Text Borrowings Detection System for Natural Language Structured Digital Documents

Olena Kuropiatnyk [0000-0003-2286-884X], Viktor Shynkarenko [0000-0001-8738-7225]

Dnipro National University of Railway Transport named after Academician V. Lazaryan
2, Lazaryan str., Dnipro, Ukraine
olena.kuropiatnyk@gmail.com, shinkarenko_vi@ua.fm

Abstract. Interpretation of results is an important stage in text borrowings detection systems. Necessary to take into consideration the tree structure of the document and the general content of structural elements (sections) is the reason for that. In article method comparison of structured document is developed. Formalization of comparison document process is based on constructive-synthesizing modeling. Document structure is processed using templates. They contain information about section and subsections titles and keywords sets. The base of natural language text comparison is text graph representation model. It represents a text as graphs set for improving borrowings retrieval in texts of database. On base of these models and method text borrowings detection system is developed for comparison digital structured natural language documents. The paper presents the features of the system and its advantages. System architecture is described and its time efficiency investigated.

Keywords: natural language text, structured document, text borrowings detection, plagiarism, constructive-synthesizing modeling, constructor

1 Introduction

Borrowing detection in natural language texts is one of the tasks of NLP. Its relevance is increasing with the development of information technology and the growth of information in the public access.

Text borrowing can be legitimate: properly designed quotes, a description of the techniques and algorithms with specification the author and references, etc. Illegal borrowings (plagiarism) are now widespread in many areas of society life, including academic [1 – 4]. There are several ways to fight is. They are organizational and administrative measures, technological and information-technical means [5, 6]. Information-technical means include anti-plagiarism programs that focus on textual borrowing detection [7 – 10].

The main stages the anti-plagiarism programs functioning are texts pre-processing, comparison – borrowings detection, results interpretation.

Pre-processing stage may include case-insensitive typing, deleting stop words, deleting or ignoring punctuation marks, blank paragraphs, extra spaces, spell checking, etc. [11].

The usage of shingles [12, 13], multivariate graph approaches [14, 15, 16], n-grams [9, 14, 17], TF-IDF based algorithms [17, 18], SCAM algorithms [19, 20] etc. are widespread for comparison stage. These algorithms and approach solve only detection task for entire text and do not for its parts. Their apply result is single mark for whole text as a percentage of borrowing or uniqueness. Algorithms and approach can be used for parts text, but need definition of text borrowing evaluation function and method for text split.

Interpreting the text borrowings detection results is particularly important. There are a large number of technical tools (resources) for borrowings detection that provide an ultimate text estimate as a percentage of borrowing or uniqueness. However, in the course of the 25 resources analysis [22], authors don't found resources, which take into account the structural features of the checked document. For example, some sections describe known techniques completing the views, and then textual borrowings are acceptable. However, existing tools do not take this into account when evaluating the document.

This paper is devoted to the development of a method of structured documents comparison and its computer implementation using constructive-synthesizing modeling [23, 24] and graph representation of text [25, 26]

2 The Task of Text Borrowings Detection in Digital Structured Natural Language Documents

Structured digital documents are documents represented by doc / docx files format that have a logical structure in content and the appropriate formatting. Logical structure determinates sections and subsections order.

Such documents are e-books and synopses, reporting documentation for educational (course, laboratory) and qualification papers (diploma thesis, PhD dissertations) etc. Next, it will be about qualification works.

Borrowings detection in structured documents is complex due to the following factors:

- the necessity to evaluate each section separately and the document as a whole;
- formatting features. The use of different techniques to indicate the structural elements of a document;
- the necessity to select materials for comparison according to the content of the structural section;
- elements variety. The presence of text fragments in different languages, hyperlinks and cross references, images, tables, formulas, other objects.

Taking into account of the above, text borrowings detection in structured documents further needs to be developed:

- representation models and comparison methods for structured documents;
- document pre-processing algorithms to reduce the amount of material to be checked by omitting some items;

– reading mechanisms for document files with text formatting.

The goal of this research is modeling process of comparing structured digital documents for construct comparison documents method. Main purpose of this method is automated processing of structured digital documents for text borrowings detection.

Research questions:

– definition of the features and the basic stages of processing structured document in borrowings detection tasks;
– definition and formalized description of the algorithms processing structured document by means of constructive-synthesizing modeling and development of structured documents compression method ;
– object-oriented modeling of structured documents and their processing;
– creation relationship between object-oriented and constructive-synthesizing models of processing structured document for development software based on them.

3 Modeling the Process of Comparing Structured Digital Documents

The process of comparing structured documents is the orderly sequence of actions performed on a document submitted for borrowings detect and the structured documents base for comparison. The actions can be completed in the following stages:

1. forming a document tree - defining structural elements (sections, subsections) and their hierarchy;
2. search in the documents database of structural elements for comparison;
3. pre-processing of the text of structural elements;
4. comparison of documents and obtaining an assessment of each structural element and the document as a whole.

To formalize this process, the means of constructive-synthesizing modeling [23] was apply. Authors defined the constructor of the process and performed refinement transformations: specialization, concretization, interpretation, and implementation.

3.1 Specialization of the Constructor of the Structured Documents Comparison Process

The specialization of the constructor [23] involves the determination of the semantic component of the carrier:

$$C = \langle M, \Sigma, \Lambda \rangle \ {}_s\!\mapsto C_{CP} = \langle M_{CP}, \Sigma_{CP}, \Lambda_{CP} \rangle, \quad (1)$$

where ${}_s\!\mapsto$ is operation of specialization, M_{CP} is the heterogeneous replenishable carrier, Σ_{CP} is the signature of operations and relation for construction, Λ_{CP} is the information support that includes the ontology (the constructive basics are outlined in

[24]), the purpose, rules, constraints, initial conditions, and construction completion conditions.

Ontology of constructor C_{CP}. Carrier is $M_{CP} \supset \{T \bigcup N\}$. Terminals ($T$) include finite set of algorithms for operations on the document, its attributes and text $\{D_i |_{X_i}^{Y_i}\}$, where D_i is identifier of algorithm and X_i, Y_i are sets of its definitions and values. Non-terminals are auxiliary elements for identify abstract algorithms set.

A document to which algorithms apply has many attributes-sections set $\{sections_i \leftarrow DOC\}$, which have the following attributes: title, level, text, and general content.

Algorithms for processing the document and its attributes:

- $D_1 |_{doc}^{sections}$ for formation *sections*, *sections* is structural tree for document *doc*;
- $D_2 |_{section, template}^{numbers}$ for definition the general content of the section (*section*) according to a given structure template (*template)*, *numbers* is the section numbers in the template, $count \leftarrow numbers$ is number sections' quantity;
- $D_3 |_{section, num}^{numbers}$ for comparison of the general contents of the section (*section)* with the corresponding element of the structure of the template with the number *num* (setting the pattern in manual mode);
- $D_4 |_{sections, db}^{s\_texts, db\_texts}$ for selection *s_texts* from *sections* by structural tree of the document (*tree*) for comparison with texts (*db_texts*) of data base documents *db*;
- $D_5 |_{fragments, marks, doc\_size}^{ul\_mark}$ for calculation the percentage of borrowings (*ul_mark*) in a document, where *doc_size* is size of the document, fragments, marks are the text and the percentage of borrowings in the section, respectively.

Algorithms for the text processing:

- $D_6 |_{text}^{text'}$ for *text* preprocessing, *text'* is preprocessed text;
- $D_7 |_{text}^{graphs}$ for construction the graphs set (*graphs)* for *text;*
- $D_8 |_{text, graphs}^{fragments, positions, marks}$ for comparison of *texts* with a set of *graphs*, *fragments*, *positions* are borrowed text fragments and their positions in the text, *marks* is percentage mark of each fragment.

The signature $\Sigma_{CP} = \langle \Xi, \Theta, \Phi, \{\rightarrow\}, \Psi \rangle$ consists of operations sets and corresponding relationships of the same name, where $\Xi \supset \{\cdot, :\}$ are the operations of linking and transforming carrier elements, $\Theta = \{\Rightarrow, |\Rightarrow, |\models\}$ are the operations of substitution and inference, Φ is relationships and attribute operations, $\{\rightarrow\}$ are the substitution relations, $\Psi = \{\psi_i : \langle s_i, g_i \rangle\}$ is the set of substitution rules, s_i is the sequence of substitution relations, g_i is the set of attribute operations. If attribute operations are not performed, the substitution rule will look like this $\langle s_i, \varepsilon \rangle$, where ε is empty symbol.

The inference rules apply the relationship from Ξ, and the corresponding operations apply in the implementation of the constructor.

Operation $\cdot(D_i, D_j)$ is concatenation of algorithms, implies sequential execution D_j after D_i.

Operation $:(b, A)$ is execution the algorithm A if the expression b is true ($b = true$).

The purpose of the constructor is to form structured documents comparison algorithms.

Initial conditions for construct: inference begins with a non-terminal CP. There are the structured document *doc*, $doc\_size \leftarrow\!\lrcorner doc$ is the document text size, *db* is the structured documents database, *template* that describes the possible sections in the document; *sections* is the structural tree for document *doc*, each element of which (section) has attributes of title (title) and text of the section / subsection (text).

Termination condition of constructing: the form does not contain non-terminals.

3.2 Document Comparison Constructor Interpretation

Let's interpret the constructor by the algorithmic structure C_A

$$\begin{array}{l} \left\langle C_{CP}, C_{A,CP} = \left\langle M_{A,CP}, V_{A,CP}, \Sigma_{A,CP}, \Lambda_{A,CP} \right\rangle \right\rangle_I \mapsto \\ {}_I \mapsto \left\langle {}_A C_{CP},\ {}_A C_{CP} = \left\langle M_1, \Sigma_1, \Lambda_1 \right\rangle \right\rangle \end{array}, \qquad (2)$$

where $\Lambda_1 \supset \Lambda_{CP}$, $V_{A,CP} = \{A_i^0 \mid_{X_i}^{Y_i}\}$ is the basic algorithms set [23], X_i, Y_i are the sets of definitions and values of an algorithm $A_i^0 \mid_{X_i}^{Y_i}$. $\Lambda_{A,CP} = \{M_{A,CP} = \bigcup_{A_i^o \in V_{A,SP}} (X(A_i^0) \bigcup Y(A_i^0)) \bigcup \Omega(C_{CP})\}$ is the heterogeneous carrier, $\Omega(C_{CP})$ is the set of algorithms implemented by the constructor C_{SC}; $\Lambda_1 = \{(A_1^0 \mid_{A_i, A_j}^{A_i \cdot A_j} \leftarrow\!\lrcorner ":");\ (A_2^0 \mid_b^{A_i} \leftarrow\!\lrcorner ":");\ (A_3 \mid_{l_h, l_q, f_i}^{f_i} \leftarrow\!\lrcorner "\Rightarrow");\ (A_4 \mid_{f_i, \Psi}^{f_j} \leftarrow\!\lrcorner "\mid\Rightarrow");$ $(A_5 \mid_{\sigma, \Psi}^{\overline{\Omega}} \leftarrow\!\lrcorner "\mid\models");\ \Lambda_1 \supset \Lambda_{CP}$.

Constructor ${}_A C_{CP}$ includes performing operations algorithms:

- $A_1^0 \mid_{A_i, A_j}^{A_i \cdot A_j}$ for algorithms composition, $A_i \cdot A_j$ is sequential execution of the algorithm A_j after algorithm A_i;
- $A_2^0 \mid_{b, A_i}^{A_i}$ for conditional execution: algorithm A_i executes if the expression b is true;
- $A_3 \mid_{l_h, l_q, f_i}^{f_i}$ for substitution, l_h, l_q, f_i are forms;

– $A_4|_{f_i,\Psi}^{f_j}$, $A_5|_{\sigma,\Psi}^{\overline{\Omega}}$ for partial and complete inference, where f_i, f_j are forms, σ is axiom, $\overline{\Omega}$ is the set of constructed structures.

Let's execute concretization of the constructor (2):

$$_{I,C_A}C_{CP} = \langle M_{CP}, \Sigma_{CP}, \Lambda_{CP}, Z \rangle \ _K\mapsto_{K,I,C_A} C_h = \langle M_{CP}, \Sigma_{CP}, \Lambda_2, Z \rangle, \quad (3)$$

where $\Lambda_2 = \Lambda_1 \bigcup \Lambda_3$, $\Lambda_3 \supset \{M_{CP} = T \bigcup N$, $T = \{\{D_i^0\},\{D_j^1\}\}\}$ is the terminals set, $N = \{CP, \beta, \gamma, \delta\}$ is the non-terminals set, CP is the initial non-terminal.

Information support: substitution rules. Consider the inference rules that allow us to formalize the process of comparing structured documents according to the previously defined stages.

Definition of structural elements (sections, subdivisions) of the document:

$$s_1 = \left\langle CP|_{doc,template}^{marks,ul\_marks} \to D_1|_{doc}^{sections} \cdot \beta|_{section,template}^{numbers} \right\rangle, \quad (4)$$

$$s_2 = \left\langle \beta|_{section,template}^{numbers} \to D_2|_{section,template}^{numbers} \cdot \beta|_{section,template}^{numbers} \right\rangle, \quad (5)$$

$$s_3 = \left\langle \beta|_{section,template}^{numbers} \to A_2^0|_{count\_numbers=0}^{D_3} : D_3|_{section,num}^{numbers} \cdot \beta|_{section,template}^{numbers} \right\rangle, \quad (6)$$

$$s_4 = \left\langle \beta|_{section,template}^{numbers} \to D_2|_{section,template}^{numbers} \cdot \gamma|_{section,template}^{numbers} \right\rangle, \quad (7)$$

$$s_5 = \left\langle \gamma|_{section,template}^{numbers} \to A_2^0|_{count\_numbers=0}^{D_3} : D_3|_{section,num}^{number} \cdot \gamma|_{section,template}^{numbers} \right\rangle, \quad (8)$$

where $count\_numbers = 0$ is an indication that the section did not match the template.

Retrieval for documents of structural elements that can be compared and preprocessed for structural elements

$$s_6 = \left\langle \gamma|_{sections,template}^{text',db\_texts} \to D_4|_{sections,db}^{s\_texts,db\_texts} \cdot D_6|_{s\_texts_i}^{text'} \cdot \delta|_{text',db\_texts}^{ul\_mark} \right\rangle. \quad (9)$$

Comparison documents and getting marks for each structural element and document as a whole

$$s_7 = \left\langle \delta|_{text',db\_texts}^{ul\_mark} \to D_7|_{db\_texts_i}^{graphs} \cdot D_8|_{text',graphs}^{fragments,positions,marks} \cdot \delta|_{text',db\_texts}^{ul\_mark} \right\rangle, \quad (10)$$

$$s_8 = \left\langle \delta|_{text',db\_texts}^{ul\_mark} \to D_7|_{db\_texts_i}^{graphs} \cdot \right.$$

$$\left. \cdot D_8|_{text',graphs}^{fragments,positions,marks} \cdot D_5|_{fragments,marks,doc\_size}^{ul\_mark} \right\rangle. \quad (11)$$

4 Method of Structured Documents Comparison

The algorithms $\{D_i\}$ and the constructive-synthesizing model (1) – (11) make up the method of comparing structured documents, the scheme of which is shown in Fig. 1. Block markings $s_1 - s_8$ comply with the rules (4) – (11) and its combinations of constructor (1) – (3).

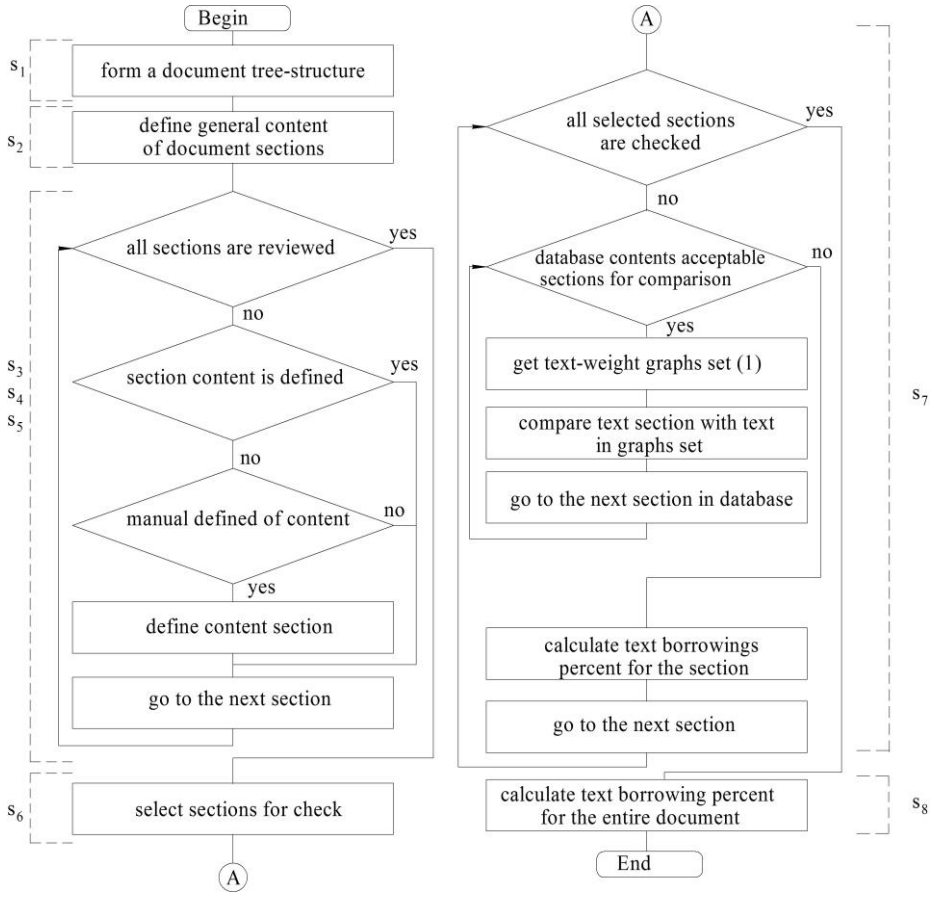

Fig. 1. Scheme for the method of structured documents comparison

In the scheme, text graphs sets in the action (1) were constructed in accordance with the text graph representation models and the text-weight graph compression method [25, 26].

The main idea of the graph representation of the text is to form a directed graph with weighted vertices and edges. The vertex weight is symbols of text; the weight of the edge is the set of cycle numbers into which this edge enters. It is advisable to preprocess of the text before form the graph.

Compression of text-weight graph occurs by combining vertices that are connected by edges of equal weight. It reduces the needed memory volume by approximately 97%. Moreover, the graph structure is simplified by 90 – 97%.

5 Computerized Comparison of Structured Documents

Object-oriented modeling of a structured document and other components were performed for the computerized implementation of the proposed method of comparison (Fig. 2).

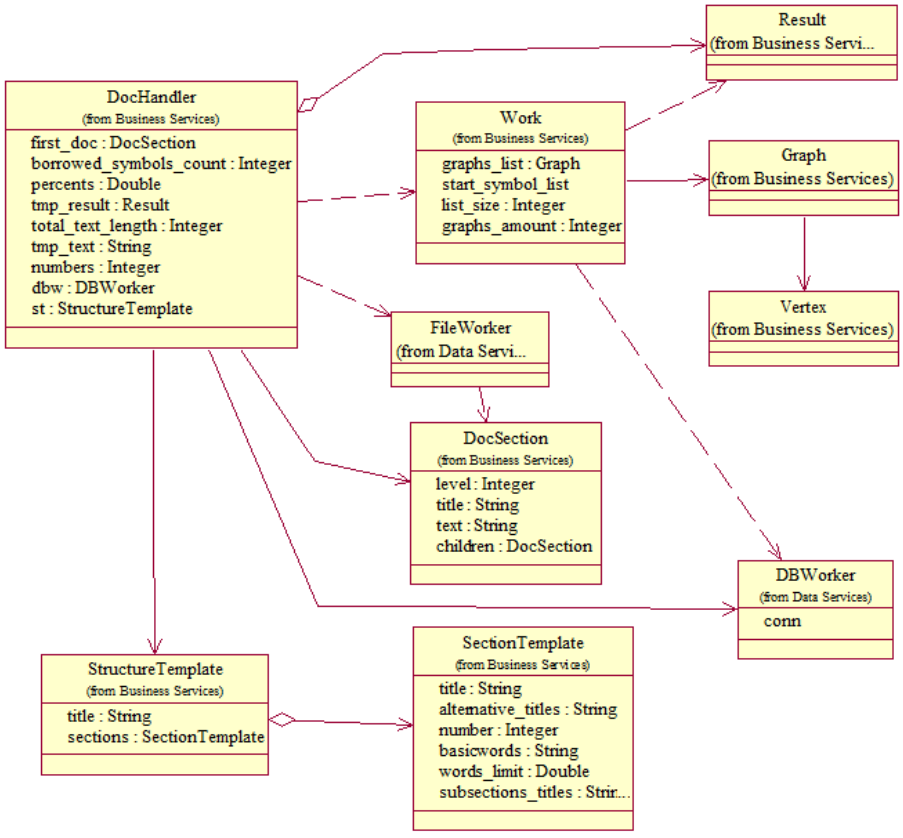

Fig. 2. Object-oriented modeling of structured document comparison

The document model is represented by the *DocSection* class, which includes such attributes as *level* of a part of the document (section, paragraph, subparagraph, up to the ninth degree of nesting), *title*, section *text* that belongs to a part of the document and delimits by headings.

The base of *DocSection* is Composite pattern, which integrates objects into a tree structure to represent the hierarchy from part to whole

Parsing of the document according to the structure is assigned to the *FileWorker* class, which creates *DocSection* object by the xml structure of the document file. *DocHandler* coordinates the rest of *DocSection* processing. Thus, three classes perform representation, initialization and access to a structured document.

To determine the general content of the sections, a structural template modeled by the classes *StructureTemplate*, *SectionTemplate* is used. The template contains information about the section and sub-sections headings of the document and keywords sets.

To compare the text of structural sections, a graph representation of the text is used [26]. Here it is represented by classes *Graph* and *Vertex*. A *text* attribute of the *DocSection* class represents the text of the compared section. The compared text is presented as a set of graphs – *Work* class.

To implement the method of comparing structured documents (Fig. 1) and the corresponding constructive and object-oriented (Fig. 2) models [26], a software system "StructuredDocComparission" (SDC) was developed with the graphic user interface (GUI). It provides cooperation with the document structure (Fig. 3).

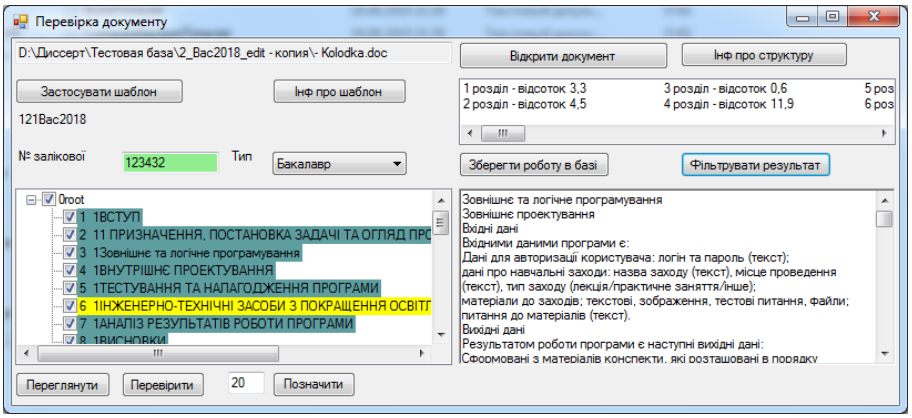

Fig. 3. The main window of the program for comparison by template (GUI language – Ukrainian)

Features of the SDC system are:

- usage a template for construction tree structure of document and text separation by section. It helps decrease texts volume that use in comparison stage. XML-representation of template makes it understandable to the user and facilitates its correction;
- check of separated sections of the document and obtaining a percentage of text borrowings by sections and the entire document. It allows performing further qualitative analysis of the submitted document content;
- results filtration for structured documents. It allows user to determine the size of text fragments that are considered to be borrowed and to obtain a recalculated estimate without re-comparing the document;

- storage in the base of graph representation of texts. This allows not to waste time preparing the texts of the database for comparison;
- usage the principles of friendliness and feedback to design a user interface simplify interaction with the system.

6 Experimental Research

Authors performed experimental researches to determine the time efficiency of the SDC system.

Experimental base – 25 files in docx format. Files are the documentation for the diploma thesis of the Bachelor of Software Engineering DNURT-2018 (size: 0.7 MB – 27.3 MB, 107.2 – 310.3 thousand characters). Each file contains 32-35 sections (explanatory note and technical documentation), each of which is matched by template to determine the general content and subsequent selection of data from text documents database. Primary language of documents – Ukrainian, languages of section "Program text": C/C++, C#, html, javascript etc. The domain is determined by the specialty and area of IT knowledge. Text databases of the above mentioned diploma thesis, size 5,10..25 qualifying works, which are formed by the developed software. Local web server xampp v 3.2.2 (components used: Apache, MariaDB). The experiment was executed on a PC with the following specifications: Intel Pentium (R) Dual Core CPU, L1 code / L1 data cache / L2 – 2 * 32/2 * 32 / 1024K, clock speed / system bus frequency / memory frequency – 2.3 GHz / 400 MHz / 400 MHz, OP access time (read / write) 5751/4253 MB / s, operating system – MS Windows 7 Ultimate SP1.

Execution of the experiment. Each of the 25 files of the experimental database is compared with the bases of 5,10..25 works formed from the same files. The texts of the checked files and the texts of the database were pre-processed (removal of unnecessary spaces, unification of punctuation marks, etc.). Comparison accuracy is one word. A word is a sequence of letters and / or numbers separated by spaces or / and punctuation delimiters. The files in the database are not duplicated. Time efficiency metrics are determined by a text matching operation. The matching operation involves the following stages: pre-processing (selection of information in the database, allocation of memory for storing results, text pre-processing); constructing text graph representation from base (multiple sets of graphs); comparison of the submitted text with the texts of the base; work evaluation (determination of the total percentage of borrowings according to all works of the base).

Experiment results. It is established that the average time of comparison of the structured document is from 11 to 65 seconds on a base of 0.6 to 3.8 million characters. It has a linear dependence on the size of the checked document and base.

7 Conclusion

To processing natural language text constructive-synthesizing model of comparison process was developed. Special comparison method was form using this model.

Constructive-synthesizing and object-oriented models of the text graph representation previously developed by authors are used to increase time effectiveness of comprising process.

That allowed to develop software system for text borrowings detect in structural digital document. When designing the system architecture, an object-oriented model of a structured document and algorithms for its processing were developed.

The advantage of the developed system compared to analog tools is that the percentage of text borrowings is calculated not only for the entire document, but also for any separate section (unit).

Promising scopes for research are parallel graphs processing and optimization of procedures for restoring graphs sets from a database to increase the time efficiency of the system.

References

1. Byvaltsev, V. A. et al.: Plagiarism and academic integrity in science. Bulletin of the Russian Academy of Medical Sciences. vol. 72, no. 4, pp. 299–304 (2017). doi: 10.15690/vramn788
2. Pierce, J., Zilles, C.: Investigating student plagiarism patterns and correlations to grades. In: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education, pp. 471-476 (2017). doi:10.1145/3017680.3017797
3. Pàmies, M. M., Valverde, M., Cross, C.: Organising research on university student plagiarism: a process approach. Assessment & Evaluation in Higher Education, pp. 1-18 (2019). doi:10.1080/02602938.2019.1658714
4. Svyrydenko, D. et al.: Plagiarism challenges at Ukrainian science and education. Studia Warmińskie, vol. 53, pp. 67-75 (2016).
5. Singh, B.: Preventing the plagiarism in digital age with special reference to Indian Universities. International Journal of Information Dissemination and Technology, vol. 6, no. 4, pp. 281-287 (2016).
6. Sanz, J. J. G.: Detecting plagiarism in micro-blogging social networks. In: Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality, pp. 1-6 (2017). doi:10.1145/3144826.3145438
7. Lovepreet, V. G., Kumar, R.: Survey on Plagiarism Detection Systems and Their Comparison. In: Proceedings of the International Conference on ICCIDM 2018, vol. 990, no. 27. Springer (ed). Computational Intelligence in Data Mining (2019). – doi: 10.1007/978-981-13-8676-3_3
8. Heres, D., Hage, J.: A quantitative comparison of program plagiarism detection tools. In: Proceedings of the 6th computer science education research conference, pp. 73-82 (2017). doi: 10.1145/3162087.3162101
9. Chowdhury, H. A., Bhattacharyya, D. K.: Plagiarism: Taxonomy, tools and detection techniques. arXiv preprint arXiv:1801.06323 (2018)
10. Kumar, S., Boriwal, C.: Plagiarism Issues: Types, Tools and Remedies. Indian Journal of Agricultural Library and Information Services, vol. 35, no. 3, pp. 477 – 482 (2019)
11. Mahmoodi, M., Varnamkhasti, M.M.: Design a Persian Automated Plagiarism Detector (AMZPPD). International Journal of Engineering Trends and Technology (IJETT) ISSN 2231-5381. Vol. 8, No. 8. pp. 465-467 (2014). doi: 10.14445/22315381/IJETT-V8P280

12. Borodashchenko, A. Yu. et al.: Search algorithm for similar media publications. Internet Journal “Naukovedeniye.”, vol. 7, no 4 (29) (2015).
13. Platonov, A. A., Potapov, R. E.: Detection of duplicate articles in the system of automatic collection of information from open sources about the ecology situation. Bulletin of the Volgograd State Technical University, no. 6, pp. 79-82 (2015).
14. Foltýnek, T., Meuschke, N., Gipp, B.: Academic plagiarism detection: a systematic literature review. ACM Computing Surveys (CSUR), vol. 52, no. 6, pp. 1-42 (2019). doi: 10.1145/3345317
15. Franco-Salvador, M., Rosso, P., Montes-y-Gómez, M.: A systematic study of knowledge graph analysis for cross-language plagiarism detection. Information Processing & Management, vol. 52, no. 4, pp. 550-570 (2016). doi.org: 10.1016/j.ipm.2015.12.004
16. Osman, A. H., Salim, N., Elhadi, A. A. E.: A tree-based conceptual matching for plagiarism detection. In: International conference on computing, electrical and electronic engineering (ICCEEE), p. 571-579. IEEE (ed) (2013). doi: 10.1109/ICCEEE.2013.6634003
17. Bensalem, I., Rosso, P., Chikhi, S.: On the use of character n-grams as the only intrinsic evidence of plagiarism. Language Resources and Evaluation. vol. 53, no. 3, pp. 363-396 (2019). doi: 10.1007/s10579-019-09444-w
18. Bhujade, K. et al.: Plagiarism Detection Based on Neural Network (2019). doi: 10.32628/IJSRST19613
19. Dwivedi, J., Tiwary, A.: Plagiarism detection on bigdata using modified map-reduced based SCAM algorithm. In: International Conference on Innovative Mechanisms for Industry Applications (ICIMIA). pp. 608-610. IEEE (ed.) (2017). doi: 10.1109/ICIMIA.2017.7975533
20. Xie, R., Zhu, L., Cheng, Y.: Review of Copy Detection Techniques for Monolingual Natural-Language Documents. In: International Conference on Web Intelligence (WI). pp. 631-634. IEEE (ed.), IEEE/WIC/ACM (2018). doi: 10.1109/WI.2018.00-24
21. Dien, T. T., Han, H. N., Thai-Nghe, N.: An Approach for Plagiarism Detection in Learning Resources. In: International Conference on Future Data and Security Engineering. pp. 722-730. Springer, Cham (ed.) (2019). doi: 10.1007/978-3-030-35653-8_52
22. Shynkarenko, V., Kuropiatnyk, O.: Plagiarism detection problems and analysis software tools for its solve. Science and Transport Progress. Bulletin of Dnipropetrovsk National University of Railway Transport, no. 1 (67), pp. 131–142 (2017). doi: 10.15802/stp2017/94034.
23. Shynkarenko, V. I., Ilman, V. M.: Constructive-Synthesizing Structures and Their Grammatical Interpretations. Part I. Generalized Formal Constructive-Synthesizing Structure. Springer (ed.), Cybernetics and Systems Analysis ISSN: 1060-0396 (Print) 1573-8337 (Online), vol. 50, no 5. pp. 665 – 662 (2014). Part II. Refining Transformations. Vol. 50, no 6, pp. 829 – 841 (2014). doi: 10.1007/s10559-014-9655-z, doi: 10.1007/s10559-014-9674-9
24. Skalozub, V., Ilman, V., Shynkarenko, V.: Development of ontological support of constructive-synthesizing modeling of information systems. Eastern-European Journal of Enterprise Technologies, vol. 6, issue 4 (90), pp. 58–69 (2017). doi: 10.15587/1729-4061.2017.119497
25. Shynkarenko, V., Kuropiatnyk, O.: Constructive-synthesizing model of text graph representation. In: CEUR Workshop Proceedings, vol. 1631, pp. 63 – 72 (2016)
26. Kuropiatnyk, O.: Constructive and object-oriented modeling text for detection of text borrowings. System technologies, no. 4 (123), pp. 34–47 (2019). doi: 10.34185/1562-9945-4-123-2019-04