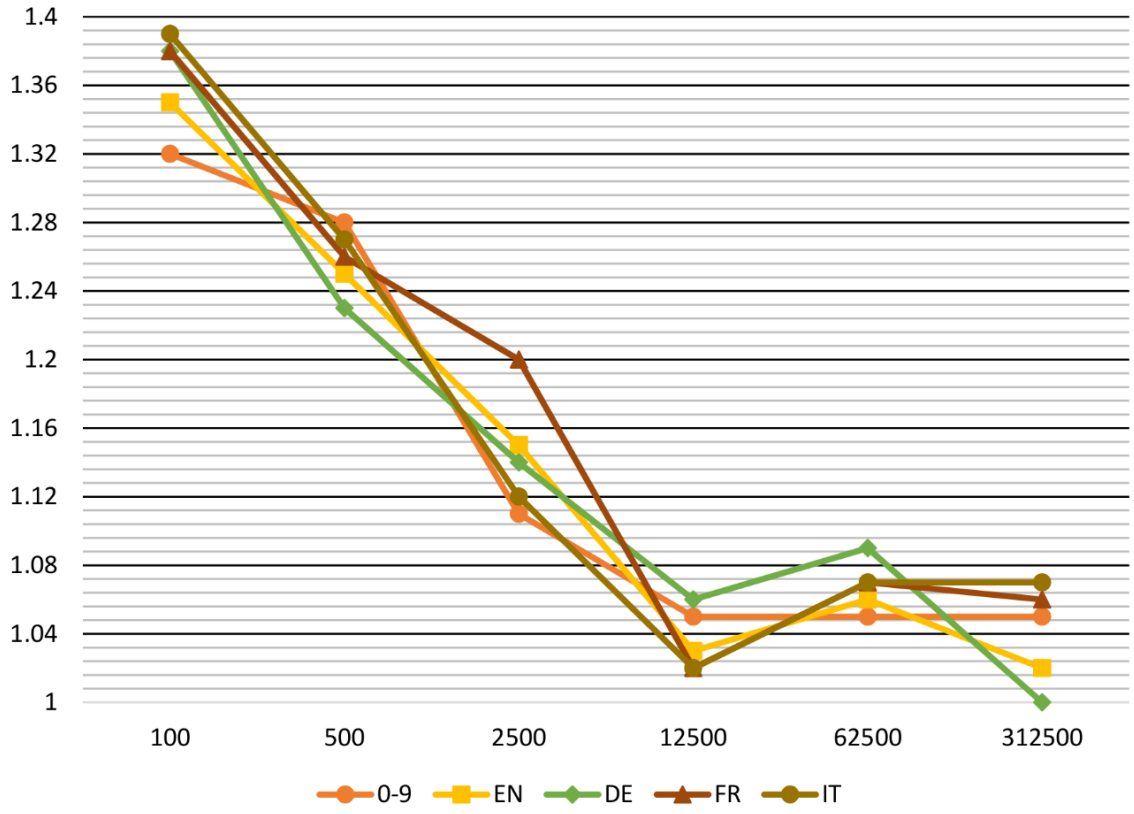

Figure 4. The ratio of the mean execution time (arithmetic) of CorDeGen+ method variants to the basic method

Fig. 5 shows the result of corpus clustering with the size of 130321 unique terms, generated by the CorDeGen and CorDeGen+$^{(EN)}$ methods with no previous corpus processing before clustering. As can be seen, both results are as expected – the obtained clusters are of almost the same size (since there are 19 documents) and the document numbers form continuous ranges.

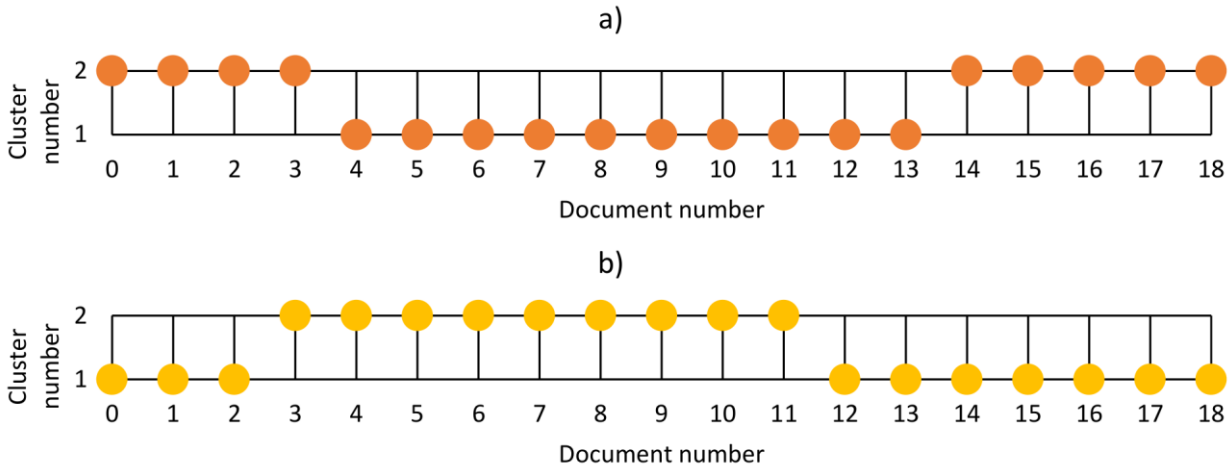

Figure 5. Results of corpus clustering without pre-processing, generated: a) by the basic CorDeGen method; b) by the CorDeGen+$^{(EN)}$ method

If, on the other hand, the corpora generated by both methods are clustered after first performing their pre-processing – removing the stop words – then the obtained results will correspond to those shown in Fig. 6.

As can be seen in Fig. 6, the clustering results of the corpus generated by the basic CorDeGen method differ from those expected. Although the numbers of texts included in both clusters form continuous ranges, the sizes of the clusters differ significantly: 7 and 12 texts, respectively. This result is explained by the distortion of the distribution of terms between the generated texts, which is caused by the removal of a part of the terms during pre-processing, as was noted in the analysis of the basic method. However, the proposed CorDeGen+ method does not have this disadvantage, so the expected result is obtained for it even in the case of performed preprocessing.

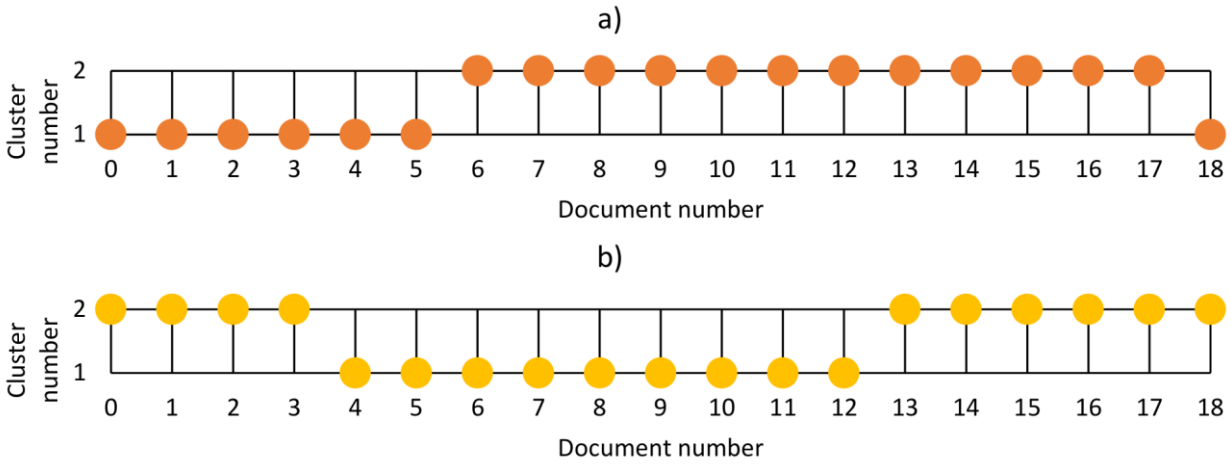

Figure 6. Results of corpus clustering with the removal of stop words, generated: a) by the basic CorDeGen method; b) by the CorDeGen+$^{(EN)}$ method

Conclusions

This study shows the feasibility of developing new and improving existing methods of generating corpora of text data intended for use in solving software engineering tasks in the context of natural language processing information systems. The performed analysis of the existing CorDeGen method revealed its certain disadvantages, which can lead to problems when using the corpus in practice with the enabled pre-processing.

Based on the performed analysis, the CorDeGen+ method is proposed, which solves the identified disadvantage of the basic CorDeGen method by introducing an additional stage of checking terms for admissibility during the generation of the corpus. At the same time, the asymptotic computational complexity of the algorithm implementing the proposed method remains unchanged compared to the basic method. This paper presents language variations of the proposed method for the four most common European languages because the additional verification stage is language-dependent.

Experimental verification (using the developed software implementation) confirmed the correctness of both main assumptions of this study: about a slight slowdown in the process of generating corpora by the proposed method and about the absence of distortion of the results of processing the generated corpus after removing stop words. As the size of the corpus increases, the degree of relative slowing down of the generation process drops sharply and for large corpora does not exceed 7%, while the clustering results of the corpus generated by the proposed method do not change even after preprocessing.

Further research on the topic of this study can be carried out in the following theoretical and practical directions:

- Fixing other drawbacks of the basic CorDeGen method.
- Consideration of other language variations of the proposed CorDeGen+ method, which were not considered in this paper.
- Implementation of the proposed CorDeGen+ method for different software platforms in different programming languages.

The created software reference implementation is published under the open MIT license and is available at the link: https://github.com/yakivyusin/CorDeGenComplete/tree/2.1.0.

References

1. Al-Thwaib E., Hammo B. H., Yagi S., “An academic Arabic corpus for plagiarism detection: design, construction and experimentation,” International Journal of Educational Technology in Higher Education Vol. 17 (1), (2020). https://doi.org/10.1186/s41239-019-0174-x.

2. Lichtarge J., “Corpora generation for grammatical error correction,” in Proc. of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (2019). https://doi.org/10.18653/v1/N19-1333.

3. Aichaoui S. B., Hiri N., Dahou A. H., “Automatic Building of a Large Arabic Spelling Error Corpus,” SN Computer Science Vol. 4, article number 108, (2023). https://doi.org/10.1007/s42979-022-01499-x.

4. Tanaka K., Chu C., Kajiwara T., “Corpus Construction for Historical Newspapers: A Case Study on Public Meeting Corpus Construction Using OCR Error Correction,” SN Computer Science Vol. 3, article number 489, (2022). https://doi.org/10.1007/s42979-022-01393-6.

5. Yusyn Y., Zabolotnia T., “Text data corpora generation on the basis of the deterministic method”, KPI Science News, no. 3, pp. 38–45, (2021). https://doi.org/10.20535/kpisn.2021.3.240780. [in Ukrainian]

6. Brigadir I., Nothman J., “stopwords/en/terrier.txt,” (2016). [Online]. Available: https://github.com/igorbrigadir/stopwords/blob/master/en/terrier.txt.

7. Gene D., “stopwords-iso/stopwords-de,” (2020). [Online]. Available: https://github.com/stopwords-

iso/stopwords-de/blob/master/stopwords-de.txt.

8. Gene D., “stopwords-iso/stopwords-fr,” (2020). [Online]. Available: https://github.com/stopwords-iso/stopwords-fr/blob/master/stopwords-fr.txt.

9. Gene D., “stopwords-iso/stopwords-it,” (2020). [Online]. Available: https://github.com/stopwords-iso/stopwords-it/blob/master/stopwords-it.txt.

10. .NET Foundation and contributors, “Home > xUnit.net,” (2019). [Online]. Available: https://xunit.net/.

11. Aichernig B., Schumi R., “Property-based Testing with FsCheck by Deriving Properties from Business Rule Models,” in Proc. of 2016 IEEE Ninth International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), 13th Workshop on Advances in Model Based Testing (A-MOST 2016), (2016).

12. .NET Foundation and contributors, “Overview | BenchmarkDotNet,” (2018). [Online]. Available: https://benchmarkdotnet.org/articles/overview.html.

13. Akinshin A., “Pro .NET Benchmarking: The Art of Performance Measurement”, Apress, (2019).

14. Microsoft, “ML.NET | Machine Learning made for .NET,” (2018). [Online]. Available: https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet.