Automation of Template Formation to Identify the Structure of Natural Language Documents

Olena Kuropiatnyk and Viktor Shynkarenko

Dnipro National University of Railway Transport named after Academician V. Lazaryan, 2, Lazaryana str., Dnipro, 49010, Ukraine

Abstract

In the task of text borrowings and plagiarism detection, it is important to take into account the structure of the document. This allows getting a more accurate assessment of the text and reducing the volume of material for comparison. Using a template allows identifying the structure of the document. The paper presents a constructive synthesizing model for automating the construction of a structural template of a document. Possible implementations of some algorithms by means of programming in C# are considered. Their comparative assessment is performed. Possible modification of the template is presented to increase the importance of keywords and simplify the xml-tree, which is a template.

Keywords 1

Natural language, document comparison, plagiarism detection, document structure, document template, constructive-synthesizing modeling, constructor

1. Introduction

A document checking for matches is actual in the tasks of plagiarism detection, rewriting, information retrieval, etc. when a document processing to detect matches and text borrowings, the structure of the document has to be taken into account. This is due to the fact that the selection of the comparison base for the content of the section will not only reduce the probability and number of random matches at the level of words and separated phrases, but also reduce the volume of materials to be compared. In addition, part of the text borrowings may be admissible given that the given material is necessary for the completeness of the image and the author of the submitted for verification document does not claim authorship of this part. Such borrowings can be found, for example, in articles, abstracts, parts of qualifying works, which have a review character.

A special method has been developed to compare documents taking into account their structure by means of constructive modeling [1]. It involves comparing documents according to the structural template and a constructive-synthesizing model of a graph representation of the text [2]. This idea is developed in the paper: the issues of automation of template construction with the possibility of editing and append new structural elements are considered.

2. Goal and questions of the research

The goal of this research is modeling and automatizing of the process of forming a structural template of a document based on a selection of natural language digital documents. This template will be used to select information for comparing text documents to detect borrowings.

Research questions:

- identification elements that indicate digital representation of a document structure;

COLINS-2021: 5th International Conference on Computational Linguistics and Intelligent Systems, April 22–23, 2021, Kharkiv, Ukraine
EMAIL: olena.kuropiatnyk@gmail.com (O. Kuropiatnyk); shinkarenko_vi@ua.fm (V. Shynkarenko);
ORCID: 0000-0003-2286-884x (O. Kuropiatnyk); 0000-0001-8738-7225 (V. Shynkarenko);

- determination the main components of the process of forming a structural template of the document;
- development of a constructive-synthesizing model of structural template forming;
- software implementation of the constructive-synthesizing model of structural template forming for automation of template formation to identify the structure of natural language documents.

3. Methods

Means of constructive-synthesizing modeling [2] are used for solve the questions of the research in this paper. The modeling is based on the apparatus of formal languages and grammars. Usage a formalized object of constructor and methods of its transformation allows creating the model of structural template forming and automate corresponding processes. Methodology of object oriented modeling and programming, methods of processing xml-nodes and their attributes which implementation in C# and components from namespace Microsoft.Office.Interop.Word are used for software implementation of the constructive-synthesizing model of structural template forming. Method based on SR-estimation (described in this work) is used for evaluation the some algorithms time efficiency of developed model software implementation.

4. Definition of structural features of the document. Review of related works

To compare documents taking into account their structure, let's define the concept of a structured document, a structural element and the features by which it can be determined.

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 [1]. Structure items are units. They can be identified by the following features:

- formatting. There are use appropriate styles: headings or other styles that indicate a non-zero paragraph level;
- title and alternative titles;
- sub-units with corresponding names and attributes format;
- keywords in the unit text.

The simplest identification of the structure in terms of software implementation is the use of the first two features. The first three have already been partially used by the authors earlier [1]. But given the possible paraphrasing of the titles will be more accurate with using the keywords.

The process of finding keywords consists of three stages: pre-processing (depending on the text language), search (extraction) of keywords, get a list of words in the format required by the user (possible reduction of the sample) [3].

The tasks of pre-processing include the implementation of actions:

- "cleaning" of the text are the mechanical actions that do not require knowledge of the language and understanding of the text: removal of hidden and white text, addresses of cross-references, double spaces, reduction to a single register, etc.;
- deletion of stop words;
- reducing the impact of the use of inflections by stemmatization, lemmatization.

The latter tasks are language-dependent, and therefore the vast majority needs dictionaries. An alternative is to use the Porter algorithm [4, 5] and similar, adapted for the Ukrainian language.

It should be noted that pre-processing is also useful when working with the titles of structural elements of the document.

Keyword search algorithms can be divided into two categories: purpose (selection of those in the dictionary) and extraction (selection directly from the text) [6]. The latter are valuable for this work.

Keyword recognition can be done by statistical and / or structural methods. Among the first widely used was the use of TF * IDF is a statistical measure of the frequency of occurrence of the word in the document. Authors of work [6] propose a model based on Key phrase Extraction Algorithms (KEA), which uses such features as TF * IDF and the distance from the beginning of the document to the first

meeting. To select keywords, the KEA (Bayesian) classifier is used, which determines the weight assigned to a potential keyword and, based on it, marks the words as "key" and "non-key". Based on these data a model is built, which assumes classes (key, non-key) for a word or phrase, depending on the value of the calculated features.

As for structural methods, there are based on graphs and templates.

Keywords can be extracted based on the graph model of the linguistic corpus [7]. When defining keywords, a sample of the highest frequency (TF * IDF) of about 40 words is created - these words are potential vertices of the graph. Then choose the 20 words that occur in the largest number of sentences. A graph is constructed from these words: vertices are extracted words; an edge between vertices exists if the corresponding words occur in one sentence. The multiplicity of an edge indicates the number of such sentences. Then select 20 words that correspond to the vertices of the graph with the greatest degree - they are taken as key. The size of the selection by the number of words can vary.

Approaches to the formation of a graph for whole or filtered text (TextRank, Rake, DegExt), are considered in [3]. New implementations of keywords extraction algorithms are formed on their basis. TextRank is one of the most popular algorithms [8] and works by analogy with PageRank. Its main idea is graph forming for a text. In the keyword identification task, vertices are words, and for annotation and abstracting tasks vertices are sentences. For each vertex, the rating is calculated by the number of edges that end in the given vertex one and by the edge rating. The connection between sentences is determined by the presence of identical words in them, and the weight of the edge by their number. Thus, the most important are those sentences that have information from several others.

Authors of work [9] use the TextRank algorithm using an additional database to complement the short text with information to reduce the number of cases of incorrect word detection. The usage of Word2Vec is proposed for an automated calculating of the similarity between words by their vectors (these vectors fix the semantic features of the word) [10]. That determined the weight of the edges between the vertices based on the similarity of the vertex words and improved keyword detection compared to TextRank by reducing the number of false positives. It should be noted that the use of the Word2Vec model is common in the task of extracting keywords [11, 12, and 13]. The use of several indicators [14] to determine the weight of the graph edge (frequency of a pair of words and each of the pair) and the selection of candidates for keywords has increased the effectiveness of keyword detection. To identify candidates, the following was taken into account: the distance from the central node, the average weight distribution in the links of one node, the importance of neighboring nodes, the position of the node (especially important for short texts), word frequency.

Thus, we can conclude that the most promising approach to keyword extraction is the construction and analysis of the graph. The accuracy and number of false positives can be improved by using not only statistical but also semantic features of words.

Further in the work the construction of a document template on such features as titles of structural elements, format, keywords is considered.

5. Results and discussion

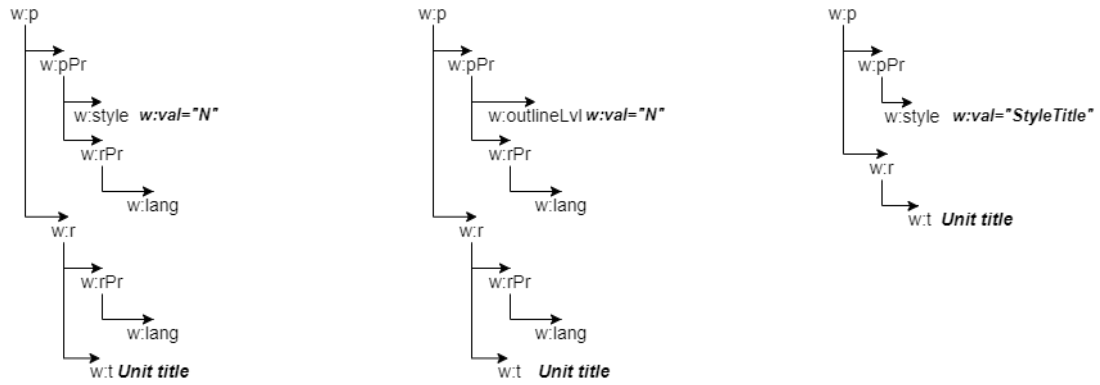
5.1. Research of digital representation of document structure

Defining the structure of the document involves the formation of some tree-like structure that reflects its units - sections and subsections - structural elements. One of the approaches to determining the structure of the document is the analysis of its formatting and separation of structural units based on the titles of sections, subsections.

Possible options for designating section and subsection headings are:

- embedded header styles;
- properties of the paragraph "level" without the use of styles;
- properties of the paragraph "level", which is determined by the style created by the user, and also on the basis of embedded style;
- selection according to the requirements for the preparation of reporting documentation: capital letters, from a new line, in the center, from a new page (with or without a gap), the appropriate numbering, etc.

To determine the most common technique to identify structural elements, the xml structure of doc / docx documents with different techniques of designating structural element titles was analyzed. The general structure is shown in Figure 1. The following tags are used for marking a document: <w:p> represents a paragraph, <w:pPr> sets the paragraph properties, <w:rPr> sets the range properties, <w:style> sets the style attributes, <w:lang> sets a language, <w:outlineLvl> sets a paragraph level, <w:r> represents a range in a paragraph, <w:t> represents the text displayed in the range. Italics indicate the attributes w: val, the value of which allow determining the selection method, given the node to which it belongs. In Figure 1a N is the number of standard header style, $N = \overline{1,9}$. In Figure 1b N is paragraph level, $N = \overline{0,8}$, zero corresponds to the first. In Figure 1c *StyleTitle* is the title of the style created by the user and applied to this paragraph. The *Unit title* is the value of tag w:t and w:t the title of the header of the structural element of the document.



a *b* *c*

Figure 1: The paragraph structure of the sectional title of the using: a – standard heading styles, b – notation of the paragraph level without style, c – notation of the level with style

A more detailed study of the xml-structure showed that the subtrees shown in Figure 1, have a common parent node, regardless of the method of marking of the section and its level. It is <wx:sub-section>. An example is shown in Figure 2. *Section 1, Section 1.1* are titles of section and subsection, *First Section Text, Subsection Text* are the texts they contain. In the example, there are two techniques to denote section and subsection title: by using a custom style with a paragraph-level label and by using a standard heading style. Also in this example, it is shown that only paragraphs with special formatting add the <wx: sub-section> tag to the xml structure. To verify that the appearance of the <sub-section> tag depends solely on the structure of the document, paragraphs were created and formatted using a user style: in *CustomStyle* the paragraph level is one, and in *CustomMainTextStyle* the paragraph level is set as main text (no level). It is confirmed that the appearance of this tag is due solely to the structure.

The use of the identified features is the basis of the process of constructing a structural tree of the document. On the base of this tree a template is formed and updated

5.2. Forming a document template based on xml-parsing

The document template allows you to determine the content of the text of the section submitted for checking to borrowings detection. The template includes a description of the sections. The description of each section contains the title of the section, the sets of alternative titles and keywords, a set of subsections. Each subsection can have a description similar to the section, but only the title is required.

Let's build a model to form a template. This model is a constructor, whose task is to form a document template on a set of structured natural language documents. Let's model the states of a template constructor (Figure 3). From their analysis we can distinguish the following actions of the constructor:

- get an xml-representation of the document;

- pre-processing of the document;
- search for a section in the xml-representation of the document;
- adding a section of the document to the template;
- adding a subsection of the document to the template;
- search for keywords in the text of the document section;
- adding keywords to a section in the template;
- search for a section in the template by name;
- replenish the set of section keywords in the template;
- adding an alternative section title in the template;
- search for a section in the template by keywords;
- manual adding a section to the template.

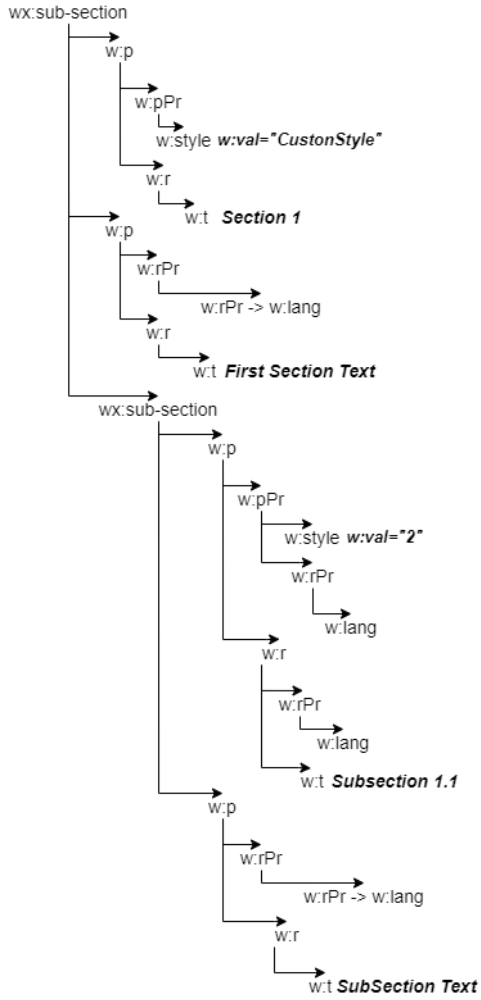


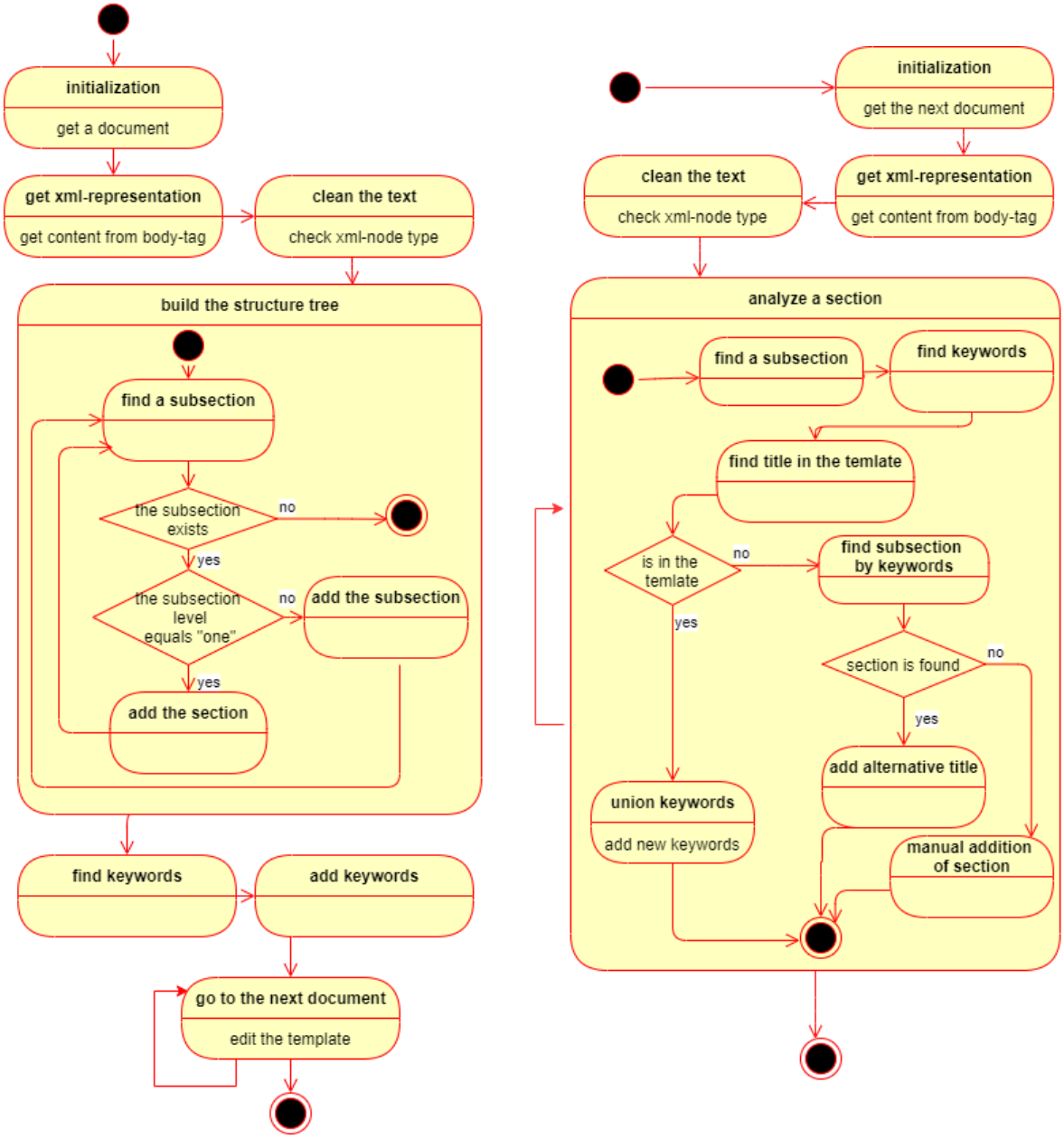
Figure 2: An example of the document structure with different designations of the titles of sections and subsections

To formalize the model, we use the apparatus of constructive-synthesizing modeling. It is based on the constructive properties of formal grammars. The approach is based on the use of a constructor. The constructor is a triple, which includes a carrier, a signature of operations and a set of statements of information constructing support that includes the ontology, the purpose, rules, constraints, initial conditions, and construction completion conditions. To use for a specific subject area the constructor, the clarifying transformations are performed: specialization ($_S\mapsto$), interpretation ($_I\mapsto$), concretization ($_K\mapsto$), implementation ($_R\mapsto$).

To form a document template lets define the constructor, performing its specialization:

$$C = \langle \mathrm{M}, \Sigma, \Lambda \rangle \ {}_S\mapsto C_{TB} = \langle \mathrm{M}_{TB}, \Sigma_{TB}, \Lambda_{TB} \rangle, \quad (1)$$

where $M_{TB} \supset T \cup N$ is the heterogeneous replenishable carrier, T is the terminals set, N is the non-terminals set, Σ_{TB} is the signature of operations and relation for construction, Λ_{TB} is the set of statements of information constructing support.



a b

Figure 3: The states of the constructor in the process of forming the template: a – the general scheme of forming, b – the specified scheme of the state called "go to the next document"

Ontology of constructor C_{TB}. Terminals include digital documents, templates, sections and strings: section titles, alternative section titles, keywords, xml-representation of the document, name of xml-nodes, text of the document section. A template is a sequence of sections.

The element of carrier $_{\overline{w}}m_i \in M_{TB}$ has a set of attributes $\overline{w} = \{w_1, w_2, \dots, w_n\}$. A heterogeneous multiset of elements with attributes is meant by the set. Belonging of the attribute w_j to the element m will be denoted as $w_j \downharpoonleft m$.

The attributes of the section are $w_{section} = \{tit, alt, kw, sub\}$, where tit is a title, alt is a set of alternative titles (it can be empty), kw is set of keywords, sub is set of subsections, each of them is a section with corresponding attribute set; sub can be empty.

Signature $\Sigma_{TB} = \langle \Xi, \Theta, \Phi, \{\rightarrow\} \rangle \cup \Psi$ consists of operations sets and corresponding relationships of the same name, where $\Xi \supset \{\ll, -, \cdot, \wedge, \cup, \triangle, \lhd, \rhd, \bowtie\}$ are the operations of linking and transforming carrier elements, $\Theta = \{\Rightarrow, |\Rightarrow, ||\Rightarrow\}$ are the operations of substitution and inference, Φ are relationships and attribute operations, $\{\rightarrow\}$ is the substitution relation. $\Psi = \{\psi_i : \langle s_i, g_i \rangle\}$ 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 an empty symbol. The inference rules apply the relationship from Ξ, and the corresponding operations apply in the implementation of the constructor.

Let's specify signature operations:

- $\ll (e, d)$ gets data (objects) d from the environment e. It is similar to image reception operation ${}_{ch}\ll (P^*)$, which involves determining (obtaining) certain code of the image m using the channel ch from the part of the external world P^* [15];
- $-(x, a)$ is text pre-processing. It involves parsing the xml-representation of document x and deleting the elements of a given set of nodes a. Such nodes indicate hidden text, cross-references, white color text, and so on;
- $\cdot (\overline{w_1} s_1, \overline{w_2} s_2)$ is concatenation. It connects sections $\overline{w_1} s_1, \overline{w_2} s_2$ so that $\overline{w_1} s_1$ precedes $\overline{w_2} s_2$. The result is a sequence of sections $\overline{w_1} s_1, \overline{w_2} s_2$;
- $\wedge (\overline{w_1} s_1, \overline{w_2} m_2)$ is the inheritance with the specification. It provides a redefinition of the section $\overline{w_1} m_1$ as a result of which it has as attributes $\overline{w_1}$ and $\overline{w_2} m_2$, where $\overline{w_2} m_2$ is a set of strings, which can be the titles or alternate titles, or keywords. It is used to add new attributes to sections and subsections. This operation is similar to the operation of inheriting an image [15], because when forming a section, determining its components, not only the object is formed, but also the corresponding image – a reflection of the object (prototype), its properties and relations on the material carrier;
- $\cup (\{m_i\}, \{m_j\})$ is the union of string sets. The result of it is some set $\{m_i\}$, that contains elements $\{m_i\}, \{m_j\}$ without duplicate;
- $\triangle (x, n, l, v)$ is search of node n in xml-representation of x. The results are a level of nesting l and value of node v. To search for sections (subsections) $n = \text{"wx: sub} - \text{section"}$, $l = \overline{1,9}$, v takes the value of a set of nested tags <w:p> in the found section "wx: sub − section";
- $\lhd (p \downarrow v, kw)$ is search for keywords kw in the text contained in the set of paragraphs p of a given section v;
- $\rhd (t, tit, s)$ is search section s in the template t by the title tit;
- $\bowtie (t, kw, s)$ is search section s in the template t by keywords kw.

The purpose of the constructor is to form constructions-templates and edit them. The purpose of the template is to describe the general structure of a particular class of documents. For example, there will be qualification graduation theses of higher education students, scientific works (articles, abstracts, essays, etc.), educational works (explanatory notes to course projects, reports on laboratory works, practice), technical documentation.

Restrictions on the form templates are imposed by a class of input documents, their quantity; the complexity of the constructed constructions depends on them.

Initial conditions for construct:

- $D = \{d_i\}, i = \overline{1,n}$ is the set of documents of a given class for which the template is form, n is the number of documents on which the template will be form;
- xml is set of tags for markup digital documents;
- "wx:sub-section" is the name of the tag that represents the sections and subsections in the xml-representation of the document;
- σ is non-terminal from which the interfere begins;
- t is an empty template construct that will be filled with sections;
- s is the construct of the section for which the attributes will be defined;
- p is a set of texts contained in the paragraphs of the section;
- p_1 is the text of the first paragraph of the section.

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

To determine the algorithms for performing possible operations and relationships on documents, templates and their components lets interpret the constructor (1) by the algorithmic structure C_A:

$$\langle C_{TB} = \langle \mathrm{M}_{TB}, \Sigma_{TB}, \Lambda_{TB}\rangle, C_A = \langle \mathrm{M}_A, \Sigma_A, \Lambda_A\rangle\rangle \; {}_I\longmapsto \; {}_{I,C_A}C_{TB} = \langle \mathrm{M}_{TB}, \Sigma_{TB}, \Lambda_1, Z\rangle, \quad (2)$$

where $\mathrm{M}_A \supset V_A$, $V_A = \{A_i^0|_{X_i}^{Y_i}\}$ is the set of basic algorithms, X_i, Y_i are the sets of definitions and values of an algorithm $A_i^0|_{X_i}^{Y_i}$, $\Lambda_1 = \Lambda_{TB} \cup \Lambda_A \cup \Lambda_2$, $Z = \{z_i\}$ is a set of possible executors of the model-constructor that can implement algorithms of C_A; $\Lambda_A = \{\mathrm{M}_A = \bigcup_{A_i^0 \in V_A}\left(X\left(A_i^0\right) \cup Y\left(A_i^0\right)\right) \cup \Omega(C_{TB})\}$, where M_A is the heterogeneous replenishable carrier, $\Omega(C_{TB})$ is a set of the construct of the template that satisfies C_{TB}.

Constructor ${}_{I,C_A}C_{TB}$ includes performing operations algorithms:

- $A_1^0|_{A_i,A_j}^{A_i \cdot A_j}$ for algorithms composition, $A_i \cdot A_j$ is sequential execution of the algorithm A_j after algorithm A_i;
- $A_2^0|_b^{A_i}$ for conditional execution: algorithm A_i executes if the expression b is true;
- $A_3|_{e,d}^{d}$ for getting data d from environment e;
- $A_4|_{x,a}^{x}$ for preprocessing text;
- $A_5|_{s_1,s_2}^{s_1 \cdot s_2}$ for concatenation of sections;
- $A_6|_{s_1,m_2}^{s_1}$ for inheritance with specification;
- $A_7|_{m_i,m_j}^{m_i}$ for union of sets;
- $A_8|_{x,n,l,v}^{l,v}$ for search of node n in xml-representation of x;
- $A_9|_{p\downarrow v,kw}^{kw}$ for search for keywords kw in the text of paragraph set $p \downarrow v$;
- $A_{10}|_{t,tit,s}^{s}$ for search section in the template by title, the result (s) is the section, if the search is successful, else $s = null$;
- $A_{11}|_{t,kw,s}^{s}$ for search section in the template by keywords, the result (s) is the section, if the search is successful, else $s = null$;
- $A_{12}|_{l_h,l_q,f_i}^{f_i}$ for substitution;
- $A_{13}|_{f_i,\Psi}^{f_j}$, $A_{14}|_{\sigma,\Psi}^{\overline{\Omega}}$ for partial and complete inference, where f_i, f_j are forms, σ is the axiom, $\overline{\Omega}$ is the set of constructed structures.

Axiomatic of linking the operations and algorithms is as follows: $\Lambda_2 = \{(A_1^0|_{A_i,A_j}^{A_i \cdot A_j} \downarrow)$, $(A_2^0|_b^{A_i} \downarrow:)$, $(A_3|_{e,d}^{d} \downarrow\ll)$, $(A_4|_{x,a}^{x} \downarrow -)$, $(A_5|_{m_i,m_j}^{m_k} \downarrow\cdot)$, $(A_6|_{x,n,l,v}^{l,v} \downarrow\wedge)$, $(A_7|_{p\downarrow v,kw}^{kw} \downarrow\cup)$, $(A_8|_{t,tit,s}^{res} \downarrow\triangle)$, $(A_9|_{p\downarrow v,kw}^{kw} \downarrow\lhd)$, $(A_{10}|_{t,tit,s}^{res} \downarrow\rhd)$, $(A_{11}|_{t,kw,s}^{s} \downarrow\bowtie)$, $(A_{12}|_{l_h,l_q,f_i}^{f_i} \downarrow\Longrightarrow)$, $(A_{13}|_{f_i,\Psi}^{f_j} \downarrow|\Longrightarrow)$, $(A_{14}|_{\sigma,\Psi}^{\overline{\Omega}} \downarrow||\Longrightarrow)\}$.

To clarify the input operations, let us perform concretization of the constructor (2):

$$ {}_{I,C_A}C_{TB} = \langle \mathrm{M}_{TB}, \Sigma_{TB}, \Lambda_1, Z\rangle \; {}_I\longmapsto \; {}_{K,I,C_A}C_{TB} = \langle \mathrm{M}_{TB}, \Sigma_{TB}, \Lambda_3, Z\rangle, \quad (3)$$

where $\Lambda_3 = \Lambda_1 \cup \Lambda_4$, $\Lambda_4 \supset \{\mathrm{M}_{TB} \supset T \cup N, T = \{t, s, tit, kw, nkw, alt, nalt\}\}$, T is the terminal set, t is a template, that is a construction in the form of a set of sections, s is a construction of the section, tit is the title of the section (it is a string), kw, nkw are keyword sets: initial and updated, respectively, $alt, nalt$ – are alternative titles sets: initial and updated, respectively, $N = \{\sigma, \beta, \gamma, \delta, \eta, \mu, \kappa, \rho, \tau, \varphi, \psi\}$ is the set of non-terminals, σ is initial non-terminal.

Let's consider the operations associated with the formation of the template. Rules $s_1 - s_7$ describe the actions corresponding to the states presented in Figure 3a. Rules $s_1 - s_2$ are a preparatory stage, $s_3 - s_7$ correspond to actions in the state "build the structure tree" (Figure 3a). Finding and adding keywords is described by an alternative right side of rule s_7.

$$s_1 = \langle \sigma \rightarrow\ll (D, d_1) \ll (xml, a)\mu\rangle, \quad (4)$$

$$s_2 = \langle \mu \rightarrow -(xml \downarrow d_1, a) \wedge (t, \beta)\varphi\rangle, \quad (5)$$

$$s_3 = \langle \beta \rightarrow\triangle (xml \downarrow d_1, \text{"wx:sub-section"}, l, v)\gamma\rangle, \quad (6)$$

$$s_4 = \langle \gamma \rightarrow\cdot (s, \beta)|s\rangle, \quad (7)$$

$$s_5 = \langle s \ _{l>1}\rightarrow \wedge (s, \gamma)\rangle, \quad (8)$$

$$s_6 = \langle s \ _{l=1}\rightarrow \wedge (s, \delta)\rangle, \quad (9)$$

$$s_7 = \langle \delta \rightarrow \wedge p_1 \downarrow v| \lhd (p \downarrow v, kw)\rangle. \quad (10)$$

Consider the rules that describe the actions of the state "go to the next document" (Figure 3a), which are presented in detail in Figure 3b. Rules $s_8 - s_9$ correspond to the states “initialization”, “get xml-representation” and “clean the text” at the state diagram in Figure 3b.

$$s_8 = \langle \varphi \rightarrow \ll (D, d_i)\psi \mid \ll (D, d_i)\psi\varphi\rangle, \quad (11)$$

$$s_9 = \langle \psi \rightarrow -(xml \downarrow d_i, a)\tau\rangle. \quad (12)$$

Rules $s_{10} - s_{15}$ corresponds to the state “analyze a section” that can repeat if the document that using for update the template has a several sections (Figure 3b)

$$s_{10} = \langle \tau \rightarrow \triangle (xml \downarrow d_i, \text{"wx:sub-section"}, l, v)\eta\tau| \triangle (xml \downarrow d_i, \text{"wx:sub-section"}, l, v)\eta\rangle \quad (13)$$

$$s_{11} = \langle \eta \ _{l=1}\rightarrow \lhd (p \downarrow v, kw_i) \rhd (t, p_1 \downarrow v, s_f)\rho\rangle \quad (14)$$

Rule s_{12} corresponds to the state called “union keywords”.

$$s_{12} = \langle \rho \ _{s_f \neq null}\rightarrow \cup (kw \downarrow s_f, kw_i)\rangle \quad (15)$$

Rule s_{13} allows finding a subsection by keywords.

$$s_{13} = \langle \rho \ _{s_f = null}\rightarrow \bowtie (t, kw_i, s_{ft})\kappa\rangle \quad (16)$$

The next rule allows adding an alternative title for exist section s_{ft}

$$s_{14} = \langle \kappa \ _{s_{ft} \neq null}\rightarrow \cup (alt \downarrow s_{ft}, p_1 \downarrow v)\rangle. \quad (17)$$

Manual additional is made by the next rule

$$s_{15} = \langle \kappa \ _{s_{ft} = null}\rightarrow \wedge (t, s)\rangle. \quad (18)$$

The implementation of the template constructor is to form templates of different types of documents in accordance with their structure, defined by sections and subsections. Forming is performed by executing algorithms associated with signature operations, according to the rules defined by the information support of constructing.

5.3. Software implementation

The model presented in the previous section is the basis for the development of software for automated template formation. The template will be used to improve the performance of the text borrowings detection system for natural language structured digital documents [1].

For software implementation of constructing of a structural tree for the document corresponding to a condition of the same name in Figure 3, an algorithm can be constructed based on paragraph objects and style retrieval methods provided by C# libraries [16, 17] and based on recursive processing of xml-tree called XmlDocument C#, in which the contents of the document in xml-format can be loaded.

Functions have been developed to read the contents of a doc / docx file and build a document tree based on the Composite pattern. They use described methods and are named object and xml. To remove hidden text, the xml-based algorithm required the development of an additional text clearing function. There was the investigation of the time efficiency of these methods for building a tree.

Experimental base: 88 files in docx format with duplicates. 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). The experiment was executed on a PC with the following specifications: Intel (R) Core(TM) i5-9400F CPU, L1 code / L1 data cache / L2 – 6 * 32/6 * 32 / 6*256 KBytes, clock speed / system bus frequency / memory frequency – 2.9 GHz / 800 MHz / 2667 MHz, RAM access time (read / write) 5751/4253 MB / s, operating system – MS Windows 10 Home.

The results were used to calculate comparative estimates for algorithms based on these methods of working with the document:

$$S = \frac{1}{N}\sum_{i=1}^{N}\frac{t(object)_i - t(xml)_i}{\max(t(object)_i, t(xml)_i)} \cdot 100\% \approx -75.5\% \quad (18)$$

$$R = \frac{1}{N}\sum_{i=1}^{N} sign(t(xml)_i - t(object)_i) \cdot 100\% = 0\%, sign(a) = \begin{cases} 1, if\ a > 0 \\ 0, if\ a \leq 0 \end{cases} \quad (19)$$

where S is the average advantage of one algorithm over the second, R is the area of the advantage of the first algorithm over the second, $t(object)_i, t(xml)_i$ are the time to build a document tree using algorithms based on object and xml, respectively. The results showed that xml-algorithm is more efficient in all cases. The average time of formation of the document tree $t(object)$ is 16,4 sec, $t(xml)$ is 3,8 sec.

Software components are currently being developed to search for keywords and edit the template based on information from new documents. The template is an xml file. Its structure is shown in Figure 4. The labels in the figure are the names of the tags.

The constructor described in the sections above builds document templates in the basic configuration (Figure 4a). It takes into account the possible depth of the structural elements of the document and allows for deep analysis of the document. Thus, for comparison, the user can select not only sections but also subsections (sub-items). This is relevant for a class of documents with significant variability of components.

However, if the documents are prepared according to certain strict standards, the structure of the template can be simplified or supplemented. Templates of this type, which were built manually, were used to recognize the structure of diploma projects. Experiments are underway. The experimental base includes graduate theses of bachelors and masters works for specialty Software Engineering. The works consist of two main parts: an explanatory note and appendices. The appendices contain the technical task, working project and scientific publications of the author. The explanatory note has strict regulated sections, which differ significantly from each other. Therefore, there is no need to consider the detailed structure. The author's publications do not have a strict structure, so they are not subject to verification by the template.

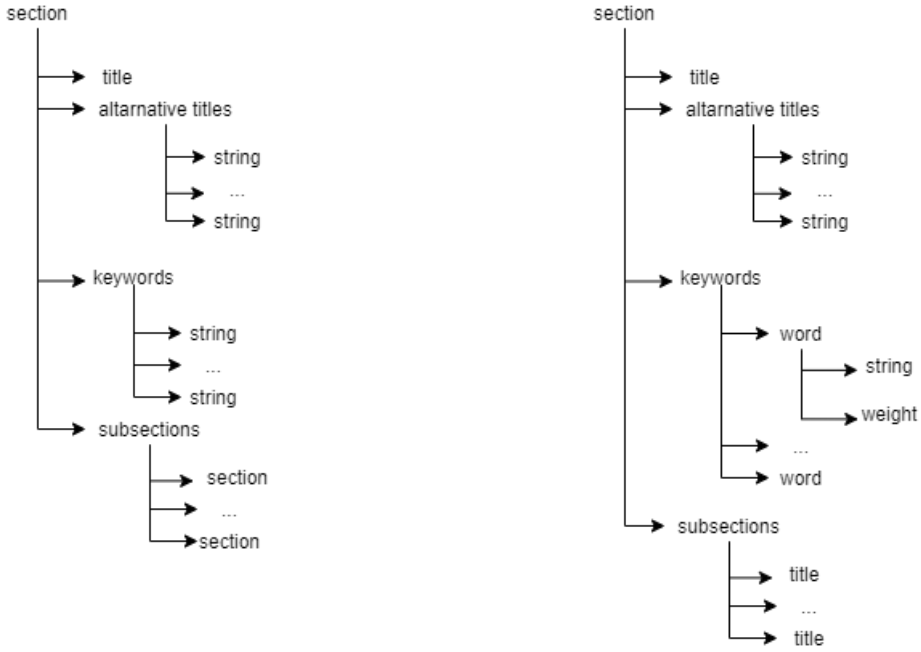


Figure 4: The structure of xml-template for the document: a – the base configuration, b – the experimental configuration

Experimental experience has shown that it is important for the subsection to specify only the title and to save all keywords for the entire section (Figure 4b). Due to significant differences in the content of sections, keywords are an important component of structure identification. Some words are compulsory, such as “search methods”, “scientific novelty”, and so on.

Some words are variable, so it is suggested that words be given weight (Figure 4b). Weight indicates the relevance and importance of the keyword. Therefore, when recognizing a section by structure, the user can set the required weight, which is the sum of the weight of the words. If a section does not have keywords with a given weight, it potentially does not match the template element.

The C# serialization mechanism is used to save the template to a file. For this reason, all information is stored as a tag value, not its attribute.

The task now is to develop an algorithm for calculating the weight of keywords and determining the percentage and total weight of words that should be in the section so that it is recognized as an element of the template.

5.4. Summary and Future Works

The document structure definition approach which proposed by the authors previously [1], requires a human expert to forming the template. Therefore, this article presents the model of the template constructing process in automatic mode.

In this paper:

1. the features of the document are defined for identifying document structure;
2. the importance of taking into account keywords in identifying the document structure has been established. The completed review of related works showed the prospects of using graph approaches for extracting keywords with a preliminary analysis of the text on the basis of TF * IDF indicators;
3. the xml-structure of documents was investigated, which made it possible to identify universal features for the identification of structural elements with different formatting;
4. modeling of the process of forming a document template by means of UML was done, that allowed determining the basic operations and algorithms of the process model;
5. the model of formation the document structure template has been developed by means of constructive-synthesizing modeling. This model encapsulates data and methods in a single formal object called constructor. It is used to automate template forming;
6. some components of the developed model are implemented in software. Computer experiments have been carried out to investigate the time effectiveness of algorithms for constructing a structural tree of a document. The average advantage of the XMLDocument-based algorithm over the Microsoft.Office.Interop.Word object-based algorithm is about 75.5% and allows reducing the average time from 16.4 seconds to 3.8 seconds.

Various configurations of the xml-template are currently being researched and discussed to improve the efficiency of its use for working with documents in the academic environment.

Further areas of work are:

- implementation and testing of keyword extraction algorithms;
- full software implementation of the developed model;
- investigation of stemmatization for working with headings of structural sections and extraction of keywords in order to improve the accuracy of the model as a whole;
- research the possibility of using existing software implementations of stemmatization for the Ukrainian language by library functions with bigger accuracy non C#.

6. Conclusion

Taking into account the structure of the document in text borrowing detection avoids false positives associated with the use of reference information and links. The proposed approach based on the template allowed performing the recognition of document parts by content in order to further select sections for check and better selection of materials for comparison.

The creation of the model that is the constructor allowed to define and formalize all operations and the data necessary for automation of construction of a template. This model can have several implementations, depending on the algorithms interpretation of the described operations, which makes it universal.

The automation of template construction allows using the template approach for comparison of the documents of different types: qualification works, technical documentation, etc. The developed automated tool for forming templates is planned to be used to check the educational and qualification works of higher education students.

7. References

[1] O. Kuropiatnyk, V. Shynkarenko, Text Borrowings Detection System for Natural Language Structured Digital Documents, Proceedings of the 4th International Conference on Computational Linguistics and Intelligent Systems, COLINS 2020, Lviv, Ukraine, 23–24 April 2020. pp. 294–305.

[2] V. Shynkarenko, O. Kuropiatnyk, Constructive-synthesizing model of text graph representation, Proceedings of the 10th International Conference of Programming (UkrPROG'2016), Kyiv, Ukraine, May 24-25, 2016. vol. 1631, pp. 63 – 72.

[3] A. S. Vanyushkin, L. A. Grashchenko, Methods and algorithms for extracting key words. New information technologies in automated systems 19 (2016).

[4] T. V. Golub, M. Yu. Tyagunova, Method of steaming Ukrainian-language texts for classification of documents based on Porter's algorithm. Scientific works of Donetsk National Technical University. Series: Informatics, cybernetics and computer engineering 1 (2017 59–63.

[5] A. Hlybovets, V. Tochytsky, Algorithm of tokenization and steming for texts in Ukrainian (2017)

[6] E. V. Sokolova, O. A. Mitrofanova, Automatic extraction of keywords and phrases from Russian-language texts using the KEA algorithm. Computational linguistics and computational ontologies 1 (2017) 157–165.

[7] E. G. Grigorieva et al. Key word extraction algorithm based on the graph model of the linguistic corpus. Bulletin of the Volgograd State University. Series 2: Linguistics 16, No. 2. (2017).

[8] D. Suleiman, A. A. Awajan, W. Al Etaiwi, Arabic Text Keywords Extraction using Word2vec in: Proceedings of the 2nd International Conference on new Trends in Computing Sciences (ICTCS), 2019, pp. 1-7.

[9] W. Li, J. Zhao, TextRank algorithm by exploiting Wikipedia for short text keywords extraction in: Proceedings of the 3rd International Conference on Information Science and Control Engineering (ICISCE), 2016, pp. 683-686.

[10] Y. Wen, H. Yuan, P. Zhang, Research on keyword extraction based on word2vec weighted textrank in: Proceedings of the 2nd IEEE International Conference on Computer and Communications (ICCC), 2016, pp. 2109-2113.

[11] D. Suleiman, A. A. Awajan, W. Al Etaiwi, Arabic Text Keywords Extraction using Word2vec in: Proceedings of the 2nd International Conference on new Trends in Computing Sciences (ICTCS), 2019, pp. 1-7.

[12] H. Benghuzzi, M. M. Elsheh, An Investigation of Keywords Extraction from Textual Documents using Word2Vec and Decision Tree. International Journal of Computer Science and Information Security (IJCSIS) 18, No 5 (2020).

[13] S. Song et al. A Novel Text Classification Approach Based on Word2vec and TextRank Keyword Extraction in: Proceedings of IEEE Fourth International Conference on Data Science in Cyberspace (DSC), 2019, pp. 536-543.

[14] S. K. Biswas, M. Bordoloi, J. Shreya, A graph based keyword extraction model using collective node weight. Expert Systems with Applications 97 (2018) 51–59.

[15] V. Shynkarenko, O. Kuropiatnyk, Constructive Model of the Natural Language. Acta Cybernetica 23, no 4, (2018) 995–1015. doi: 10.14232/actacyb.23.4.2018.2.

[16] E. T. Sakhno, E. V. Romashka, Comparative analysis of existing libraries for creating and processing documents with C# .NET means. Bulletin of Lugansk National University named after Volodymyr Dahl 1-2 (2017) 80–82.

[17] A. N. Vildanov, From the experience of automating Word in C# on the example of creating a table of contents. Cloud of science 5, No 1. (2018).