Method for Measuring the Semantic-similarity of Textual Document and Web-pages

Jasurbek Atajanov, Boburbek Atadjanov, Shakhboz Abdulazizov, Orif Makhmano

Abstrac;t ;In this article, the process of semantic comparison of web pages in the global network is addressed. Comparing process is done based on set of words in text. Google Ajax Search and Yandex-API services are used to search and find web pages in global network. The local database is used for the determination of synonyms of the words. For the experiment, we took a document from Global network and its text was exchanged to synonyms and generated new document. The generated document compared with the given algorithm and it showed 90-95% similarity. The developed system with this algorithm used in Tashkent University of Information Technologies named after Muhammad Al-Khwarizmi for verifying students Graduation Qualification works for plagiarism.

Keywords: stemming text, search information from document, semantic analyze documents.

I. INTRODUCTION.

Developing internet is connected with enlarging the mass of information and new one is being loaded in each minute. It is so difficult copyright protection; however, it is impossible [1]. Plagiarism means to perform a thing as own without showing the source [2, 3]. On base a text document, the task of a plagiarism detection system is to find if the document is copied partially or fully from other documents from the Web or any other repository of documents. It has been observed that plagiarists use different means to hide plagiarism so that a plagiarism detection system cannot catch plagiarism cases [4, 5]. Nowadays, there are a number of algorithms, models, and systems which can detect plagiarism between documents.

II. RELATED WORKS

In work, Vera Danilova (2013) showed methods of cross-language plagiarism detection between local digital documents. There are described process of comparing documents which are written in different natural languages [6]. In paper, Zaid Alaa, Sabrina Tiun, and Mohammedhasan Abdulameer (2016) it was described the method of cross-language method documents in Arabic and English. Also, it is showed compare documents accounting with synonym forms of words. But in this work also are **addressed local documents** [7].

In an interesting paper [8], Daniele Anzelmi and colleagues report the SCAM (StandardCopy Analysis Mechanism) algorithm which is a relative measure to detect overlap by making comparison on a set of words that are common between test document and registered document. This plagiarism detection system, like many Information Retrieval systems, is evaluated with metrics of precision and recall. To compare documents taking into account the synonym of the form of words, this algorithm suggests checking each synonym of forms. In this case, total amount operation of will be calculated on base the following formula

$$s = \sum_{i=1}^{l} c_i \qquad (1)$$

Here, l - count of words in document, c_i - count of synonym forms i - word, s - total amount operation. The total amount of the comparison operation will be even greater if we use algorithms of the class shilling [9].

$$s = \prod_{i=1}^{l} c_i \qquad (2)$$

In paper [10] introduced a cross-language plagiarism system for English-translated copies of Spanish document's detection. Their system was comprised of three stages; namely translation detection, internet search and report generation.

Today every internet user is familiar with web search engines like Google Web Search, Yahoo, Yandex, and etc. The searching algorithm of these engines are based on partial text search, which means search can not be done using a whole document as search item. There are a number of systems, which can detect document plagiarism by using web search engines, like AntiPlagiarism.NET, Advego Plagiatus, Unplag. If new document uses synonym of the text from existing source, the aforementioned tools do not detect the plagiarism [1, 2]. In addition, there is a number of difficulties during comparing documents which is located on the Internet than comparing documents on a local database. If the document exists in the inner database we can prepare (normalization, stemming, indexing, and etc) before start the compare process. As result, it can improve the process of compare. We can not do these steps if document on the internet.

Revised Manuscript Received on December 30, 2019.
* Correspondence Author

Jasur Atadjanov*, Software Developer at Uztelecom, Tashkent University of Information Technologies (TUIT), Tashkent,Uzbekistan.

Bobur Atadjanov, Software Developer at Uztelecom, Tashkent University of Information Technologies (TUIT), Tashkent, Uzbekistan.

Shakhboz Abdulazizov Software Developer at Uztelecom, Stock Company, Bachelor of Science in Computer Science and Software Engineering from Inha University in Tashkent (IUT).

Orif Makhmanov, Deputy Director at the Center of Introduction and Development of Information and Communication Technologies, Tashkent, Uzbekistan.

III. DETECTION PROCESS

This work describes an algorithm that detects plagiarism of a given document throughout Internet resources (Web-pages), **taking the synonymity of the words into account**.

The following figure illustrates the plagiarism detection process of the algorithm.

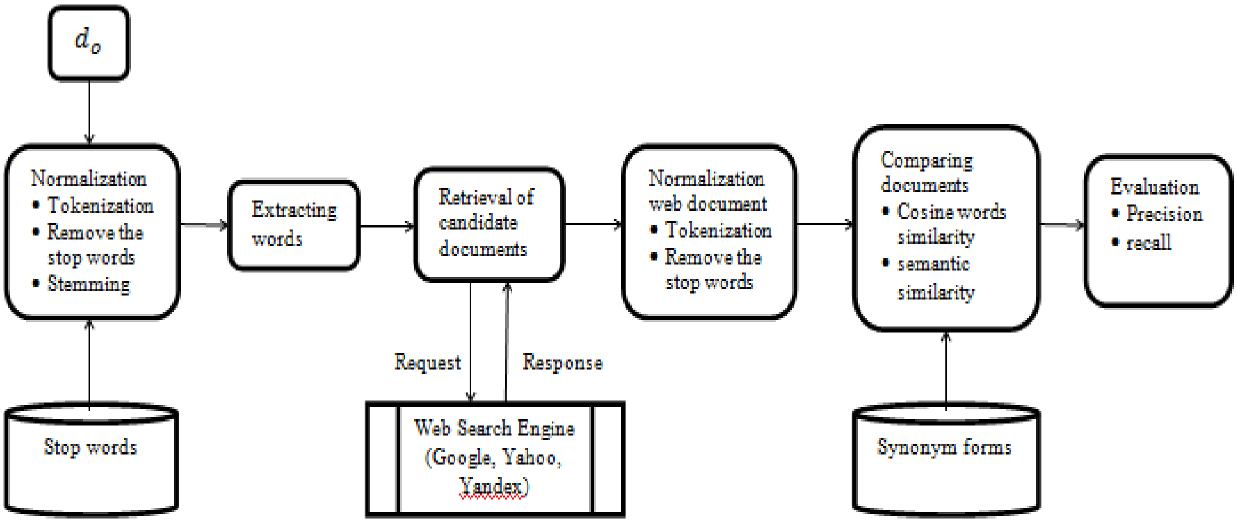

Fig. 1. Plagiarism detection process.

A. Document normalization

The document normalization phase consists of (1) content analyzing, (2) tokenization, (3) stop word removal steps. The main aim of this phase is to prepare the original document's dataset for similarity comparisons with other texts.

Content analysis is a research tool used to determine the presence of certain words, themes, or concepts within some given qualitative data (i.e. text). Using content analysis, researchers can quantify and analyze the presence, meanings and relationships of such certain words, themes, or concepts. Researchers can then make inferences about the messages within the texts, the writer(s), the audience, and even the culture and time of surrounding the text. In this step we can use Apache Tika toolkit. The Apache Tika™ toolkit detects and extracts metadata and text from over a thousand different file types (such as PPT, XLS, and PDF). All of these file types can be parsed through a single interface, making Tika useful for search engine indexing, content analysis, translation, and much more [11]. After this step document which in any file formats will be converted into plain text format, in Table-1 it is given the result of this step.

Table-I. Result of content analyzing step

Source text in HTML format (Input)	Plain text (Output)
<p>In mathematics and computer science, <b>an algorithm</b> is a finite sequence of well-defined, <a href="/">computer-implementable instructions</a>, typically <a>to solve</a> a class of problems or to perform a computation. <b><i>Algorithms</b></i> are unambiguous specifications for performing calculation, data processing, automated reasoning, and other tasks.</p>	In mathematics and computer science, an algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation. Algorithms are unambiguous specifications for performing calculation, data processing, automated reasoning, and other tasks.

Tokenization is the process of converting text into elements (words, phrases, symbols, or other meaningful elements) called tokens, and condition process of documents will be based on a set of these tokens. There are a number of algorithms for document tokenization. In this algorithm, we used tokenization on-base Regular Expressions (sometimes called a rational expression) [12]. Usually such patterns are used by string searching algorithms for "find" or "find and replace" operations on strings, or for input validation. It is a technique developed in theoretical computer science and formal language theory. There are given regular expressions that parse text into a collection of words.

~ [A - Z] . * [. , : ! ? ;] (? = \s|$) ~ s (3)

Table-2 shows how the tokenization has been done each word and component, including the stop words and special characters.

Table-II. Result of tokenization

Plain text (Input)	Words collection (Output)
In mathematics and computer science, an algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation. Algorithms are unambiguous specifications for performing calculation, data processing, automated reasoning, and other tasks.	"in", "mathematics", "and", "computer", "science", "an", "algorithm", "is", "a", "finite", "sequence", "of", "well", "defined", "computer", "implementable", "instructions", "typically", "to", "solve", "a", "class", "of", "problems", "or", "to", "perform", "a", "computation", "algorithms", "are ", "unambiguous", "specifications", "for", "performing", "calculation", "data", "processing", "automated", "reasoning", "and", "other", "tasks"

It is known, that every natural language has **stop words**, which used inside of sentence to relate words to each other.

There is no single universal list of stop words used by all-natural language processing tools; and indeed, not all tools even use such a list. The next step consists of removing stop words from collection words. The list of the English stop words that has been used in this study is a default English stop words list, and is a well-known list used by many researchers, including [14]. In Table-3 it is displayed text after removing stop words step.

Table-III. Removing stop words

Words collection (Input)	Words collection (Output)
“in”, “mathematics”, “and”, “computer”, “science”, “an”, “algorithm”, “is”, “a”, “finite”, “sequence”, “of”, “well”, “defined”, “computer”, “implementable”, “instructions”, “typically”, “to”, “solve”, “a”, “class”, “of”, “problems”, “or”, “to”, “perform”, “a”, “computation”, “algorithms”, “are ”, “unambiguous”, “specifications”, “for”, “performing”, “calculation”, “data”, “processing”, “automated”, “reasoning”, “and”, “other”, “tasks”	“mathematics”, “computer”, “science”, “algorithm”, “finite”, “sequence”, “well”, “defined”, “computer”, “implementable”, “instructions”, “typically”, “solve”, “class”, “problems”, “perform”, “computation”, “algorithms”, “unambiguous”, “specifications”, “performing”, “calculation”, “data”, “processing”, “automated”, “reasoning”, “tasks”

In linguistic morphology and information retrieval, **stemming** is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. The stem need not be identical to the morphological root of the word; it is usually sufficient that related words map to the same stem, even if this stem is not in itself a valid root [15]. There are a number of algorithms for stemming words in natural languages. There are many algorithms for stemming words in natural languages like Snowball, a framework for writing stemming algorithms, and implemented an improved English stemmer together with stemmers for several other languages [16, 17]. During stemming process we can use algorithms from Snowball Framework. Result of the stemming process for our words we can see in Table-4.

Table-IV. Result of stemming step

Words collection (Input)	Stemmed words collection (Output)
“mathematics”, “computer”, “science”, “algorithm”, “finite”, “sequence”, “well”, “defined”, “computer”, “implementable”, “instructions”, “typically”, “solve”, “class”, “problems”, “perform”, “computation”, “algorithms”, “unambiguous”, “specifications”, “performing”, “calculation”, “data”, “processing”, “automated”, “reasoning”, “tasks”	“math”, “compute”, “science”, “algorithm”, “finite”, “sequence”, “well”, “define”, “compute”, “implement”, “instruct”, “typical”, “solve”, “class”, “problem”, “perform”, “compute”, “algorithm”, “ambiguous”, “specific”, “perform”, “calculate”, “data”, “process”, “automat”, “reason”, “task”

B. Extracting words

In this phase, we convert stemmed words into formal form. In this form, every element of the document will be consists of stemmed word and term frequency words in the document.

$$D_j = \left(\left(d_1, n_1\right), \left(d_2, n_2\right), .., \left(d_p, n_p\right)\right) \qquad (4)$$

$$\forall d_i \notin H_j \qquad (5)$$

Here, j - the natural language of D document, H_j - the collection of stop words j natural language, d_i - the term on the text, n_i - the number of occurrences d_i term in the text. In Table 5 it was shown the structure documents in (4) version. This table also described synonym forms of every word.

Table-V. Term Frequency of the document's words

Term	Term Frequency	Synonym forms	Count of synonym forms
compute	3	PC, laptop, electronic brain	3
algorithm	2	algorism, logic	2
math	1	-	0
science	1	knowledge, ology, study	3
data	1	information, evidence, facts, material	3

3.4. Target form of term synonyms

The reason for displaying the term`s synonym forms above related to explain to the reader explain compare process terms accounting with its synonym forms. Before compare terms, this algorithm detects the target synonym form every term. In this case, we use the following data structure.

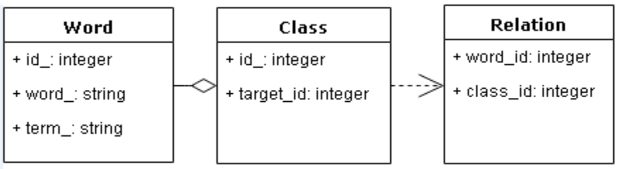

Fig. 2. Target form of terms detection data structure.

Here, word – table to store a list of words and its terms, class – table to store word category and target word pointer, relation – table to storing word and its category relation. After retrieving a set of terms, we get target versions of every term on base this data structure.

According to this operation, every term will be converted into its target word. As a result, on-base one compare operation we can check all existing forms of term synonyms. In this case, total compare operation count will be equal **count of terms**. This is the main difference in this algorithm than existing algorithms like SCAM and shilling. The total count of compare operation showed for the SCAM algorithm in (1) and in (2) showed shilling class algorithms.

C. Retrieval of candidate documents

In this section, we will describe how to find Web pages by collection of terms. As noted above to find web pages we will use search engine systems like Yahoo, Google Ajax, Yandex-XML, etc.

These systems have own API working with them. At first, we will divide *D* document on the (4) form into some parts *k* sequence of term which consists of ($k \geq 1$) terms. As a result, we will have the following collection:

$$R = [[d_1, d_2, ..., d_k], [d_{k+1}, d_{k+2}, ..., d_{k+k}], ..] \qquad (6)$$

We will identify the list of Web-pages which are similar with each part. This process has been illustrated as an example of Google service.

GET REQUEST
http://google.com/search?q=d_1*+* d_2*+..*d_k (7)
GET REQUEST
http://google.com/search?q=d_{k+1}*+* d_{k+2}*+..*d_{k+k}

On the base of search results, we can build (8) object which consists set of l_j URL address of Web-pages and p_j number of their occurrences.

$$L = ((l_1, p_1), (l_2, p_2), ..., (l_s, p_s)) \qquad (8)$$

Next step, the elements of L object will be placed in descending sorting by the value of the p_j parameter.

$$L' = ((l'_1, p'_1), (l'_2, p'_2), ..., (l'_s, p'_s)) \qquad (9)$$

$$\forall p'_j \in [p_1, p_2, .., p_s], \ \forall l'_i \in [l_1, l_2, .., l_s],$$

$$\forall p'_{i-1} \leq p'_i$$

The list of Web-pages which are similarity with the *D* document we can get on base (9) formula. At the next step, we get content of Web-page from the Internet on base their URL addresses.

$$L_t = [T_1, T_2, .., T_s] \qquad (10)$$

Here T_i - content of i Web-page, L_t - the collection of Web-pages content. It is recommended to save up these contents on the file system during the programming via this algorithm.

D. Normalization web document

As it is known, Web-page is the text on HTML format; before starting the compare process we will repeat **Document Normalization, Extracting word** and **Target form of term synonyms** phases for the content of every Web page. As a result, each T_i document can be transformed in the following form.

$$T = \left(\left(t_1, m_1\right), \left(t_2, m_2\right), .., \left(t_g, m_g\right)\right)$$

(11)

$$\forall t_i \notin H_j \qquad (12)$$

Here j - the natural language of T document; H_j - the list of stop words which belongs *j* natural language; t_i - the words in the text, m_i - the number of occurrences t_i word in the text.

E. Comparing documents

Detecting plagiarism is not a simple string match. Our algorithm detecting similarity two documents based on the set of terms both documents. In other words similarity of both documents calculates on-base similarity (3) and (11) objects. At first, we will calculate the weight of both documents.

$$N = \sum_{i=1}^{p} n_i \qquad (9)$$

$$M = \sum_{i=1}^{s} m_i \qquad (10)$$

Here, N - the weight of the D document, M - the weight of document T. Next step, we will get the list of words that exists in both D and T documents by intersection set of their words.

$$[d_1, d_2, ..., d_p] \cap [t_1, t_2, ..., t_s] = [x_1, x_2, .., x_k]$$

(11)

$$X = ((x_1, n_1, m_1), (x_2, n_2, m_2), ..., (x_k, n_k, m_k))$$

(12)

Here x_i - the term in the *D* and *T* documents, n_i - the number of recurrence x_i term in the D document, m_i - the number of occurrence x_i term in the *T* document. The similarity degree of *D* document to *T* document will be calculated as the following formula.

$$dt = \sum_{i=1}^{k} \frac{n_i \cdot m_i}{N^2} \qquad (13)$$

In line the similarity degree of *T* document to *D* document will be calculated as the following formula.

$$td = \sum_{i=1}^{k} \frac{n_i \cdot m_i}{M^2} \qquad (14)$$

On the base (13) and (14) we can calculate the total similarity degree of both documents. It will be calculated on-base (17) formula.

$$sim(D, T) = \max(dt, td) \qquad (15)$$

Based on the execution of these actions for documents in all (8), we will have the collection of documents that are similar to *D* document.

In this section, we describe the algorithm which found plagiarism documents on the Internet resources taking into account the synonym version of the terms.

This algorithm can help improve plagiarism detection systems during check from Internet resources.

IV. EXPERIMENTAL RESULTS

The proposed model of this study was programmed with Java programming language. The objective of the proposed model is to detect plagiarism Internet web-pages on-base document according to synonym forms terms. During the test process, we get a file from the internet (**we marked it file1.html**) and replaced its some words with synonym forms, and marked it **file2.html**. Next, some paragraphs' place changed in file1.html and created new **file3.html** file. After it, some terms from file.html were be replaced its paraphrase terms, implemented some homonyms words and saved this file with **file4.html** name.

These three files (file2html, file3.html, and file4.html) were given to experts and asked them to rate plagiarism degree between source file (file1.html).

After receiving answers from experts, we took the arithmetic mean values according to their results (fiel2.html – 84%, file3.html – 73%, file4.html – 50%). also, these files were checked by AntiPlagiarism.NET and Unplag systems. As we said before these systems detect plagiarism on-base Internet resources. In the Figure-3 it was given experiment result with diagram version.

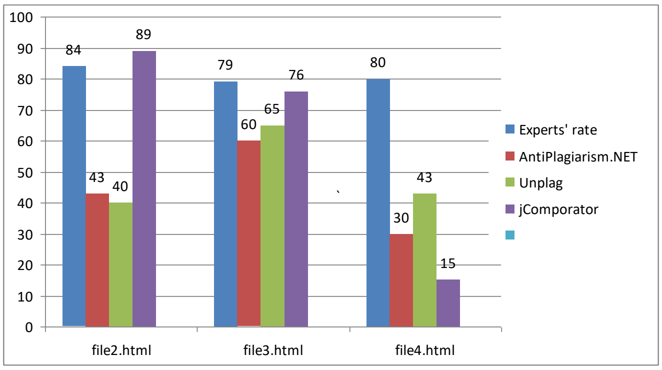

Fig. 2. Experiment result

The result of the experiment showed that in file4html we get the really bad results because there were used homonyms words and our algorithm did not give the right result. But the rest of the files its results are closer to the result of experts.

V. CONCLUSION

In conclusion, both algorithms are used to take the list of Web-pages which are similar to the text. The main drawback of the following algorithms is Web-pages search information only pages which are generated from GET request. Additionally, the resulting array value depends on the results of searching information systems on the internet. It is known that there are CEO engineering workpieces on the internet that allow Web-pages to be viewed on the foreground during the searching information from the internet. It will impact the process of searching for similar information to the text. This algorithm implemented into jComporator information system used to detect document plagiarism and tested on Tashkent University of Information Technologies named after Muhammad Al-Khwarizmi in 2013-2014 years. In this process, systems helped to detect a number of plagiarism documents.

REFERENCES

1. Bretag, T., & Mahmud, S. 2009. A model for determining student plagiarism: Electronic detection and academic judgement. Journal of University Teaching & Learning Practice, 6(1). Retrieved from http://ro.uow.edu.au/jutlp/vol6/iss1/6
2. Macdonald, R., & Carroll, J. 2006. Plagiarism—a complex issue requiring a holistic institutional approach. Assessment & Evaluation in Higher Education, 31(2), 233–245. doi:10.1080/02602930500262536
3. Lancaster, Thomas. 2003. Effective and Efficient Plagiarism Detection (PhD Thesis), School of Computing, Information Systems and Mathematics South Bank University
4. Roy, Chanchal Kumar;Cordy, James R. 2007. "A Survey on Software Clone Detection Research". School of Computing, Queen's University, Canada.
5. Salha M. Alzahrani, Naomie Salim, and Ajith Abraham, Senior Member. 2012. Understanding Plagiarism Linguistic Patterns, Textual Features, and Detection Methods. IEEE Transactions on systems, man, and cybernetics—part c: Applications and Reviews, Vol. 42, no. 2, March 2012
6. Vera Danilova (2013), Cross-Language Plagiarism Detection Methods. Proceedings of the Student Research Workshop associated with RANLP 2013, pages 51–57, Hissar, Bulgaria, 9-11 September 2013.
7. Zaid Alaa, Sabrina Tiun, and Mohammedhasan Abdulameer (2016) Cross-language plagiarism of arabic-english documents using linear logistic regression. Journal of Theoretical and Applied Information Technology 10 th January 2016. Vol.83. No.1
8. Daniele Anzelmi, Domenico Carlone, Fabio Rizzello,Robert Thomsen, D. M. Akbar Hussain (2011) Plagiarism Detection Based on SCAM Algorithm // Proceedings of the International MultiConference of Engineers and Computer Scientists 2011 Vol I? IMECS 2011, March 16-18, 2011 Hong Kong
9. Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich (7 July 2008). "w-shingling". Introduction to Information Retrieval. Cambridge University Press. ISBN 978-1-139-47210-4.
10. Anguita, A., Beghelli, A., &Creixell, W. 2011. Automatic Plagiarism Detection. In NLPKE 2011,7th International Conference on Natural Language Processing and Knowledge Engineering.(2011).PP.1-10.
11. Chris A. Mattmann and Jukka L. Zitting. 2011. Tika In Action. ISBN 9781935182856.- 256 p. 2011 y.
12. Ruslan Mitkov. 2003. The Oxford Handbook of Computational Linguistics. Oxford University Press. p. 754. ISBN 978-0-19-927634-9.
13. Erik H., Otis G., Michael McC. Lucene in Action – Covers Apache Lucene v.3.0// Manning Publications.-486p.,2009 y.
14. Tong, Simon, Uri Lerner, Amit Singhal, Paul Haahr, and Steven Baker. 2012. "Locating meaningful stopwords or stop-phrases in keyword-based retrieval systems." U.S. Patent 8, issued July 3, 2012. PP.214-385.
15. Lovins, Julie Beth. 1968. "Development of a Stemming Algorithm" (PDF). Mechanical Translation and Computational Linguistics. 11: 22–31.
16. Popovič Mirko, Willett Peter. 1992. The Effectiveness of Stemming for Natural-Language Access to Slovene Textual Data, Journal of the American Society for Information Science, Volume 43, Issue 5 (June), pp. 384–390
17. Erik H., Otis G., Michael McC. Lucene in Action – Covers Apache Lucene v.3.0// Manning Publications.-486p.,2009 y.

AUTHORS PROFILE

Jasur Atadjanov is a software developer at Uztelecom. He has more than 17 years of experience in software development. He earned a Bachelor of Science at Urganch State University (USU) and a Master's Degree at Tashkent University of Information Technologies (TUIT). He also gained a Ph.D. degree at TUIT through thesis on 'Automated Library System' based on which 15 academic articles were published.

Besides, he created 'ARMAT' software based on the research work and implemented it in more than 60 higher education institutions. Moreover, he is head of the Billing System Development department at Uztelecom. In Uztelecon he developed a billing system for the telecommunication area. Nowadays it is used for all regional branches of the company for more than 1.5 mln clients. Nowadays, he is working on creating a multilanguage plagiarism detection system and has published the article "Cross-Language Plagiarism Detection Based on CLAD Method" concerning the outcome achievements.

Bobur Atadjanov is a software developer at Uztelecom. He has more than 10 years of experience in software development. He earned a Bachelor of Science at Tashkent University of Information Technologies (TUIT) through a thesis on 'Cryptography by DIA Matrixes' and a Master's Degree at the same university through thesis on 'Software Development by Grid Computing'.

His honorable positions in Republican Subject Olympiads can be attributed to his academic skills. As for work experience, he participated in several corporate projects including Armat.uz (Corporate electron library), LarStat(universal parser for CDR files), v3.esud.uz(an online platform for the judicial system) and the like. His research interest is the Cross-Platform plagiarism system.

Shakhboz Abdulazizov is a software developer at Uztelecom Stock Company with 2 years of experience. He has a Bachelor of Science in Computer Science and Software Engineering from Inha University in Tashkent (IUT) and has experience with a wide range of programming tools and concepts. At the end of his undergraduate, he achieved Academic Excellence Award for outstanding academic performance and topped in his department. Besides, as an individual work in course projects, he created a wide range of applications ranging from hardware programming in assembly ("Smart Room", IoT in room-scale) to high-level java software ("IUT-Cafe", an internal platform for online food order). Concerning his after-graduation work experience, he worked on Uztelecom Billing System under the supervision of Jasur Atadjanov to make the system more robust and multifunctional. Moreover, he created the "Electron Information Directory Service" (EIDS) which is an online platform to find information about organizations and services all around the country. He is currently working on the detection of plagiarism all-around the web considering synonyms regardless of languages and shows a strong interest in real-time applications of machine learning.

Orif Makhmanov is a deputy director at the Center of Introduction and Development of Information and Communication Technologies. He has more than 10 years of experience in software development. He earned a Bachelor of Science at Tashkent University of Information Technologies (TUIT) and a Master's Degree at the same university. He also gained a PhD degree at TUIT through thesis on 'Development of methods and models for monitoring the scientific potential of higher educational and research institutions' based on which 34 academic articles were published. Besides, he created 'UNISYS' software based on the research work and implemented it in more than 15 higher education institutions. And, he created 'Scientific potential' software based on the research work and implemented it in more than 70 higher educational institutions and scientific research institutions. Moreover, he is the deputy director of the Center of Introduction and Development of Information and Communication Technologies. In the Center, he developed a 'Scientific potential' software system for the monitoring of scientific potential of higher educational and research institutions. His research interest is the Cross-Platform plagiarism system and machine learning