The problem of fuzzy duplicate detection of large texts

E V Sharapova[1] and R V Sharapov[1]

[1]Vladimir State University, Orlovskaya street 23, Murom, Russia, 602264

Abstract. In the paper, considered the problem of fuzzy duplicate detection. There are given the basic approaches to detection of text duplicates–distance between strings, fuzzy search algorithms without indexing data, fuzzy search algorithms with indexing data. Thereview of existing methods for the fuzzy duplicate detection is given. The algorithm of fuzzy duplicate detection is present. The algorithm of fuzzy duplicate texts detection was implemented in the system AVTOR.NET. The paper presents the results of testing of the system.The use of filtering text, stemming and character replacement, allow the algorithm to found duplicates even in minor modified texts.

1. Introduction

Today computer technology and the World Wide Web have become part of our lives. Billions of users use the Internet every day. The Internet contains a huge number of documents. Many books are transformed in a digital form. Users can read books, articles, newspapers directly from computers and electronic gadgets. Today, in the Internet there are many studying materials: lectures, textbooks, methodical literature, etc. There are big collections of essays, course and diploma projects and scientific dissertations. It leads to problem un-controlling coping of information.

The text of document may be a part of another document, formed from several documents, may be modified part of another document (with a change of words to synonyms, endings, times, etc.). In this way, the document is fuzzy duplicate of other documents.

There is the problem of text checking to presence of fussy duplicates. The purpose of work is the algorithm creation of fuzzy duplicate detection.

2. Methods of the duplicate texts detection

2.1. The distance between strings

The task of comparing texts reduced to comparing their strings. For this reason, assume that the text document is aone string of a large length. There is a requirement to determine the measure of the string similarity. This measure is called the distance between strings.

The distance is the minimum number of string characters changes, which is necessary for the transformation of one string within another. The Hamming and Levenshtein algorithms used to determine the distance [1].

(a) The algorithm of Hamming [2].

The algorithm of Hamming is used to search the distance between strings of equal length, by using the operation "replacement". This algorithm is not suitable to search the distance between the strings, various by length.

(b) The algorithm of Levenshtein [3]

The algorithm of Levenshtein uses operations "replacement", "insert", "delete". They allow to search the distance between strings, different by length. But time of calculation of distance between strings is disproportionately increases with increases of strings size. Therefore, the use of this algorithm is only suitable for comparing multiple pages of documents.

(c) The algorithm of Damerau-Levenshtein [4].

This is modification of the Levenshtein distance. It is also consider the operation of "transposition" the next two letters.The method of calculation of Levenshtein distance was the basis for several algorithms of search for common subsequences.

(d) The algorithm of Wagner and Fisher [5].

This method is based on calculating the Levenshtein distance between the strings prefixes (substrings). The matrix of editorial prescription is made. It contains a summary value of Levenshtein distance (minimum weight operations to change characters). The size of editorial prescription matrix is $(p+1)\cdot(b+1)$, where p and b – compared strings prefixes.

The number of string comparisons is $k\cdot p\cdot b$, where k – coefficient (for natural language $k=0,2$). The complexity of algorithm is $O(p\cdot b)$ [6].This method is the easiest way to create of editorial prescription.

(e) The algorithm of Masek and Paterson [7].

The algorithm is modification of Wagner and Fisher algorithm with applying the approach Arlazorov, Diniz, Kronrod and Faradjev. The distance matrix in it is separated into submatrixes with edges computed in accordance with contiguous to their submatrixes.

The complexity of algorithm is $O(k\cdot(p\cdot b/log(p)))$.

This algorithm faster than the algorithm of Wagner and Fischer, but, at the direction of the authors themselves [8], it reaches real speed only when comparing very long strings.

(f) The algorithm of Ukkonen [6].

For this algorithm it is required the construction of suffix tree of strings set to minimize search time in substring.

Complexity of algorithm is $O(k\cdot m)$, where m – maximum vertex depth of suffix tree.

Algorithm of Ukkonen is more applicable for search of exact matches strings. If searching for very different texts, then working time is greatly increased.

(g) The algorithm of Hirschberg [9].

The algorithm is modification of Wagner and Fisher algorithm. It is not calculated the distance between the strings but the distance between the longest common subsequence.

The complexity of algorithm is $O(k\cdot(p+b))$.

(h) The algorithm of Hunt and Szymanski [10].

It is based on searching of maximum increasing path on a matching elements graph of compared strings.

The complexity of algorithm is$O(k\cdot(g+b)\cdot log(b))$, where g – the number of positions in which the string symbols are the equal.

Under certain circumstances, this algorithm shows good results, but in other cases, the comparison time is quadratic.

(i) The suffix tree [11].

This method is proposed of McCreight in 1976. It is similar to the Ukkonen algorithm, but the suffixes are added in reverse order.

The complexity of algorithm is $O(k\cdot m)$.

Disadvantage of algorithms working with suffix trees is use a fixed alphabet. It is suitable only for search of exact match strings.

2.2. Fuzzy search algorithms without indexing data

Let's consider fuzzy search algorithms without data indexing [12]. They are used to search for the previously unknown and unmanufactured texts.

(a) Linear search algorithm [13].

The algorithm uses the distance metric and applies it to the text words. Efficiency of the method depends on the number of errors and mismatches of texts. More numerous they are the more increases the comparison.

The algorithm complexity is $O(s{\cdot}p)$, where s – the number of errors made when checking, p – the text length.

(b) The Bitap algorithm [14].

This algorithm is applied more than the linear search. In view modifications it calculates the Levenshtein distance between words. At normal conditions, speeds of these two algorithms are the same. The algorithm complexity is $O(s{\cdot}p)$, but the speed of this algorithm significantly higher on long words than a linear method.

2.3. Fuzzy search algorithms with indexing data

It is necessary to index the text for the application of this group of algorithms. The dictionary is constructed by thetext. It contain words and their position in the text. On base of dictionary creates the index required for further search.

(a) The algorithm of sample expansion.

This algorithm is composedfor the search of many erroneous words and then look for them in the dictionary.

Complexity of the algorithm is $O((b \cdot |A|)^s \cdot b \cdot \log p)$, where A – the dictionary size, b and p – compared words, s – the number of errors in a word.

Disadvantage of the algorithm is that with increasing number of errors, the time of its operation also increase.

(b) The N-gram algorithm [15].

This algorithm is based on words separation to parts (substrings) with the length N (N-grams). It is compiled N-gram list, contain all the words of this text with occurrences of the substring. Next separated similarly inquiry is searched by this list.Most often used division into trigrams.

Complexity of algorithm is$O(w{\cdot}H)$, where w – the probability of occurrence of N-grams in the word, H – the number of words in the N-gram list.

(c) Hashing by signature [12].

This algorithm represents a word of the text in a code ("hash"), consisting of digits. Indexed words recorded in the table. Query also indexed is make the search in the table.

Complexity of algorithm is $O(|E|^s \cdot \frac{p}{2^{|E|}})$, where E – the hash function, representing the word as a code, p – the text length, s – the number of errors.

(d) The shingles algorithm [16].

The source text is separated into a set of sequences of a defined length (shingles). Documents are similar if the equal some amount of their shingles. Number of shingles is sufficiently large. Therefor are used methods reduce of shingles set.

For example, can be considered only those shingles, whose fingerprint is divided into a defined number n. Other way are selected shingles with the minimal fingerprint, or simply taken a defined amount of shingles.

(e) The algorithm of D. Fetterly [17].

For each document are computed 84 fingerprint by algorithm of Karp-Rabin using a random sample of shingles. These shingles are separated into 6 groups, which are called "super shingle". The combination of 2 super shingle is "mega shingle". 15 mega shingle are computed for the each document. Two documents are similar if they have the same at least one mega shingle.

Algorithm complexity is $O(S1{\cdot}S2)$, where $S1$ – number of shingles from document 1, $S2$ – number of shingles from document 2.

(f) The I-Match algorithm [18].

Dictionary A is built for a group of documents. Dictionary consist of words with the average values of a code IDF. For each document is created dictionary B and determine the intersection of A and B.

Words of document, included in the intersection, are used to build I-Match signature of the document. If I-Match signatures are equals, then texts are similar.

(g) The algorithm of key words [19].

From the index are selected "key" words with threshold frequency. For a document constructed binary vector (signature). Value of i-th coordinate of the vector is equal to 1 if the frequency of i-th "key" word in the text is more than the threshold value, or 0 if its frequency in the text is less than the threshold. Texts are similar if they have the same signature.

*(h) The TF*IDF algorithm [20].*

It is a calculated inverse frequency of word in the collection of documents, called as Inverse Document Frequency (IDF). It is a calculated frequency of the word in the documents called as a Term Frequency (TF). A Values TF*IDF helps measure similarity documents to query. This is a very popular method. The disadvantage of this method is that it ignores a relative arrangement of the words in the documents.

3. The algorithm of fuzzy duplicate texts detection

At present, there are a large number of approaches for finding similar strings and small texts [21-25]. For large texts (from 1000 words), the search for coincidences is difficult due to the large time expenditure. Therefore, it is of interest to construct an algorithm for finding coincidences in large texts in a short time.

Let's consider our algorithm of fuzzy duplicate detection.

3.1. A character set conversion.

A checked text document can have different encodings. For a Russian language it is win1251, KOI8-R, UTF-8, ASCII. Therefore, the checked text document is converted to the uniform character set UTF-8.

3.2. Converting a text to the single long length string.

A text document is a set of strings. Formatting of the same documents may be very different. Therefore, it is necessary to simplify the presentation of the text. For this purpose, we used a linear representation of the text as a sequence of characters. In other words, the text appears as a single long length string (an wide string).

3.3. Pre-processing of a text.

It is perform pre-processing of a text: the text of a document is replaced by a filtered copy. For this purpose the following steps are performed:

- Removal of HTML tags.

A text document can to contain HTML tags. Because HTML tags are used for a document formatting, they do not affect to its contents. Therefore presence of HTML tags will just interfere to checking and must be removed from a document.

- Removal of punctuation marks and special characters.

In the text there are punctuation marks and special characters, including tabs, new line, etc. All this characters are removal from the document.

- Case conversion.

In the text of documents many words can be written with a lowercase or uppercase letters, consist entirely of uppercase letters or contain mixed uppercase and lowercase letters in words. Therefore, all characters are converted to lowercase.

- Processing of replacement characters.

Sometimes, in text documents there are replacement characters. For example, in Russian words several letters of the Russian alphabet replaced the similar Latin letters (such as “o”, “a”, “e”, “y”, “x”, “c”, “p”, etc.). In this case, it is processed the replacement characters and they transform to an original form.

- Removal of stop words.

Stop words are common words occurring in an almost every document, such as "the", "of", "on", "a", "at", "by" etc. Therefore, these words do not help in the search of text duplicates, but increase the processing time. They must be removed from the document.

- Filtration of the text (not informative words removal).

A filtering of a text is a removal the most frequent words, not informative words etc. Also, filtering words that contain special characters, digits, words of great length, etc. This procedure allows to asignificantly reducing the amount of a computation (the length of the checked text).

- Stemming (processing of word endings).

A stemming is processed closure words. In this case, they are simply deleted. This avoids the effect of such modifications of the text, as a change in the singular and plural forms, masculine and feminine, a present and a past tense etc.

3.4. Detection of fuzzy duplicates for a document.

Our algorithm is based on a method of shingles. The text is divided into a sequence of words. A length of the sequence is fixed (5 words). For the each sequence of words is computed a MD5 code (shingles). The algorithm works with that code (figure 1).

A classical method of shingles used all words to create a sequence of words. We propose to consider not all the text of document but its a processed and filtered copy.

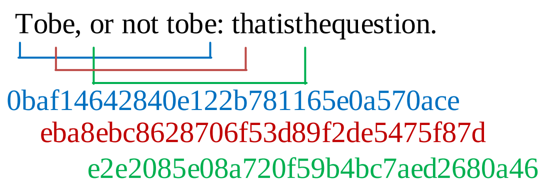
Tobe, or not tobe: thatisthequestion.
0baf14642840e122b781165e0a570ace
eba8ebc8628706f53d89f2de5475f87d
e2e2085e08a720f59b4bc7aed2680a46

Figure 1. Dividing the text on the shingles.

The number of matching MD5 codes shows the measure of matching documents. If all codes in two documents are match, then documents are full duplicates. If there aren't match codes, then documents are different. If only a part of MD5 codes is match, documents are fuzzy duplicates.

4. The system structure

On the basis of the Vladimir State University, the authors developed the system of fuzzy duplicates detection. The system checks both the sources available on the Internet and on an internal database (databases of articles, course works and examinations etc.). The system generates a report with colouring duplicate parts of texts and viewing found duplicate sources. Let's consider the system structure [26] (figure 2).

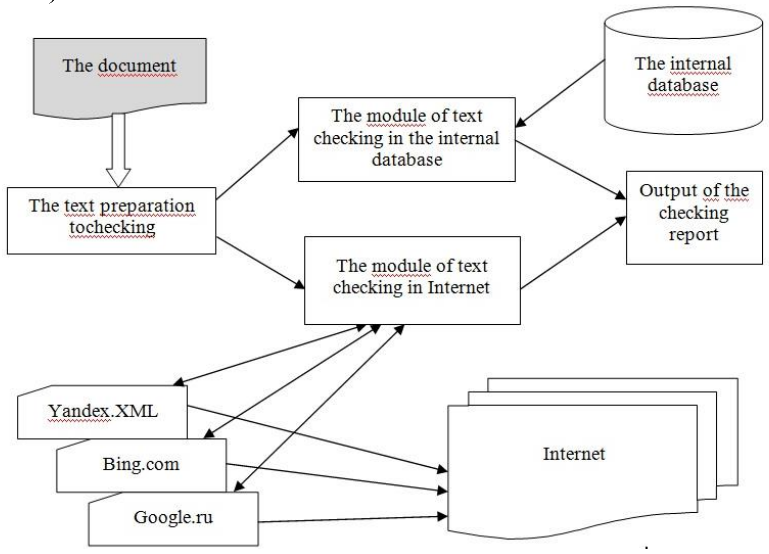
The document
The internal database
The module of text checking in the internal database
The text preparation tochecking
Output of the checking report
The module of text checking in Internet
Yandex.XML
Bing.com
Google.ru
Internet

Figure 2. The structure of the system forfuzzy duplicates detection.

A text preparation includes removal of HTML tags, removal of punctuation marks and special characters, case conversion, processing of replacement characters, removal of stop words, filtration of the text, a stemming.

The system of fuzzy duplicates detection includes two modules. An each module operates independently of each other.

The first module checks the internal database sources. Base sources include a database of articles, course works and examinations, diploma projects, textbooks, lecture courses. Sources are stored in the form of a full-text and in the form of specially organized search index. A search index is needed for quick checking duplicates of a text in a database sources. There is no need for each check to a view all the available texts and make them fairly laborious process. All the necessary information for search is already included in a structured search index. A search index generated from the pre-processing text.

The second module checks the documents from the Internet. Text of the document is divided into informative parts. The number of parts depends on the size of the document. Next, using a search engine is searched for a sources containing these informative pieces. A search module use Yandex.XML, Yandex.com, Google.ru, Rambler.ru, Yahoo.com, Poisk.Mail.ru, Nigma.ru etc. Received documents are compared with the original document. To do this, the document format is determined (html-document, txt-file, doc-or rtf-document, pdf-file). From a html-document all markup tags are removed. The files *.doc, *.docx, *.rtf and *.pdf converted in a plain text format with no markup. In a next step documents are pre-processed and computed the similarity with the original document.

The main requirements for the system are the completeness and accuracy of duplicates detection. We did not set thetask of reducing a check time.

The algorithm of fuzzy duplicate texts detection was implemented in a system AVTOR.NET [27]. The system checks both the sources available on the Internet and on internal database (databases of articles, course works and examinations, etc.). The system generates a report with colouring duplicate parts of texts and viewing found duplicate sources.

5. Practical use

In 2016 the system Avtor.NET was used to check for a plagiarism of diploma projects full texts in department of technospheric safety of Vladimir State University. During the 10 days was checked more than 70 diploma projects. An each work has been from 60 to 120 pages. For this purpose, it was necessary to get up a new module to reduce the load on the search engines. A system Avtor.NET coped with the task. All diploma projects have been checked and we have received detailed reports on the presence of a plagiarism.

To test the efficiency of the algorithm we were compiled of three group of tests:

- Fussy duplicates of one document with a reordering of the sentence (T1);
- Fussy duplicates with a change in the text of the times and kinds of words (T2);
- Fussy duplicates of one document with a reordering of the sentence and the addition of the original text between sentences (T3);
- Fussy duplicates of some documents with a reordering of the sentence (T4).

All the tests had a size about 4000 characters and contained an average of 400 words. We used collection of essays, available on the Internet, to creating of tests. 10 tests of each groups was composed.

We compared the results of systems Antiplagiat, AdvegoPlagiatus, AntiPlagiarism.NET, Etxt.ru,Content-watch.ru, Text.Rucont.ru, Unicheck.comwith our algorithm (system Avtor.NET). To assess the quality of detection was used Recall, showing what percentage of duplicate text was really detected. Precision of all the systems was high and trends to 1.

All systems correctly handle test T1. Slightly worse result of Etxt.ru (Recall = 0.75) is due of features comparison algorithm implementation. With the test T2, the systems AdvegoPlagiatus, Text.Rucont.ru and Avtor.NET coped well. Worst results showed systems Unicheck.com (Recall = 0.32), AntiPlagiarism.NET (Recall = 0.58) and Antiplagiat (Recall = 0.63).

Table 1. Test results.

System	T1	T2	T3	T4
Antiplagiat	0.98	0.63	0.89	0.99
AdvegoPlagiatus	0.99	0.92	0.73	0.89
AntiPlagiarism.NET	0.83	0.58	0.72	0.90
Etxt.ru	0.75	0.83	0.65	0.49
Content-watch.ru	0.90	0.76	0.87	0.90
Text.Rucont.ru	0.94	0.95	0.94	0.91
Unicheck.com	0.95	0.32	0.74	0.96
Avtor.NET	0.99	0.97	0.94	0.99

Test T3 was well overcome by Text.Rucont.ru, Avtor.NET (Recall = 0.94), Antiplagiat (Recall = 0.89) and Content-watch.ru (0.87).

Test T4 badly passed only Etxt.ru (Recall = 0.49). All other systems showed very good results.

How can we see the system Avtor.NET coped well with all four tests. The algorithm copes well with the replacement of endings, permutations of pieces of text, compilation from several sources and the alteration of texts with the insertion of new words.

6. Conclusion

The system Avtor.NET correctly handle all texts and show results that are not inferior and sometimes exceed the results of existing systems.Thus, it was created the algorithm of fuzzy duplicates detection. The use of filtering text, stemming and character replacement, allow the algorithm to found duplicates even in minor modified texts.

7. References

[1] Sharapova E V 2014 Analysis of methods and systems for fuzzy duplicate detection *Proc. of 14 International multidisciplinary scientific Geoconference SGEM2014. Informatics, Geoinformatics and Remote Sensing* **1** 27-33

[2] Hamming R W 1950 Error detecting and error correcting codes *Bell System Technical Journal* **29** 147-160

[3] Levenshtein V I 1966 Binary codes capable of correcting deletions, insertions, and reversals *Soviet Physics Doklady* **10** 707-710

[4] Damerau F J 1964 A technique for computer detection and correction of spelling errors *Communications of the ACM* **7** 171-176

[5] Wagner R A and Fischer M J 1974 The string-to-string correction problem *Journal of the ACM* **21** 168-173

[6] Gusfield D 2003 *Strings, trees and sequences in the algorithms: Computer Science and Computational Biology* (Cambridge University Press) 654

[7] Masek W J and Paterson M S 1980 A faster algorithm for computing string-edit distances *Journal of Computer and Systems Sciences* **20** 18-31

[8] Masek W J and Paterson M S 1983 How to compute string-edit distances quickly *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison* (Addison Wesley, Reading, MA) 337-349

[9] Hirschberg D S 1975 A linear space algorithm for computing maximal common subsequences *Communications of the ACM* **18** 341-343

[10] Hunt J W and Szymanski T G 1977 A fast algorithm for computing longest common subsequences *Communications of the ACM* **20** 350-353

[11] McCreight E M 1976 A space-economical suffix tree construction algorithms *Journal of the ACM* **23** 262-272

[12] Boytsov L M 2004 Classification and experimental study of the modern algorithms for fuzzy dictionary search *Proc. of 6-th Russian Scientific Conference Digital Libraries: Advanced Methods and Technologies, Digital Collections* 10

[13] Knuth D 1997 *The Art of Computer Programming* (Addison-Wesley) 396-408

[14] Baeza-Yates R and Navarro G 1996 A faster algorithm for approximate string matching *Proc. Combinatorial Pattern Matching* 1-23

[15] Nagao M and Mori S 1994 A New Method of N-gram Statistics for Large Number of n and Automatic Extraction of Words and Phrases from Large Text Data of Japanese *Proc.of the 15th International Conference on Computational Linguistics* 611-615

[16] Broder A 1998 On the resemblance and containment of documents *Compression and Complexity of Sequences SEQUENCES'97* 21-29

[17] Fetterly D, Manasse M and Najork M 2003 On the Evolution of Clusters of Near-Duplicate Web Pages *Proc. of the First Conference on Latin American Web Congress* 37-45

[18] Kolcz A, Chowdhury A and Alspector J 2004 Improved Robustness of Signature-Based Near-Replica Detection via Lexicon Randomization *Proc. of KDD* 605-610

[19] Ilyinsky S, Kuzmin M, Melkov A and Segalovich I 2002 An efficient method to detect duplicates of Web documents with the use of inverted index *Proc. of WWW Conference*

[20] Salton G and McGill M J 1983 *Introduction to modern information retrieval* (New York: McGraw-Hill) 448

[21] Mikhaylov D V, Kozlov A P and Emelyanov G M 2015 An approach based on TF-IDF metrics to extract the knowledge and relevant linguistic means on subject-oriented text sets *Computer Optics* **39(3)** 429-438 DOI: 10.18287/0134-2452-2015-39-3-429-438

[22] Mikhaylov D V, Kozlov A P and Emelyanov G M 2016 Extraction of knowledge and relevant linguistic means with efficiency estimation for the formation of subject-oriented text sets *Computer Optics* **40(4)** 572-582 DOI: 10.18287/2412-6179-2016-40-4-572-582

[23] Evdokimova N I and Kuznetsov A V 2017 Local patterns in the copy-move detection problem solution *Computer Optics* **41(1)** 79-87 DOI: 10.18287/2412-6179-2017-41-1-79-87

[24] Mikhaylov D V, Kozlov A P and Emelyanov G M 2017 An approach based on analysis of n-grams on links of words to extract the knowledge and relevant linguistic means on subject-oriented text sets *Computer Optics* **41(3)** 461-471 DOI: 10.18287/2412-6179-2017-41-3-461-471

[25] Sharapova E V 2014 One way to fuzzy duplicates detection *Proc. of 14 International multidisciplinary scientific Geoconference Informatics, Geoinformatics and Remote Sensing* **1** 273-277

[26] Zelenkov Y G and Segalovich I V Comparative analysis of methods for fuzzy duplicate detection for Web-documents *Proc. of 9-th Russian Scientific Conference Digital Libraries: Advanced Methods and Technologies, Digital Collections* 9

[27] Sharapova E V 2014 System of fuzzy duplicates detection *Applied Mechanics and Materials* **490-491** 1503-1507