# Data Deduplication in Slovak Corpora

Vladimír Benko

Ľ. Štúr Institute of Linguistics, Slovak Academy of Sciences, Bratislava, Slovakia

**Abstract.** Our paper describes our experience in deduplication of a Slovak corpus. Two methods of deduplication – a plain fingerprint comparison and an n-gram comparison – are presented and their results compared.

## 1 Introduction

Deduplication is a special technique of detection and removal of duplicate contents in digitally stored data. Motivations for such activity include a more efficient use of data storage space (duplicate data can then be stored in a single copy only), detection of plagiarism (sections of identical text without proper quoting usually indicates an author's inappropriate activity), or decreasing the size of index structures in data retrieval systems.

In text corpora, the problem of duplicate contents started to be strongly felt with the advent of web corpora. Duplicate texts distort frequencies of occurrence of lexical units and bias the statistics used to compute collocations. Expressions of low frequency found repeatedly in duplicate documents or paragraphs tend to receive very high scores of salience. In the Word Sketch Engine, that is being used at our Institute in lexicographic projects [1], such collocations appear in first place in the respective tables, causing undesirable noise. The resulting concordance then looks as follows:

Corpus: **Iura (legal-1.0.10) 142 M (#591)**
Hits: **35** (0.2 per million)
Page 1 of 2 Go Next | Last

| | |
|---|---|
| 1.0000265 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov ( Protokol |
| 1.0000938 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov ( Protokol |
| 1.0002059 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov ( Protokol |
| 1.0006681 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov ( Protokol |
| 1.0014582 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov ( Protokol |
| 1.0015063 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015063 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015063 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015064 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015064 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015064 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015065 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015065 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015065 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015066 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015066 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015066 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015067 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |
| 1.0015067 | 12 . augusta 1949 o ochrane ***obetí*** **medzinárodných** ozbrojených konfliktov a konfliktov |

Subsequent analysis shows that out of 35 occurrences of the collocation *obete medzinárodných ozbrojených konfliktov* "victims of international armed conflicts", only 2 are really unique and all the rest are just repetitions of the same sentence within the same law or in its various revisions.

## 2 Plain Fingerprint Deduplication

Detecting duplicates by direct pair-wise text comparisons in large collections is technically not feasible as the number of comparisons grows quadratically. There exists, however, a simple method that decreases the computational complexity of this task dramatically. It is based on the idea that, for each data segment under comparison, it is sufficient to compute a "fingerprint", i.e. a short fixed-length bit pattern [2] where equal data will have an equal value of their fingerprints.[1] The fingerprints can be computed, e.g., by means of a cryptographic hashing algorithm. Duplicate fingerprints can be easily detected by their sorting and subsequent unification [3]. This method leads to detection of exact (100%) duplicates.

In reality, this method can be easily implemented by means of standard Linux utilities *sort*, *uniq* and *md5sum*, complemented by two simple filters.

In regard to paragraph-level deduplication in a text corpus, the whole procedure can look as follows. Firstly, a paragraph identifier is assigned to each paragraph. Than a filter is used that will save the contents of each paragraph into a work file and call the *md5sum* program to compute the fingerprint that will be appended (along with the paragraph's identifier) to the fingerprint file.

```
1c32b3d252f2f66207352c95e02f04f5  0000000.00000
4d7d068d0e8c37aaf76619afdb41c937  0000000.00001
41a4459ed67cf1d15cca63b8e3efac6c  0000000.00002
ae5eae7f1645cbaa0d617a2089feea89  0000000.00003
...
af22e8df9bc9bfd3930d433b9fec39c7  1500125.00827
e98135d33b38729a90c3fb0465e25b62  1500125.00828
52ab380379a284145b89cca9a3581567  1500125.00829
d19c7528f54b7c4a968899c804675b0a  1500125.00830
```

After sorting the fingerprint file according to the first column, the duplicate fingerprints will appear together (we have marked them with an asterisk).

```
0000002797f70f8e9f666fb407db5195  1499872.00389  *
0000002797f70f8e9f666fb407db5195  1499876.00388  *
000000466f8914041e68767a38f392a0  0601609.01350
00000097f3f4b3521ceb78e26c000213  1465277.00301
0000019b4aeb3b3f8bf80bef210361ed  1304808.00003
000001f224498662798071a5580c7d80  0660013.00012
00000216af425ae2ac4112994546c9ef  0089946.00013  *
00000216af425ae2ac4112994546c9ef  0091158.00012  *
00000257b12f4211909124d2b7f18fc5  0979319.00019
...
```

[1] In using hash functions, there exist situations where two unequal segments have equal hash values (so-called *collision*). Although the probability of this happening is non-zero, it is so small that (in the context of language corpora) it can be safely ignored.

As a result of unification, each different fingerprint value in the file will appear just once.

```
0000002797f70f8e9f666fb407db5195  1499872.00389  *
000000466f8914041e68767a38f392a0  0601609.01350
00000097f3f4b3521ceb78e26c000213  1465277.00301
0000019b4aeb3b3f8bf80bef210361ed  1304808.00003
000001f224498662798071a5580c7d80  0660013.00012
00000216af425ae2ac4112994546c9ef  0089946.00013  *
00000257b12f4211909124d2b7f18fc5  0979319.00019
```

In the end, the file will be sorted according to second column, which will result in the list of paragraphs that are *not* to be removed. The final simple filter will remove the duplicate paragraphs in the original source file.

### 2.1 Deduplication in the Slovak National Corpus

Up to Version 5.0 of the Slovak National Corpus (SNC), the data duplicity problem has not been seen as very important, as most duplicities were avoided by careful selection of the source texts. The situation, however, has rapidly changed with the advent of Version 6.0 that received a large collection of newspaper texts form the Petit Press Publishing House. These contained a large amount of data from Slovak regional weeklies where many articles were identical. The respective documents represented print pages converted from PDF format, where, due to imperfection of the conversion procedure, the paragraph breaks of identical texts were not identical.

### 2.2 Paragraph-level Deduplication

In the following text we shall present the results of the plain fingerprint deduplication method applied to the largest SNC corpus – *prim-6.0-juls-all* [4]. The first filter mentioned in the previous section was modified so that it would not take into account punctuation, special graphic characters, and digits. This allowed us to also identify as duplicates paragraphs

```
<p>19.30 Noviny STV</p>
<p>23.45 Noviny STV</p>
<p>1.40 Noviny STV</p>

<p>12. Marseille 14 5 4 5 13:13 19</p>
<p>15. Marseille 15 4 5 6 13:15 17</p>
<p>10. Marseille 18 6 6 6 18:17 24</p>
```

representing items of the TV schedule and the football league table, respectively, the differences of which are not lexicographically interesting.

The input source file contained 1,390,408 documents with 51,536,717 paragraphs containing 1,226,218,915 tokens. The main procedure lasted approximately 19 hours,

while the computation of fingerprints took 18 hours and 20 minutes[2] (Intel Xeon 2.83 GHz, 8 GB RAM, hardware RAID, Ubuntu 12.10 LTS). The duplicate paragraphs were not deleted from the corpus but rather just marked so that they would not be taken into account in computing the word sketches. The advantage of such an approach is that the corpus user is not deprived of the context at the boundary of duplicate and unique content.

The result of deduplication is shown in the following table.

| | **Paragraphs removed** | **Paragraphs left** | **Total** |
|---|---|---|---|
| **Paragraphs** | 21,251,221 | 3,085,496 | 51,536,717 |
| **Paragraphs in %** | 41.24 | 58.76 | 100.0 |
| **Tokens** | 167,743,453 | 1,058,475,462 | 1,226,218,915 |
| **Tokens in %** | 13.68 | 86.32 | 100.0 |

Now, we would like to know what kind of data have been removed. It is obvious that only a tiny fraction of the millions of removed paragraphs can be inspected "manually". We have therefore decided to perform a frequency analysis of the removed paragraphs according to their lengths (in tokens). Respecting the expected distribution, paragraphs were grouped by power of 2, i.e. group "1" contained paragraphs of 1 token, group "2" paragraphs of 2 and 3 tokens, group "4" paragraphs of 4 to 7 tokens and so on. The results are summarized in the following table:

| **Paragraph length** | **Paragraphs removed** | **Paragraphs left** | **Total** |
|---|---|---|---|
| 1 | 2,899,765 | 313,757 | 3,213,522 |
| 2 | 5,385,687 | 1,760,629 | 7,146,316 |
| 4 | 6,430,346 | 5,065,587 | 11,495,933 |
| 8 | 4,369,821 | 6,858,103 | 11,227,924 |
| 16 | 1,459,344 | 6,202,353 | 7,661,697 |
| 32 | 512,667 | 5,349,893 | 5,862,560 |
| 64 | 166,836 | 3,425,382 | 3,592,218 |
| 128 | 24,435 | 1,083,748 | 1,108,183 |
| 256 | 2,218 | 200,193 | 202,411 |
| 512 | 93 | 22,537 | 22,630 |
| 1,024 | 8 | 2,811 | 2,819 |
| 2,048 | 1 | 443 | 444 |
| 4,096 | 0 | 46 | 46 |
| 8,192 | 0 | 13 | 13 |
| 16,384 | 0 | 1 | 1 |
| **Total** | **21,251,221** | **30,285,496** | **51,536,717** |

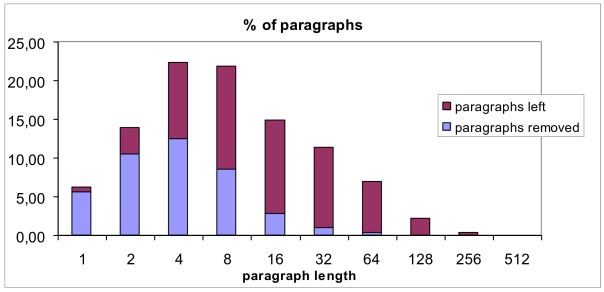
[2] It is obvious that a weak point of our implementation is the calculation of fingerprints by calling an external computationally "expensive" utility. Using a simpler hashing algorithm computed internally, it can be expected that the processing time could be significantly decreased.

The table shows that most of the paragraphs in groups 1 and 2 were removed, in groups 4 and 8 about 50% of the paragraphs were deleted, and from group 16 upwards most of the paragraphs were left.

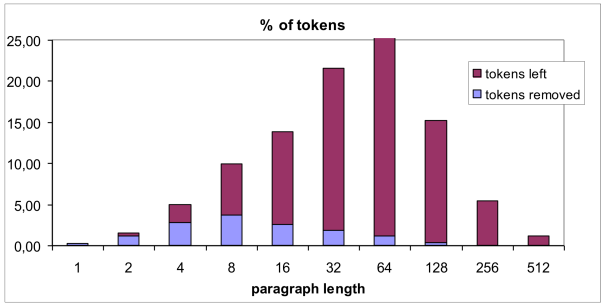
It is, however, more important to find out the token count in the deleted paragraphs.

| Paragraph length | Tokens removed | Tokens left | Tokens total |
|---|---|---|---|
| 1 | 2,899,765 | 313,757 | 3,213,522 |
| 2 | 13,714,814 | 4,557,662 | 18,272,476 |
| 4 | 33,490,542 | 27,787,256 | 61,277,798 |
| 8 | 45,839,898 | 75,856,713 | 121,696,611 |
| 16 | 30,790,283 | 139,059,109 | 169,849,392 |
| 32 | 22,213,759 | 241,725,320 | 263,939,079 |
| 64 | 14,072,053 | 300,236,783 | 314,308,836 |
| 128 | 3,949,428 | 182,717,735 | 186,667,163 |
| 256 | 700,310 | 66,312,816 | 67,013,126 |
| 512 | 58,514 | 14,489,082 | 14,547,596 |
| 1,024 | 10,412 | 3,835,447 | 3,845,859 |
| 2,048 | 3,675 | 1,169,544 | 1,173,219 |
| 4,096 | 0 | 250,981 | 250,981 |
| 8,192 | 0 | 146,055 | 146,055 |
| 16,384 | 0 | 17,202 | 17,202 |
| **Total** | **167,743,453** | **1,058,475,462** | **1,226,218,915** |

We can visualize the above data expressed in percentages in two graphs.




The columns in the first graph represent percentage shares of the respective paragraph groups with respect to the total number of corpus paragraphs. The light-coloured shading depicts the removed paragraphs and the dark shading indicates the paragraphs left. We can see that the first four groups contain the major portion of the removed paragraphs. The share of removed paragraphs declines sharply with the increasing length of the paragraphs. This is quite consistent with our intuition, as we can expect to have more matches in shorter paragraphs.




The second graph depicts the situation with tokens. The tendency is similar to the previous graph, and we can see that the largest contribution to the removed tokens comes from group "8". It is also quite interesting to find out that even group "64" contributes to the removed tokens considerably.

### 2.3 Sentence-level Deduplication

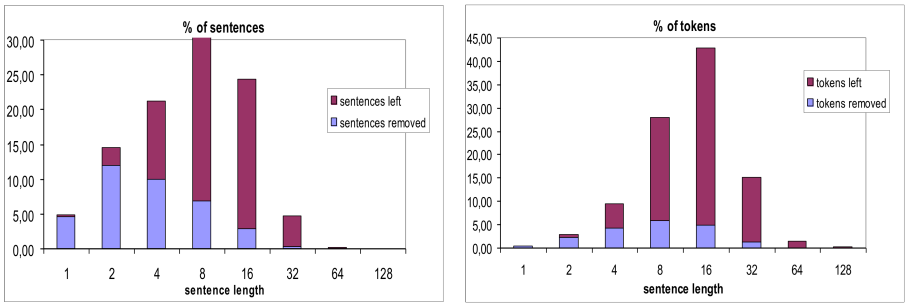
After having been deduplicated at the paragraph level, our corpus was processed by the Word Sketch Engine. Despite the removal of most duplicate concordance lines in the corpus, our lexicographers were not completely satisfied with the result. We therefore decided to repeat the whole procedure again, using the same technology at the sentence level.

The assignment of sentence identifiers revealed that there were 100,915,602 sentences in the corpus, which was roughly twice as many as the number of paragraphs. The deduplication was performed on a different computer (Intel Core i5 3.2 GHz, 12 GB RAM, software RAID, Ubuntu 12.10) and lasted approximately 55 hours[3]. The results were evaluated in a similar way as those of the paragraphs.

| | **Removed** | **Left** | **Total** |
|---|---|---|---|
| **Sentences** | 36,704,850 | 64,210,752 | 100,915,602 |
| **Sentences in %** | 36.37 | 63.63 | 100.00 |
| **Tokens** | 231,847,624 | 994,371,291 | 1,226,218,915 |
| **Tokens in %** | 18.91 | 81.09 | 100.00 |

If we compare the share of removed tokens by sentence-level deduplication with that of paragraph-level, we shall see that the number of removed tokens has increased 1.38 times and it represents almost 19% of all corpus tokens. The following graphs visualize the distribution of removed sentences and tokens by sentence length. The tendency shown in the graphs is similar to that of paragraph deduplication, with the difference being the greater contribution of shorter deleted segments (sentences).

[3] The computing time compared to the previous run was unexpectedly long. It is not clear what was the cause of this behaviour as all parameters of the computer used were higher (with the exception of the software RAID).




After this second deduplication phase the number of duplicate concordances observed by our users dropped to a minimum. We have decided to use this method also for other corpora of the SNC collection.

## 3 Detecting Near-duplicate Contents

As an alternate tool we decided to use the recently released open-source utility Onion[4] designed to detect near-duplicate contents in language corpora. This program was created within the framework of the PhD research of Jan Pomikálek at Masaryk University in Brno [5].

Onion ("One Instance Only") is also based on fingerprints but it does not compare whole segments but rather just n-grams of selectable length (7 by default). The input file is expected to be in one-column vertical format and it is processed in one pass. In the default mode, the deduplication is performed at the level of paragraphs marked by the ** ** ... ** ** tags. A paragraph is considered duplicate if it contains more than the threshold level of n-grams already encountered in previous text. The similarity threshold is a value in the range between 0 and 1, where 1 means a 100% match. The user can select deduplication at the level of segments marked by any pair of tags, with the most obvious values being documents and sentences. The duplicate segments can either be removed completely or indicated by a special mark in the first column of the output file. Implementation is optimized for speed (the fingerprints are computed by a computationally "cheap" routine *BUZ Hash* [6] and all data structures are stored in main memory) so the size of the corpus processed is limited by the size of available RAM only. Memory requirements can be substantially decreased by an alternate mode of program operation, where all computed fingerprints are saved into a file and deduplicated first. In the second pass it is necessary to keep only the duplicate fingerprints in the memory. According to information provided by the author, under typical conditions memory use can drop to only 10%. In this alternate mode of operation, the saved fingerprints can also be reused in subsequent experimentation with different values of similarity threshold and/or different levels of deduplication.

[4] URL: `http://code.google.com/p/onion/`

### 3.1 The Onion Experiment

To get an idea of the number of near-duplicates detected by n-grams of tokens, we decided to run an experiment with Onion applied to the corpus mentioned in the previous section. As Onion expects to get the input data in one column, at the beginning of our experiment the columns containing morphological annotation (Lemma, Tag) were removed from the source file.

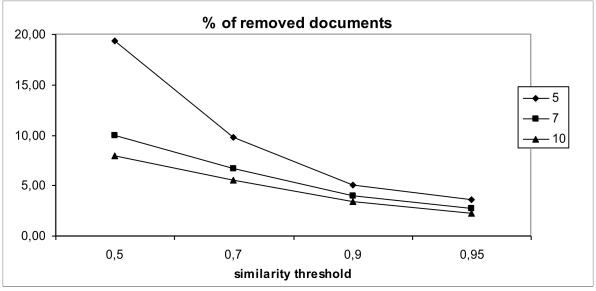
### 3.2 Document-level Deduplication

As a first step, we decided to observe the level of deduplication at the document level by means of 5-, 7- and 10-grams with four values of similarity threshold (0.5, 0.7, 0.9, and 0.95, respectively). For each value of n-gram we let Onion pre-compute the fingerprints first, which were subsequently used for deduplication with different values of similarity threshold. The computation of fingerprints lasted on average 27 minutes and the respective deduplication passes lasted typically 32 minutes.

The results of the deduplication are summarized in the following tables. The first one shows the numbers of removed documents with different values of n-grams and threshold levels.

| Similarity threshold | 5-grams | 7-grams | 10-grams |
|---:|---:|---:|---:|
| 0.5 | 269,076 | 137,780 | 110,108 |
| 0.7 | 136,158 | 92,215 | 77,183 |
| 0.9 | 69,572 | 54,864 | 47,381 |
| 0.95 | 49,498 | 38,140 | 31,098 |

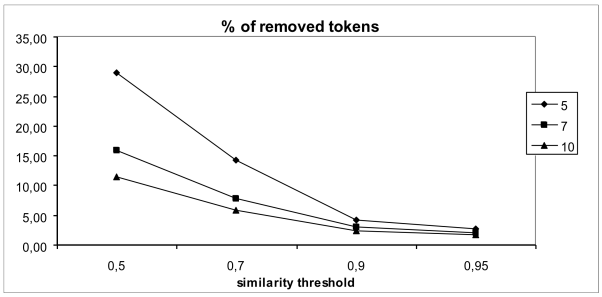
The above values expressed in per cents can be visualized as follows:




The next table indicates how the various deduplication parameters influence the numbers of removed tokens.

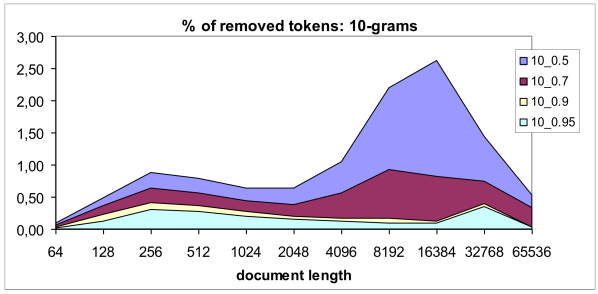
| Similarity threshold | 5-grams | 7-grams | 10-grams |
|---|---|---|---|
| 0.5 | 354,704,820 | 194,226,021 | 139,317,046 |
| 0.7 | 174,064,712 | 94,776,159 | 71,304,396 |
| 0.9 | 50,434,177 | 36,612,604 | 29,284,459 |
| 0.95 | 32,465,363 | 24,242,558 | 21,209,472 |

Again, the situation expressed in percentages can be visualized by a graph.




The graphs show clearly that with a low similarity threshold (0.5), the share of removed texts and tokens is strongly dependent on the value of n-grams. On the other hand, with a "conservative" setting of the threshold (0.9, 0.95) the value of the n-grams has only a limited influence.

In the end we show the frequency distribution of the removed tokens by the length of documents (for 10-grams).



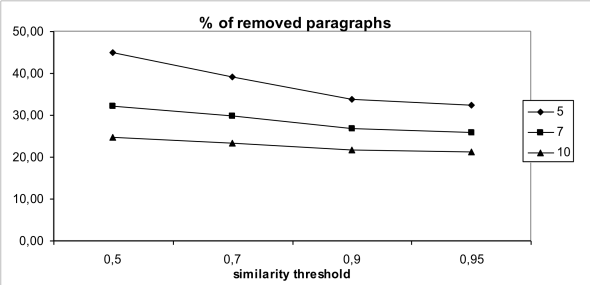

An interesting observation is the rapid increase in the number of deleted tokens with the low frequency threshold within the longer documents. This phenomenon deserves further inspection.

### 3.3 Paragraph-level Deduplication

The second experiment aimed at deduplicating paragraphs was performed with identical settings. Onion was run in the "no smoothing" mode[5]. The following tables report the numbers of removed paragraphs.

| Similarity threshold | 5-grams | 7-grams | 10-grams |
|---:|---:|---:|---:|
| 0.5 | 23,119,807 | 16,541,810 | 12,697,603 |
| 0.7 | 20,164,592 | 15,294,473 | 12,027,316 |
| 0.9 | 17,353,000 | 13,738,966 | 11,187,422 |
| 0.95 | 16,652,531 | 13,285,899 | 10,890,177 |

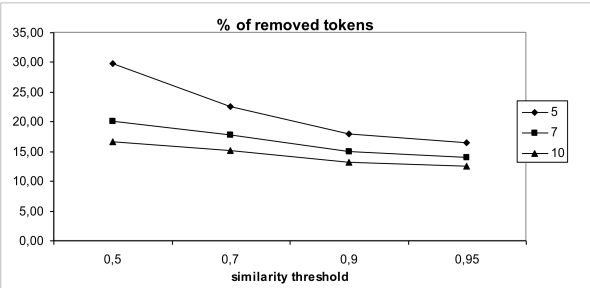
And the graph expressing this in percentages.




The situation with tokens looks like this.

| Similarity threshold | 5-grams | 7-grams | 10-grams |
|---:|---:|---:|---:|
| 0.5 | 364,942,345 | 245,171,349 | 203,061,064 |
| 0.7 | 276,284,030 | 217,137,998 | 184,798,690 |
| 0.9 | 218,857,869 | 184,131,198 | 161,354,010 |
| 0.95 | 201,514,920 | 171,852,183 | 152,246,068 |

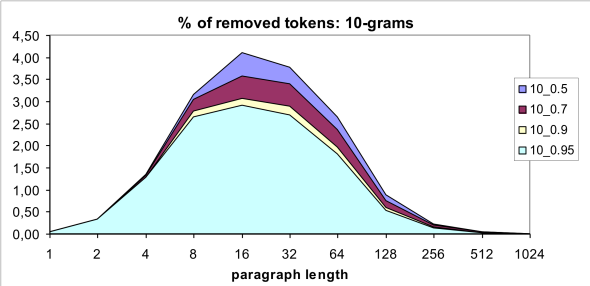
The last graph shows the percentage of removed tokens.




[5] In the "smoothing" mode Onion removes also short non-duplicate paragraphs between two duplicate ones.

We can see that with "aggressive" parameter settings, the deduplication procedure would remove 45% of paragraphs containing 30% of tokens. With the more conservative settings the respective curves approach the 15% level, which is quite similar to the 13.7% achieved by the plain fingerprint method.

We also show the frequency distribution of removed tokens (for 10-grams).

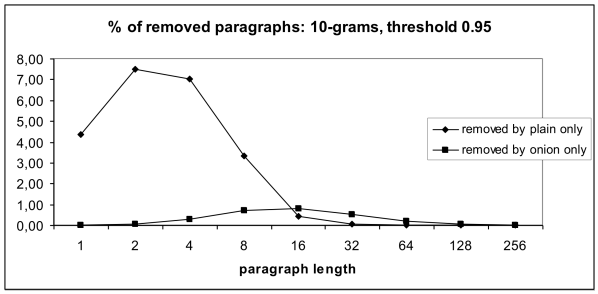


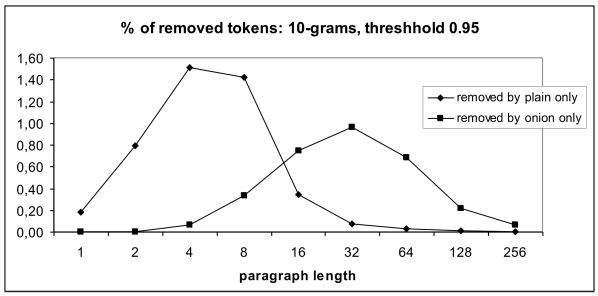

If we compare this graph with the similar one from the plain fingerprint method, we can see that Onion prefers removing paragraphs of medium length where partial match is more likely to happen. Based on these findings we decided that Onion will not be used for deduplication on the sentence level.

### 3.4 What Has not Been Removed by Onion

The last interesting question is which paragraphs were removed by the plain fingerprint method but remained undetected by Onion. Our analysis was performed just with the most conservative values of Onion settings (10-grams, threshold-level of 0.95), where the results were expected to be most similar.

The frequency distributions of the removed paragraphs and tokens are depicted in the following graphs.






We can see that Onion's weak point is the ignorance of duplicates in short paragraphs. According to our analysis, this is mainly caused by the fact that paragraphs shorter than the length of the n-gram are considered duplicate only in the case when the n-grams also match the respective tokens from the end of the previous paragraphs. With the shorter paragraphs, there is also a greater chance of partial match with ignored punctuation and digits implemented in our simple method.

## 4 Conclusion and Further Work

In our paper, we have compared the results of deduplication achieved by two methods – with a plain fingerprint method and by means of Onion. While in detecting exact duplicates the situation is fairly simple, in the detection of near-duplicates there is always a trade-off between the amount of "good" text to be lost and the amount of duplicate contents that will remain in the corpus.

Onion is a very fast and versatile tool that can be conveniently used to detect near-duplicates both at the document and the paragraph level. Its main deficiency is the inability to detect duplicates in short paragraphs. Our suggestion for corpus deduplication is therefore based on a combination of both tools. The whole process would consist of three stages. In the first stage the corpus is deduplicated by Onion at the document level with conservative levels of the parameters (duplicates are removed). In the second stage Onion deduplicates paragraphs (duplicates are marked). And in the last stage the short duplicates are "cleaned" by the plain fingerprint method at the sentence level.

In the future we want to optimize computation of fingerprints in the plain method and apply the results of our research to the whole SNC collection of corpora, as well as to the newly created Slovak web corpus.

## References

[1] Benko, V. (2010). *Optimizing Word Sketches for a large-scale lexicographic project*. Invited lecture. URL: `http://videolectures.net/korpusi2010_benko_ows`.

[2] Rabin, M. O. (1981). *Fingerprinting by Random Polynomials. Center for Research in Computing Technology*. Harvard University. Tech Report TR-CSE-03-01. URL: `http://www.xmailserver.org/rabin.pdf`, retrieved 10 May 2013.

[3] Broder, A. Z. (1993). Some applications of Rabin's fingerprinting method. In *Sequences II: Methods in Communications, Security, and Computer Science*. Springer-Verlag. URL: http://xmail.eye-catcher.com/rabin_apps.pdf, retrieved: 28 April 2013.

[4] Slovak National Corpus – prim-6.0-juls-all. (2013). Bratislava: Ľ. Štúr Institute of Linguistics, Slovak Academy of Sciences. Accessible at: http://korpus.juls.savba.sk.

[5] Pomikálek, J. (2011). *Removing Boilerplate and Duplicate Content from Web Corpora*. Ph.D. Thesis, Faculty of Informatics, Masaryk University in Brno. URL: http://is.muni.cz/th/45523/fi_d/phdthesis.pdf, retrieved 14 June 2012.

[6] Uzgalis, R. (1995). *Random Numbers, Encryption, and Hashing*. Lecture Notes. Computer Science Department, University of Auckland. URL: http://www.serve.net/buz/Notes.1st.year/HTML/C6/rand.012.html, retrieved 20 April 2013.