PLAGIARISM DETECTION IN SLOVAK TEXTS ON THE WEB

Daniela Chudá, Jozef Lačný, Maroš Maršalek, Pavel Michalko, Ján Súkeník

Abstract: Nowadays, access to all kinds of information is greatly simplified. The Web contains enormous amounts of documents available for free. One negative consequence of this is the increase of plagiarism, not only in the fields of research and education, but also in journalism etc. Authors incline to plagiarism for various reasons and the Internet makes it extremely easy. A lot of effort and research have been devoted into developing tools that detect plagiarism on the Web. While the majority of these tools worked correctly during our tests (some had better results than the others), we have found room for improvements. First of all, they are designed to work with English language and therefore their accuracy for texts in more complicated languages (Slovak for example) is not satisfactory. Moreover, the presentation of the results to the user was often very unintuitive and limited.

We have built an application which provides automated plagiarism detection in Slovak texts among the documents on the Web. Our tool utilizes reliable search engines like Google and Bing. The main advantage of this approach is that it does not require building a custom index of the Web which would be time and resource consuming. Additionally, we provide advanced visualization of results and we also offer an option to manually tweak them. The application is available at http://wanted.fiit.stuba.sk.

Introduction

Problems associated with plagiarism started to rise quickly with the development and accessibility of the Internet. Improvements of internet search engines and their widespread popularity influenced the way of retrieving and utilising information sources. Most people nowadays prefer digital information sources available on the Internet.

Why is plagiarism considered to be a problem? We can get our answer by looking at the field of education. Students are the ones who mostly incline towards using the Internet as primary source of information for their work and who often fail to reference their sources properly. Sometimes the reason might just be carelessness or lack of experience. However, the quality of knowledge gained by students who intentionally commit plagiarism is often decreased (Chudá et al., 2012).

Another example is the field of research. When working on a research project, one should rely mostly on his or her own knowledge and experience. However, it is usually helpful or even necessary to use related work from other authors. In these cases it is required to clearly identify all the references so that author’s own contribution can be clearly identified. Plagiarism can also be a problem in many other fields, for example in mass-media (stealing content from each other).

To alleviate negative impacts of plagiarism it is necessary to be able to discover it. That means comparing all the new works to every paper, book or article on similar or overlapping topic available not only in the libraries but also on the Internet. When trying to detect plagiarism manually, the effort required is simply too big and can even
be multiplied if the author tried to disguise it. It is therefore desirable to fully automate the process of measuring similarity between documents.

We have designed and developed a tool that deals with this issue. Our application provides automatic similarity evaluations of submitted document against related documents available on the Internet. We have focused on processing texts in Slovak language, since there are almost no tools optimised for languages other than English. Our main goals are improving quality in education, boosting creativity of students and motivating journalists to create original content. Indirectly, our tool can provide help when trying to find related information sources. We faced several problems in order to create the application and our approach is presented in this paper.

In the paper, we firstly take a quick look at the existing research and tools. We continue with the high-level overview of our method and consequently describe individual steps of the process in greater detail. At the end we outline our experiments and discuss possible use cases of our application.

**Related work**

We divide the problem of plagiarism detection in Slovak language into multiple smaller problems.

First of all it is necessary to extract raw text from different document formats. Some of them contain a lot of formatting information (DOC, DOCX), others are designed for printing (PDF) which means that the formatting information even takes precedence over the text information and its semantics (titles, captions, etc.). Extraction of raw text from HTML documents is even more complicated because of all the clutter that websites contain (navigation, headers and footers, advertisements, etc.). There has been done a lot of research in this area (Nguyen Minh Trung et al., 2011), (Weninger, 2010) and many tools were implemented (The Apache Software Foundation, 2013).

To make text processing easier and results more accurate, the input document should be divided into smaller parts. Multiple ways of achieving this are outlined in (Pataki, 2003).

Another part of text pre-processing especially important for Slovak language is the lemmatization. Slovak Academy of Sciences provides morphological database which contains 2.5 million words (approximately 75000 different lemmas) and can be used for this purpose.

Very important problem is building of search queries. These queries must reflect the source text so that the same or similar texts will be found during the web-search. This is the crucial step when trying to find related documents on the Internet (Butakov and Shcherbinin, 2009).

The core of plagiarism detection is comparison of texts or documents. Lots of approaches for performing this task were suggested, each with its own advantages and disadvantages (Genči et al., 2009). As examples we choose methods based on n-grams (Češka, 2007), longest common substrings (Chong et al., 2010) or stylometry (Clough, 2000).

The research in the field of plagiarism detection has resulted in a big number of plagiarism detection tools. We divide them into two groups: (i) the tools that use
Table 1  
Comparison of some of the tools that detect plagiarism on the Web

<table>
<thead>
<tr>
<th></th>
<th>Turnitin</th>
<th>EVE2</th>
<th>DOC Cop</th>
<th>PlagScan</th>
<th>Copyscape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>subset/no</td>
</tr>
<tr>
<td>Detection of citations</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Advanced support for Slovak language</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Detection of anti-plagiarism</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Support for local corpus</td>
<td>yes</td>
<td>no</td>
<td>limited</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Web-search</td>
<td>custom</td>
<td>plugins</td>
<td>Bing</td>
<td>Yahoo!</td>
<td>Google, Yahoo!</td>
</tr>
<tr>
<td>Another limitations</td>
<td>known methods of cheating</td>
<td>platform dependency</td>
<td>input text limit</td>
<td>lower effectiveness</td>
<td>designed for websites</td>
</tr>
</tbody>
</table>

only local corpus of documents, (2) the tools that use more general, shall we even say global corpus—the Web. An example from the first group is CRZP (central repository of final theses). CRZP is used in Slovak Republic to ensure that the theses (bachelor, diploma, etc.) do not contain plagiarism. CRZP only detects plagiarism in local corpus of documents primarily made of all the theses ever submitted to it.

One of the tools that are capable of detecting a plagiarism on the Web is Turnitin (iParadigms, LLC., 2013). Main advantages of Turnitin are extensive local corpus of documents which includes documents from the Web, detailed results presentation and visualization, support for quotations of external sources, constant development and improvements. The biggest limitations of Turnitin are lack of advanced support for Slovak language and publicly available methods of cheating.

Other similar tools are: EVE2 (Essay Verification Engine) (CaNexus, 2000), DOC Cop (McCrohon, 2013), PlagScan (Resonet UG, 2013) or Copyscape (Indigo Stream Technologies, 2013). We summarize comparison of all mentioned tools in table 1. These tools and many others are compared more extensively in (Bull, 2001) or (Plagiats Portal, 2012).

All of the tools mentioned above share one common disadvantage. They can process texts in different languages, but (to our knowledge) none of them considers specifics of those languages. This is particularly important when dealing with partially modified copied text. Using stop-words removal, performing lemmatization etc. can yield more accurate results and more precise plagiarism detection.

Method description

In this chapter we present our own solution for plagiarism detection on the Web. We have designed a unique process that deals with this problem and one of most important characteristics of our solution is the utilization of already available Internet search engines like Google and Bing.

As we have mentioned before, the problem we are dealing includes many sub-problems which had to be solved. Therefore we have split this process into several steps.
For each of these steps we have designed an approach which enables us to achieve best performance and most accurate results.

The six steps of the whole plagiarism detection process (see figure 1):

1. pre-processing and chunking of input document,
2. construction of search queries,
3. web-search for related documents or possible sources of plagiarism,
4. plagiarism detection,
5. calculation of similarity level for the entire document,
6. visualization of the results.

These steps are described in detail in the following sections.

**Text pre-processing**

After the user submits a document to our application, we perform several steps of pre-processing before the analysis can be started. Firstly, we have to extract raw text from the document. We have experimented with multiple tools or libraries and at the end
we have settled with Apache Tika for formats such as PDF, DOC, DOCX or pure text and Readability for HTML documents.

In the next step we split the raw text into smaller parts—chunks. Processing of these chunks is more suitable for plagiarism detection, since only few sentences are usually copied. Moreover, different parts of text can be copied from different sources.

**Query building**

The first major issue we needed to solve was how can we possibly compare a submitted document with every document on the Web? This is clearly not possible. In fact we can only compare input document with very small subset of all the documents available on the Web. So the question is immediately changed to how can we obtain this subset of documents so that it will (with reasonably high probability) contain the documents we are searching for? The documents in this subset will probably have the same topic as the input document, which means they will contain similar keywords. This means that we need a full-text index of the Web and some search engine built on top of this index.

From this emerged the next question: how should we use such search engine? If we obtain keywords from the whole input document and use those as the search query, the probability of finding correct results (documents used as the source of plagiarism) is rather low. In fact, the longer the input document, the lower is the probability. On the other hand, we can use every sentence as the search query, or more general, every n-gram of the input document. This way it would be possible to detect plagiarism almost always. The number of web-searches would however be very high, the subset of documents to examine would be very big and the whole process would take a lot of time. Therefore we needed to find a solution between these two extremes.

In order to design a good query building algorithm, we needed to know how search engines work. We have done experiments with Google and Bing. Results of one experiment are summarized in table 2. Since we focus on documents in Slovak language, we decided not to translate sample queries.

We have obtained few very important conclusions from our experiments:

- query does not need to contain consecutive words,
- order of the words in the query does not matter at all,
- query does not need to contain many words, few are usually enough,
- query should not contain any words that are not present in the original document,
- query should not contain any words that were partially changed (prefixes, postfixes or any other modifications of words as a result of inflection or conjugation),
- if correct result is found, it is always in top results (usually first, but we have seen second or third).

Results of our experiments suggest that the query can be built from document’s keywords, but with two big and important constraints. Selected keywords have to be exactly the same in the original document as in the plagiarized document, no modifications are allowed at all. That is rather unfortunate when considering that we
Table 2  
*Manual experiments performed to find best automatic query building method*

<table>
<thead>
<tr>
<th>Description</th>
<th>Search query</th>
<th>Result count</th>
<th>Position of the original article in the results</th>
</tr>
</thead>
<tbody>
<tr>
<td>- initial query</td>
<td>čierne diery považované nutné pozerat hľadiska mimoriadne dôležitú úlohu vzniku galaxií</td>
<td>3 132</td>
<td>Bing 1 Google 1</td>
</tr>
<tr>
<td>- as no. 1</td>
<td>čierne diery považovali treba pozerat hľadiska mimoriadne podstatnú úlohu vzniku galaxií</td>
<td>1 113</td>
<td>Θ Θ</td>
</tr>
<tr>
<td>- changed “považované” → “považovali”</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- changed “nutné” → “treba”</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- changed “dôležitú” → “podstatnú”</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- as no. 2</td>
<td>čierne diery považovali pozerať hľadiska mimoriadne podstatnú úlohu vzniku galaxií</td>
<td>1 113</td>
<td>Θ Θ</td>
</tr>
<tr>
<td>- removed word “treba”</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- as no. 3</td>
<td>čierne diery považovali pozerať hľadiska mimoriadne úlohu vzniku galaxií</td>
<td>8 125</td>
<td>Θ Θ</td>
</tr>
<tr>
<td>- removed word “podstatnú”</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- as no. 4</td>
<td>čierne diery pozerať hľadiska mimoriadne úlohu vzniku galaxií</td>
<td>102 128</td>
<td>1 1</td>
</tr>
<tr>
<td>- removed word “považovali”</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- the query is equal to no. 1 with three words removed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- as no. 5</td>
<td>čierne diery pozrieme hľadiska mimoriadne úlohu vzniku galaxií</td>
<td>8 1080</td>
<td>Θ Θ</td>
</tr>
<tr>
<td>- changed “pozerat” → “pozrieme”</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- as no. 6</td>
<td>mimoriadne úlohu vzniku galaxi čierne diery pozerať hľadiska</td>
<td>96 1280</td>
<td>1 1</td>
</tr>
<tr>
<td>- changed order of words</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- short query, removed majority of words</td>
<td>čierne diery vesmírne pasce</td>
<td>15 6400</td>
<td>1 1</td>
</tr>
<tr>
<td>- as no. 8</td>
<td>čierne diery vesmírne pasce astronómovia</td>
<td>141 101</td>
<td>1 1</td>
</tr>
<tr>
<td>- added word “astronómovia” (part of original article)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- as no. 9</td>
<td>čierne diery vesmírne pasce astronómovia pohľadu</td>
<td>73 4150</td>
<td>Θ 2</td>
</tr>
<tr>
<td>- added word “pohľadu” (not part of original article)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- another query</td>
<td>považované pasce stiahnu stiahnu všetko nebezpečne astronómovia čierne nutné úplne mimoriadne</td>
<td>1 7</td>
<td>1 Θ</td>
</tr>
<tr>
<td>- duplicated word “stiahnut”</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

deal with an inflectional language. Secondly, if only as little as one word in a query is not present in the original document, search engine will probably not find it.

As we stated earlier, we divide the input document into small chunks. For each chunk we build one search query. Query contains few words from chunk (less than 10) which have the high probability of satisfying the two constraints mentioned above. This query
is the core of our method and allows us to find documents which contain similar or exactly the same text as the text in the chunk.

We have designed multiple methods for query building, such as:

- facts extraction—numbers, dates, names, in other words parts of text that are almost impossible to modify without changing the meaning of the text,
- nouns selection—although not impossible, nouns can also be difficult to modify,
- most frequent words—simple keywords of individual chunks,
- TF-IDF based selection of keywords—regular TF-IDF uses whole documents and returns keywords for one document; we use chunks and we are interested in keywords for one chunk,
- random selection,
- combinations of the above.

We performed several tests to determine the best method (see table 3). The primary method which we currently use is combination of facts, nouns and most frequent words.

Web-search

Next step of the process is the web-search. We do not have enough resources to create our own index of the Web and search engine. It is incredibly complex topic and not our focus at all. Moreover, there are companies, organizations or other initiatives that already have web-search engines running, so we decided to outsource this part of the process. The two major players in this area are currently Google and Microsoft with their Bing. We decided to support both of them. Currently we do not utilize digital libraries (e.g. ACM, IEEE) because they do not provide free access. We rely solely on the results from web-search engines.

At this point of the overall process we have the input document divided into chunks and search query built from each chunk. Our application now starts performing web-searches using all the available queries (that means one web-search per chunk). The third conclusion derived from the experiments described in section about query building is that it is sufficient to evaluate plagiarism in only few top results. Therefore when we receive results of the web-search, we only download first 5 documents.

Similarity detection

So far we have divided input document into chunks, created search queries and downloaded search results for each available chunk. During this step we pre-process documents acquired from the Internet (just like the input document but with the exception of chunking). We also optionally perform lemmatization on both (1) the text of the chunk as well as (2) the text of the document from the Web.

After this additional processing we calculate similarity by comparing the n-grams (on the word level) of the chunk with the n-grams of search result. The outcomes of this comparison are (1) the levels of similarity between a chunk and the respective web document and (2) the list of identical n-grams.
Similarity level calculation

In this step we merge partial results obtained in the previous step. The result of this process is one number that represents final level of plagiarism of the whole input document. During this process we had to solve some typical problems, for example the same text being found in multiple web documents, or one chunk being copied from two non-overlapping sources.

Visualization

Our main contribution lies in the first five parts of document processing, nevertheless we consider proper and user friendly visualization of results to be very important for this kind of application. We have built the user interface that tries to preserve balance between user-friendliness, ease-of-use and proper visualization of all important results of document analysis.

One of the problems here was correct highlighting of copied parts of the input document. As we mentioned earlier, the documents are being modified during the analysis (removing whitespace, special characters, stop words or lemmatization). Because of these modifications it is difficult to correctly highlight copied parts of the original (not modified) text for the purpose of presentation to the user. The documents may moreover sometimes contain parts of copied text that are in fact correctly cited or false positives. For these cases our application offers an option to mark every detected positive text matching as correctly cited. After this operation, the selected text is excluded from plagiarism evaluation and results are recalculated and refreshed in real-time so that the user can always see the most up-to-date information.

Architecture and technology

As we mentioned earlier, we have developed an application that implements described solution. The architecture of the application conforms to standard client-server model with business and data layers located on a centralised server and presentation layer running in user’s web browser as a thin client.

The business layer is implemented in Ruby on Rails and provides the core functionality of our solution through RESTful web services. This implementation allows clear separation between business and presentation layer and the functionality of our solution can be used by multiple different clients.

To ensure the best possible user experience we have chosen HTML5 to be our main technology for presentation layer development. Our goal was to provide intuitive and easy to use graphical interface.

Experiments

The query building process plays the key role in our solution. We have spent a lot of time designing query building methods as well as analysing their performance. In one experiment we analysed plagiarism level of a set of artificially prepared documents using different query building methods and search engines. These documents contained
Table 3
Comparison of query building methods using different search engines

<table>
<thead>
<tr>
<th></th>
<th>random</th>
<th>facts + nouns + most frequent words</th>
<th>facts + nouns + most freq. words + random</th>
<th>facts + nouns + tfidf + typos</th>
<th>expected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Google</td>
<td>Bing</td>
<td>Google</td>
<td>Bing</td>
<td>Google</td>
</tr>
<tr>
<td>1.</td>
<td>87%</td>
<td>30%</td>
<td>87%</td>
<td>49%</td>
<td>79%</td>
</tr>
<tr>
<td>2.</td>
<td>95%</td>
<td>79%</td>
<td>100%</td>
<td>84%</td>
<td>97%</td>
</tr>
<tr>
<td>3.</td>
<td>45%</td>
<td>27%</td>
<td>37%</td>
<td>26%</td>
<td>21%</td>
</tr>
<tr>
<td>4.</td>
<td>21%</td>
<td>1%</td>
<td>42%</td>
<td>0.01%</td>
<td>51%</td>
</tr>
</tbody>
</table>

text copied from one or more sources available on the Internet. This experiment shows that our query building methods are in fact able to find correct documents.

Table 3 shows the results of the experiment. There are the three most significant query building methods compared to the random query building method as well as the expected results.

The expected result is the real plagiarism level of an artificially prepared plagiarized document. The results show that the best query building method for finding relevant documents on the Internet is combination of facts, nouns and most frequent words. We also found that search for similar documents using Google is much preferable compared to Bing when using queries in Slovak language.

Conclusions

The result of our work is a tool whose primary goal is to help with plagiarism detection. We took advantage of current research and performed necessary experiments in order to make this tool efficient. Thanks to our solution the whole process of document analysis is performed only by just a few simple user interactions.

Comparison to domain expert

The human judges are very good at determining whether some particular part of the text is copied from some specific source. But as soon as they try to find those sources themselves, various problems come up. First of all, such plagiarism evaluations may be different each time. Manual searches and manual comparisons can also be very slow and inaccurate, since the number of possible sources (even about one specific topic) is usually big and one person cannot possibly be familiar with all of them. Walenstein et al. (Walenstein, et al., 2003) addressed similar issue in their work with source code clones. They conclude that single judge cannot be trusted to give unbiased answers (they actually used multiple judges who always had to go through consensus-building process). Although Walenstein et al. focused on different problem, we believe that their conclusion can be generalized. A domain expert’s opinion can therefore be considered helpful at best, not obligatory.
Secondly, we have never intended our tool to replace the domain experts or to compete with them. We believe that no matter how good the automatic plagiarism detection is, it will never be possible to completely leave out the final judgement of the domain expert. Our goal is just to simplify and automate the repetitive tasks which are part of the plagiarism detection process. We hope that it will make the whole process not only easier and faster, but also more objective and accurate. Purpose of our tool is therefore to support the domain experts and to minimize the impact of all the disadvantages of manual plagiarism detection.

Use cases of an application

We have identified several use cases for our application. The main use case is of course plagiarism detection where a typical scenario would be a teacher checking assignments of his or her students. Besides plagiarism detection, we think it would also be helpful if students or researchers could check their papers before submitting them and therefore had a chance to correct the references to other sources in case of any mistakes.

Another interesting use case is effort analysis. If we look at the elementary or high schools, it is not unusual that the teachers want the students to write papers which consist exclusively from the work of other authors. In such cases it is difficult and time consuming to find out whether a student copied the text from one or two sources or if the student found multiple interesting sources and combined them in a creative way. The teacher could use our application to find the list of sources used in such papers and consider this during paper evaluation.

In its core, our application tries to find documents which address the same topic as the input document. This feature is just one part of our plagiarism detection solution, but it could be utilised on its own. The goal would be to discover new sources of information on certain topic that were yet unknown to the user. Such feature could for example find additional literature just by submitting an unfinished paper. Despite the fact that our method is capable of providing this feature, our user interface, in its current version, is not adjusted for it. We are however planning to update the application because we think that the ability to conduct literature review on a topic would be useful.

Other languages

This application was developed mainly for processing texts written in Slovak language, but it can be used with different languages too at the cost of slightly lower precision. On the other hand, if the language dependent parts of application will be replaced (e.g. normalisation or lemmatization modules) the final results would be as precise as in case of the Slovak texts.

Our application is available at http://wanted.fiit.stuba.sk and we encourage everybody to try it out.
References


Clough, P. (2000). Plagiarism in natural and programming languages: an overview of current tools and technologies Keywords. 1–31


Tim Weninger, William H. Hsu & Jiawei Han (2010) CETR: content extraction via tag ratios. In Proc. of the 19th international conference on World wide web (WWW ’10). ACM, New York, USA, 971–980


Acknowledgement

This work was partially supported by the Scientific Grant Agency of the Slovak Republic, grant No. VG1/0971/11 and is the partial result of the Research & Development Operational Prog. for the project ITMS 26240220039, co-funded by the ERDF.
Authors

Daniela Chudá, chuda@fiit.stuba.sk,
Jozef Lačný, lacny.jozef@gmail.com,
Maroš Maršalek, maros.mars@gmail.com,
Pavel Michalko, michalko.pavel@gmail.com,
Ján Súkeník, sukenik08@student.fiit.stuba.sk
Slovak University of Technology, Faculty of Informatics and Information Technologies, Slovakia

Copyright © 2013 Chudá, D., Lačný, J., Maršalek, M., Michalko, P., Súkeník, J.: The authors grant to the IPPHEAE 2013 Conference organisers and educational non-profit institutions a non-exclusive licence to use this document for personal use and in courses of instruction provided that the article is used in full and this copyright statement is reproduced. The authors also grant a non-exclusive licence to Mendel University in Brno, Czech Republic, to publish this document in full on the World Wide Web (prime sites and mirrors) on flash memory drive and in printed form within the IPPHEAE 2013 conference proceedings. Any other usage is prohibited without the express permission of the authors.