# Document Plagiarism Detection On The Internet With Accounting Synonym Forms Of Words

**Jasurbek Atadjanov**

**Abstract:** This article provides an algorithm for checking the Internet for similarity with the full text of the various types of content. There are also some ways to convert full-text text in different formats. The information search process is based on Internet search engines like Google, Yahoo, and Yandex.XML. Based on this algorithm, it is possible to develop a system that checks the synonyms of words on the Internet.

**Index Terms:** stemming text, search information from document, semantic analyzing documents, compare documents, plagiarism detection, and synonym.

## 1 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]. There are a lot of services, algorithms and software for detecting plagiarism by machine [4], for example, «Антиплагиат», Advego Plagiatus, Unplag, miratools.ru, istio.com, Praide Unique Content Analyser II, Plagiatinform. For now, there is not possible to detect plagiarism on-base Internet resources existing systems if used words synonym. In this paper, we will describe a new algorithm that can detect plagiarism by Web pages if it was used word synonyms. During find Web pages on-base keywords it was used existing search engine systems like Google Web Search, Yahoo, and Yandex.XML [5], [6].

## 2 Related Works

Work by Vera Danilova (2013) showed methods of cross-language plagiarism detection between local digital documents. It described process of comparing documents which are written in different natural languages [7]. In the paper by Zaid Alaa, Sabrina Tiun, and Mohammedhasan Abdulameer (2016) the method of cross-language method documents in Arabic and English was described. The paper also showed comparison of documents considering synonymity of words. But, the drawback only local documents were addressed [8].

In an interesting paper [9], Daniele Anzelmi and colleagues report the SCAM (Standard Copy 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 other Information Retrieval systems, is evaluated with metrics of precision and recall. Today every internet user is familiar with web search engines like Google Web Search, Yahoo, Yandex.XML, and etc. The searching algorithms of these engines are based on partial text search, which means search cannot 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 greater 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 local database we can prepare (normalization, stemming, indexing, and etc.) before start the compare process. As result, it can improve the process of compare. We cannot do these steps if document on the internet.

• *Jasurbek Atadjanov is head of Software development department at Information Systems branch of Uztelecom Stock Company, Uzbekistan, PH-998998702297. E-mail: j.atadjanov@gmail.com*

## 3 Detection Steps

In this section, we report a detailed overview of the main methods implemented and the steps followed to achieve a detection algorithm. This process consists of the following steps:

- Convert document into simple text format – in this section, we can use open-source system Apache Tika. On base this system we can convert PDF, HTML, MsWord files into simple text [10];
- Parsing text – we generate word Vector on-base text and exclude from this collection stop words. In computing, stop words are words which are filtered out before processing of natural language data (text) [11].
- By using Yahoo, Google, Yandex.XML services we can get the list of web pages on-base collection of words [5], [6], [12].
- Compare documents and sort web pages list.

### 3.1 Convert document into simple text format

As we discussed above in this section we will use the open-source system Apache Tike. 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 [10].

### 3.2 Parsing text into words vector

In this section, we will parse text into a collection of words by using regular expression. A regular expression, regex or regexp [13] (sometimes called a rational expression) is a sequence of characters that define a search pattern. 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*** (1)

It is known, that every natural language has stop words, which used inside of sentence to relate words to each other. In the [14] and [15] there is a list of stop words for English and Russian languages. The next step consists of removing stop words from collection words. After removing stop words we can describe document the following version.

$$D_j = \left((d_1, n_1), (d_2, n_2), .., (d_p, n_p)\right) \quad (2)$$

$$\forall d_i \notin H_j \quad (3)$$

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 1 it was shown the structure documents in (2) version.

### 3.3 Searching Web pages

In this section, we will describe how to find Web pages by collection of words. As noted above to find web pages we will use search engine systems like Yahoo, Google, Yandex.XML, etc. These systems have own API working with them. In the [12] has been shown the process of finding Web-page based on the terms via Yandex.XML service. We will divide *D* document on the (1) form into some parts *k* sequence of term which consists of ($k \geq 1$) terms. As a result, we will have the following collection:

$$R = [[d_1, d_2, ..., d_k], [d_{k+1}, d_{k+2}, ..., d_{k+k}], ..] \quad (4)$$

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=$d_1$+ $d_2$+..$d_k$* (5)
*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 (6) object which consists set of $l_j$ URL address of Web-pages and $p_j$ number of their occurrences.

$$L = ((l_1, p_1), (l_2, p_2), ..., (l_s, p_s)) \quad (6)$$

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

$$L' = ((l'_1, p'_1), (l'_2, p'_2), ..., (l'_s, p'_s)) \quad (7)$$
$$\forall p'_j \in [p_1, p_2, .., p_s], \ \forall l'_i \in [l_1, l_2, .., l_s], \ \forall p'_{i-1} \leq p'_i$$

The list of Web-pages which are similarity with the D document we can get on base (7) formula. At the next step, we get content of Web-page from the Internet on base their URL addresses.

$$L_t = [T_1, T_2, .., T_s] \quad (8)$$

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.
As it is known, Web-page is the text on HTML format; it will be shifted to simple text with the help of the system Apache Tika. The formed text is taken apart to words based on (1) then the stop words will be removed from the text. As a result, each $T_i$ document can be transformed in the following form.

$$T = ((t_1, m_1), (t_2, m_2), .., (t_g, m_g)) \quad (9)$$
$$\forall t_i \notin H_j \quad (10)$$

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.

### 3.4 Compare Documents

Detecting plagiarism is not a simple string match. Our algorithm detecting similarity two documents based on the collection of words both documents. In other words similarity of both documents calculates on-base similarity (2) and (9) objects. At first, we will calculate the weight of both documents.

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

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

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.

$$[d_1, d_2, ..., d_p] \cap [t_1, t_2, ..., t_s] = [x_1, x_2, .., x_k] \quad (11)$$

$$X = ((x_1, n_1, m_1), (x_2, n_2, m_2), ..., (x_k, n_k, m_k)) \quad (12)$$

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.

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

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

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

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

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

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.

### 3.5 Compare on base synonyms of terms

As it is known, the majority of words in the natural language have synonyms forms, and the meaning of the sentences isn't usually ruined by using word's some synonym forms. In the last section, we did not mention about synonymous. In this section, we will implement changes that algorithm to support checking also word synonyms during compare documents. As it is known, majority of words in the natural language have synonyms forms, and the meaning of the sentences isn't usually ruined with using other forms. On the (11) formula there isn't mentioned about synonymous. The following algorithm is devoted to compare texts based on synonym forms of the word.

1. At first, we will get term collection from $D$ document which does not exist in the $[x_1, x_2, ..x_k]$ collection, and we will mark them $[d_{11}, d_{21}, .., d_{r1}]$. Here $r$ - count of words that not exist in the $[x_1, x_2, ..x_k]$ collection.
2. Next step, we define $[d_{i2}, d_{i3}, .., d_{it}]$ a list of synonym forms $d_{i1}$ term, if $d_{i1}$ term has not got any synonym forms we go to 4th step.
3. After getting all synonym forms of $d_{i1}$ term we looking for them from the collection of words in the $T$ document. If any synonym version $d_{i1}$ term contains in the $T$ document, we will add it into $X$ object as $(x_{k+1}, n_{k+1}, m_{k+1})$ new element. $n_{k+1}$ - the occurrence number of $d_{i1}$ term in the document $D$, $m_{k+1}$ - the occurrence number of $d_{ij}$ term in $T$ document.
4. Steps 2 and 3 are followed for the all next terms.

5. As a result, we have an $X$ object which is filled with the synonym of the form of terms. On-base new-formed $X$ and by recalculating (13) and (14) formulas we will have a similarity degree of both $D$ and $T$ documents taking into account the version of the synonym terms.
6. On-base new formed $X$ and by recalculating (13) and (14) formulas we will have similarity degree of both $D$ and $T$ documents taking into account the version of the synonym terms.

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.

## 4 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 SEO (Search Engine Optimization) engineering workpieces on the internet that allow Web-pages to be viewed on the foreground during the searching information from the internet [16]. It will impact the process of searching for similar information to the text.

## 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] The Anatomy of a Large-Scale Hypertextual Web Search Engine". Computer Science Department, Stanford University, Stanford, CA. Retrieved January 27, 2009.

[6] Pitta, Julie (April 13, 1996). "Yipee for Yahoo! IPO Spurs Trading Frenzy as Shares More Than Double". Los Angeles Times. Archived from the original on June 20, 2013. Retrieved July 18, 2017.

[7] 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.

[8] 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

[9] 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

[10] Chris A. Mattmann and Jukka L. Zitting. 2011. Tika In Action. ISBN 9781935182856.- 256 p. 2011 y.

[11] Erik H., Otis G., Michael McC. Lucene in Action – Covers Apache Lucene v.3.0// Manning Publications.-486p.,2009 y.

[12] Yandex.XML API. https://passport.yandex.ru/ Retrieved: 4 November 2019.

[13] Ortiz-Cordova, A. and Jansen, B. J. (2012) Classifying Web Search Queries in Order to Identify High Revenue Generating Customers. Journal of the American Society for Information Sciences and Technology. 63(7), 1426 – 1441.

[14] English Stop Words. http://xpo6.com/list-of-english-stop-words/ Retrieved: 4 November 2019.

[15] Russian Stop Words. https://github.com/stopwords-iso/stopwords-ru/blob/master/stopwords-ru.txt Retrieved: 4 November 2019.

[16] Beel, Joran and Gipp, Bela and Wilde, Erik (2010). "Academic Search Engine Optimization (ASEO): Optimizing Scholarly Literature for Google Scholar and Co" (PDF). Journal of Scholarly Publishing. pp. 176–190. Retrieved April 18, 2010.