



FACULTY OF INFORMATICS
MASARYK UNIVERSITY

Systems for Discovering Similar Documents

Jan Kasprzak

Ph.D. Thesis Proposal
Brno, September 3, 2009

Advisor: *doc. Ing. Michal Brandejs, CSc.*

Advisor's signature

Contents

1	Introduction	3
1.1	Similar Documents	3
1.2	Plagiarism	4
1.3	E-learning Systems Integration	5
2	State of the Art	6
2.1	Implementations	6
2.2	Evaluation of the Plagiarism Detection Methods	6
2.3	Classification of Anti-Plagiarism Approaches	7
2.4	Detecting the External Plagiarism	8
2.5	Chunking Approach	8
2.6	Constructing the chunks	9
2.7	Fingerprints	10
2.8	Similarity Measure	10
2.9	Data Structures	11
2.10	Postprocessing	12
2.11	Valid Intervals	12
2.12	Distributed Computing	13
3	Thesis Goals	15
3.1	Objectives and Expected Results	15
3.2	Schedule	16
4	Results of Study	17
4.1	On Topic of the Ph.D. Thesis	17
4.2	Academic Work	17
4.3	Other Results	18
5	Summary/Souhrn	19
5.1	Summary	19
5.2	Souhrn	19
6	References	20

1 Introduction

With a more and more widespread availability of documents in an electronic form, the task of *discovering similar documents* has become increasingly important. There is a wide area of applications of this task, from evaluation of students' work to the crawling the World-Wide-Web. In my Ph.D. thesis, I want to focus on some aspects of this task, especially on efficiently discovering copied passages in a large set of documents.

*discovering
similar
documents*

1.1 Similar Documents

The criteria of the document similarity are different depending on the purpose of the evaluation. For some purposes it is sufficient to know that the pair of documents focus on the same topic, while for other purposes we need to know whether there are word-by-word copied passages and possibly how many per cent of the whole document is covered by those copied passages.

Some of the common applications of discovering similar documents are the following:

Removing duplicate copies

In the large sets of documents there are often present duplicates which can skew the results (such as statistical data or full-text search results) computed from this set of documents. Thus it is desirable to detect and remove duplicate copies of one document.

An example of duplicate documents is an article published simultaneously in more sections of a news WWW site or available under more different URLs of the same WWW server.

A full-text search engine may want to remove the exact duplicates from the indexing altogether.

For example, the full-text search application of Masaryk University WWW server [MU09] presents four identical pages (with different URLs) as the top-most results for the query term "*rektor*". The discovery of duplicates would greatly help to improve the results here.

A specialized algorithm for removing duplicates has been discussed for example in [PR08].

Finding related documents

The documents are considered being *related documents* in a broad sense of this word when they focus on the same or similar topic. In the narrower sense, the related documents can be for example different versions of the same document taken from the version control system.

*related
documents*

The ability to find related documents can be also useful for the full-text search engine: the search engine may want to hide related documents having the same or similar source from the user, in order to present more meaningful results.

The task of finding derivative documents has been discussed in [BZ04].

Detecting plagiarism

This is a problem which I will focus on, so I will describe it in a detailed way in the next section.

1.2 Plagiarism

Encyclopædia Britannica defines plagiarism as “*the act of taking the writings of another person and passing them off as one’s own.*” [Bri09]. There are many different forms of plagiarism:

Copying the ideas

In this form the plagiarist presents other person’s ideas as his own. Most often the plagiarist uses his own wording to present the idea so the machine detection of this kind of plagiarism is not possible without the detailed semantic analysis of the text.

Copying the text without citing the source

This form of plagiarism is relatively easy to discover by computer-assisted approaches, even though some plagiarists try to obfuscate the original text to some extent: a common technique is to replace words by synonyms or change the word ordering (provided that the language in question allows at least some reordering without the result sounding weird).

However, while the discovering of copied passages is not hard, it is often hard to decide whether the copied passage has been plagiarized (and if so, which document is the original one). The main problem here is to be able to tell apart the correct citation from the plagiarized text with vague citation or no citation of the source at all.

As for deciding which document is the original one, the publishing date of the document can help a bit. But in some cases even this is not enough: often the publishing process can take quite a long time, during which preliminary versions of the text can be published elsewhere (for example, publishing the full/extended version of the research paper as a local technical report in parallel to publishing the paper in a journal or in conference proceedings).

Buying the work and using it as one’s own

There are various services which offer essentially to write the seminar work or thesis for a fee (often this kind of service is masqueraded as “providing groundwork or supporting materials for the thesis”).

Translating the text

This is another common case of plagiarism. This can be relatively hard to detect, depending of how much the two languages in question differ in their structure. On the other hand, detecting translated passages of the text can be easy for a pair of similar languages, like Czech and Slovak.

Copying the document structure

This is often used by students when copying laboratory protocols or other text with experimental data. These documents contain relatively small amount of text and big amount of numbers, which the plagiarist

can only slightly modify, and use the work as his own result of supposedly experimental work.

Reusing own texts

Plagiarizing one's own texts can seem strange at first, but the problem can be when the author presents his own already published idea as a new result. A well-known local case of this form of plagiarism has been published in local media [iDn08].

1.3 E-learning Systems Integration

Plagiarism detection software can be a valuable part of an e-learning system. The tight integration with this system provides better usability of the system to the users. Also when it becomes widely known that the e-learning system contains also the anti-plagiarism software, it results in better works submitted by students.

The Masaryk University Information System [IS 09] has a full-featured subsystem for e-learning. The e-learning subsystem includes since August 2006 a system for discovering similar documents. This system is usable for computer-assisted discovery of plagiarism. The system currently has a base of over 1,300,000 documents, indexed and searched for similarity. This number includes also data from the Czech National Archive of Graduate Theses [The09] and also from the new project Odevzdej [Ode09], which all share the same anti-plagiarism subsystem.

IS MU

*theses.cz
odevzdej.cz*

I have been directly involved in the design and implementation of the anti-plagiarism system used by Masaryk University in its Information System. This anti-plagiarism system serves both as a testbed for the research of systems for discovering similar documents, and as a proof that the developed algorithms and approaches are in fact usable in a large-scale distributed system.

2 State of the Art

2.1 Implementations

The discovery of similar documents has been implemented in several real-world and proof-of-concept systems:

Turnitin

Turnitin.com [Tur09] is probably the most widely used system for discovering similar documents. It stores the documents in an internal database, compares the new submissions against this database, and also adds the new submissions to this database. This is a proprietary system, I have not been able to find any paper describing its implementation.

Copyscape

Copyscape [Cop09] is the system for finding similar documents in the WWW. The system tries to extract characteristic phrases from the document, and use a general full-text search engine—Google—for searching the duplicates.

DocCop

DocCop [Doc09] is an Australian system for finding document overlap. It can either annotate the overlap in the two given documents, or can find overlaps using Google API. It can handle even Microsoft Word (.doc) and PDF files.

CoPS

An experimental system, described in [BDGM95]. Their approach is to find common sentences in the set of documents.

SCAM

Another experimental system, described in [SGM95]. The authors use the word frequencies from the documents to compute the similarity of the documents.

2.2 Evaluation of the Plagiarism Detection Methods

There are several commonly used criteria for estimating the quality of plagiarism detection methods.

Recall

Let S be a set of all plagiarized passages in a given set of documents, $s_i \in S$ be a particular plagiarized passage and $d(s_i)$ be a number of characters from s_i , detected by a particular detection algorithm. Then we define the *recall* of this method as follows:

recall

$$\text{Recall} = \frac{1}{|S|} \sum_{i=1}^{|S|} \frac{d(s_i)}{|s_i|}$$

The recall of an algorithm is a measure of how much of a plagiarized text passes unnoticed by the algorithm.

Precision

Let R be a set of all detections produced by the measured algorithm, $r_j \in R$ be a particular detected passage, and $p(r_j)$ be a number of characters of that passage, which are in fact plagiarized (i.e. they belong to some s_i from the previous paragraph for some i). Then we define the *precision* of the algorithm as follows:

$$\text{Precision} = \frac{1}{|R|} \sum_{j=1}^{|R|} \frac{p(r_j)}{|r_j|}$$

The precision of an algorithm is a measure of how many false positives the algorithm produces.

Granularity

Let $S_R \subset S$ be a subset of S , where each $s_i \in S_R$ overlaps with at least one $r_j \in R$ at least by one character. Then we define the *granularity* of an algorithm as follows:

$$\text{Granularity} = \log_2 \left(1 + \frac{1}{|S_R|} \sum_{i=1}^{|S_R|} |\{j | s_i \cap r_j \neq \emptyset\}| \right)$$

The granularity is a measure of how good the algorithm is in discovering larger plagiarized but obfuscated passages as single entities instead of many small word-by-word copied fragments. For human post-processing of the algorithm output it is usually better to know that “*this large passage has been roughly copied from this document*” instead of “*these small passages are exact copies from there*”.

Security

The *security* of an algorithm on a source document D is defined as a number of words in D that have to be deleted, inserted, or modified, in order to make the algorithm not discover the resulting document as a plagiarized passage. Some authors define the security measure as a number of characters instead of words [BDGM95]. Security is a measure of resistance of an algorithm against the attack where the attacker knows the inner workings of the algorithm and actively tries to sabotage it. For example, the algorithm with security of 1 can discover only the exact copies. The higher this number is, the more secure the algorithm is against targeted attack.

When evaluating the algorithm for the detection of plagiarism and/or document overlap, it is common to use the combination of the above measure. For example, the organizers of the First International Competition on Plagiarism Detection used the harmonic mean of precision and recall, divided by the granularity [PAN09].

2.3 Classification of Anti-Plagiarism Approaches

The methods for detecting plagiarism can be divided into two main classes. The first one tries to match a (part of a) *suspicious document* to one or

more text passages from the big set (corpus) of the *source documents*. In the extreme case, the set of the source documents can be the whole world-wide web, and matching the suspicious document can be done by means of automated querying a WWW search engine. Detecting plagiarism by finding the source document is called detecting the *external plagiarism*.

*external
plagiarism*

On the other hand, the research is being done on finding plagiarism inside the document itself, for example by detecting changes in the writing style. This approach is called detecting the *intrinsic plagiarism*.

*intrinsic
plagiarism*

Because for our systems, maintaining the large database of source documents and probably even querying the general full-text search engines is entirely feasible, I will focus more on the methods of detecting the external plagiarism.

As for the intrinsic plagiarism, the approaches used there are methods using various text statistics (N-Gram based categorization, as described in [CT94], would be a feasible approach here), syntactic features, text structural features (like formatting the text), etc. A brief classification of intrinsic plagiarism approaches has been done by Meyer and Stein in [zES06].

2.4 Detecting the External Plagiarism

There are several methods for detecting external plagiarism and/or document overlap. Brin in the COPS system proposes splitting the document to sentences, storing fingerprints (e.g. hash values) of the sentences, and comparing only those fingerprints [BDGM95].

Shivakumar in the SCAM system proposes using the word frequencies as a similarity model and computes the document similarity from the vector of the word frequencies. He claims to have even better results than COPS, but their data set is relatively small to which we have to handle now [SGM95]. This approach is similar to general information retrieval methods.

Heinze in the Koala systems tries to solve the scalability of the system upto millions of documents [Hei96]. His approach is to store only very small fingerprints of the parts of the document (several hundred bytes per document at most), thus trading the scalability for accuracy.

2.5 Chunking Approach

Many of the methods for finding the similar documents can be generalized as a chunking approach. With this approach, the document is split into words, which then form possibly overlapping *chunks*, and the system then stores and compares only fingerprints of those chunks. The general form of this algorithm is outlined here:

chunks

1. For each source document, do
 - (a) Split the document into separate words
 - (b) Generate chunks of several consecutive words (in the extreme case the whole sentences or paragraphs)

- (c) Use a hash function to generate the fingerprint of each chunk
 - (d) Store the (fingerprint, document ID) pairs to the database
2. For the suspicious document, do
- (a) Generate chunks of this document and their fingerprints, as in steps 1a-1c.
 - (b) Look up those fingerprints in the database, generated in step 1d.
 - (c) When the source document with more than a certain threshold of n common fingerprints is found, consider it a similar document to this suspicious document.

2.6 Constructing the chunks

Different authors use different methods of constructing the chunks. As an example, let us have the document with a single sentence *“If you want the holes in your knowledge showing up try teaching someone.”* (from an interview with Alan Cox on Slashdot [Cox02]). When using 10-word chunks, the document could be split into the following chunks:

- if you want the holes in your knowledge showing up
- you want the holes in your knowledge showing up try
- want the holes in your knowledge showing up try teaching
- the holes in your knowledge showing up try teaching someone

Another approach is to use the whole sentences as chunks, which would in our case lead to a single chunk covering the whole document.

Yet another difference can be whether to use overlapping chunks or non-overlapping, starting from the beginning of the sentence. For our document, the following five-word non-overlapping chunks can be constructed:

- if you want the holes
- in your knowledge showing up

Note how this approach does not cover the whole sentence.

Another difference might be whether the system stores all the chunks, or some subset of them.

The performance of various chunking methods are compared in [MFZ⁺02].

2.7 Fingerprints

Generating the fingerprints from chunks can be done in different ways. The most common approach is to use a hash function with good statistical properties (such as equal distribution). One of the frequently used hash functions in the chunk fingerprinting is MD5 [Riv92]. This function however has a relatively large value space (it produces 128 bits of digest data from any input), so the common approach is to use only the first n bits of the MD5 digest.

2.8 Similarity Measure

similarity

When the pair of documents is flagged by the algorithm described in section 2.5 as similar, it is desirable to know the amount of *similarity* between the two documents. Monostori et al define three different types of similarity [MFZ⁺02]:

Asymmetric similarity

Let $d(F)$ denote the set of fingerprints of the chunks of the document F . The *asymmetric similarity* of the document F to the document G can be defined as

$$a(F, G) = \frac{|d(F) \cap d(G)|}{|d(F)|}$$

This kind of similarity is indeed asymmetric, and depends highly of the size of both documents. For example, when the smaller document F is as a whole contained in the larger document G , it has 100 % asymmetric similarity to the document G . However, the reverse is not true: the asymmetric similarity of the document G to the document F is much smaller.

Symmetric similarity

The *symmetric similarity* can be defined as

$$s(F, G) = \frac{|d(F) \cap d(G)|}{|d(F)| + |d(G)|}$$

The symmetric similarity can be useful when the task is to evaluate both the document F and G .

Global similarity

And finally, the *global similarity* of the document F to the set of documents \mathbb{G} can be defined as follows:

$$g(F, \mathbb{G}) = \frac{|d(F) \cap (\bigcup_{G \in \mathbb{G}} d(G))|}{|d(F)|}$$

The global similarity is useful for evaluating the overlap of the suspicious document with the set of source documents.

The most useful measure of similarity is the first one (asymmetric), but it can depend on the workflow: for example, in the `theses.cz` system, the user selects the document (the reviewer selects the thesis he has to review), and wants to know to which other documents it is similar. The usage of asymmetric similarity here is quite natural.

2.9 Data Structures

The chunking algorithm needs only one simple data structure: mapping the chunk ID (fingerprint value) to the list of documents, which contain the chunk with the same fingerprint. Finkel et al use the BerkeleyDB hash table tied to the Perl hash variable [FZMS02], but the table can grow relatively fast out of the available memory, should the algorithm decide to store all the fingerprints, not only the selected ones (thus aiming to the highest possible *recall* value).

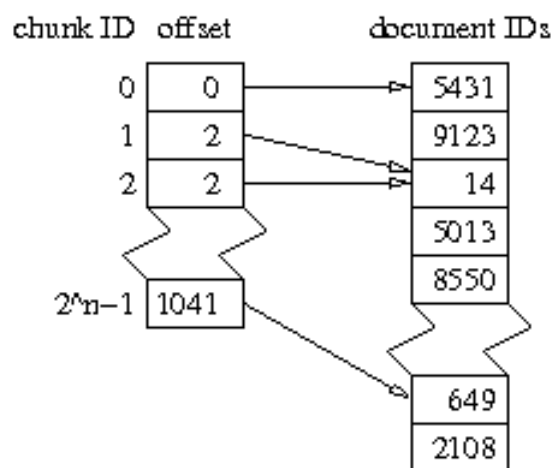


Figure 1: The data structure mapping chunk ID to the list of document IDs.

In our experiments we have decided to use the size of the fingerprint that is proportional to the expected number of chunks stored. For example, for about 2^{28} to 2^{29} chunks we have used 30-bit fingerprints. This lowers the effect of hash table collisions, but on the other hand keeps the data structure relatively dense, so it is feasible to use an array for storing the fingerprints, thus allowing the extremely fast access times.

The data structure we have used is described in the figure 1. It is an array mapping the fingerprint (chunk ID) to the list of the document IDs in another file. Both files are accessible by the `mmap()` system call, so they can be used quite fast even without the need to read them to the RAM first, and with read-only mapping, they can be even sharded in memory between more processes doing the lookup in this data structure.

2.10 Postprocessing

One of the possible drawbacks of the fingerprinting approach is the usage of the hash function, which can produce hash collisions (i.e. two or more different chunks map to the same fingerprint), and thus generate false positives (in other words, lower the *precision*, as defined in section 2.2). Finkel et al [FZMS02] try to mitigate the problem by using relatively large length of the fingerprint (they take upper 10 hex digits of the MD5 hash, i.e. 40 bits). The storage required for all the fingerprints is then relatively large, so they retain only some of the fingerprint (proportional to $\sqrt{|d(F)|}$). This can however lower the *recall* of the algorithm.

On the other hand, Monostori et al evaluate the properties of the chunking method with much smaller—24-bit and 32-bit hash functions. Because of a higher number of hash function collisions, they use the chunking algorithm only for preselection of suspicious document pairs. These pairs are then compared using their plain text form with the *suffix array* method [MM90]. However, constructing the suffix trees can be relatively expensive, especially when the hash function collisions generate lots of false positives.

suffix array

2.11 Valid Intervals

In the PAN'09 plagiarism detection competition we have proposed and used a different approach [KBK09] for post-processing of the matched document pairs: we have decided to retain all the fingerprints (thus preserving the high *recall* value of the algorithm), and store not only the mapping the fingerprint to the list of document IDs, but to store also the exact location of the chunk represented by a particular fingerprint in the document. In fact, we have stored three values for each chunk in the document:

Offset of the first byte of the chunk in the document

This gives an information on where the chunk starts.

Length of the chunk in bytes

This value together with the above one allows to highlight the matched similarity in the user interface. The users of the system want to know not only the similarity value of the document, but often also *where* the similar passages are. For example, at Masaryk University, all the master and bachelor theses contain the disclaimer about the authorship, which is a paragraph with more-or-less similar wording across all the theses. Such a similarity is unimportant for discovering the plagiarism.

Sequence number of the chunk in the document

This additional value allows to recognize the adjacent chunks for estimating the coverage of the area of the document, without the need to estimate the average length of the chunk in bytes, and without the need to account for an unknown amount of whitespace between the words of a chunk.

With this data, we can do a post-processing of the matches without the need for parsing the full text of the documents again, as described in the previous section. For 32-bit values of the above three fields and 32-bit document IDs, the additional overhead is only three times more than when storing document IDs only.

We define a *valid interval* of chunks $i..j$ from the document F (with respect to the similar document G) as the interval with the following properties:

valid interval

- both the i -th and j -th chunks are also present in the document G .
- in the interval $i..j$ there are at least 20 chunks, which are also present in the document G .
- for every $x \geq i$ and $y \leq j$, such that $x < y$, and both the x -th and y -th chunks of the document F are also present in the document G , and there does not exist z such that $x < z < y$ where z -th chunk is also present in the document G , $y - x \leq 50$.

As an overlapping (possibly plagiarized) passage we then consider only those valid intervals in the document F , which also map to the valid intervals in the document G .

The key idea here is to ignore the common chunks which are too far away from other common chunks in a given document pair. This allows us to ignore both the different parts of text, matched only because of a hash function collision, and random short sequences of words, occurring legally in both documents, without them being the case of plagiarism.

2.12 Distributed Computing

The chunking algorithm can be distributed to a SMP or a cluster of computers. Our approach is described in [KBB09]. The algorithm essentially computes an inverted index which maps the chunk fingerprint to the list of document IDs. When computing inverted indices, there are two possible approaches: The first one is *document partitioning*, where a subset of documents is assigned to a computing node, which then handles all the terms (words, or in our case, chunk fingerprints) for this set of documents. This approach can be useful for search engines, where it keeps also the evaluation of phrase queries local, and then allows to merge only the most relevant results.

document partitioning

The other approach is *term partitioning*, where a subset of terms (in our case chunk fingerprints) is assigned to a particular computing node. This node then handles only these terms (range of chunk fingerprints) for all the documents. We have found that this approach is more useful for the chunking algorithm, which has to compute not only the inverted index, but also to compute the similarities from this index. In this second phase (step 2 of the algorithm in section 2.5), the algorithm has to evaluate each term from the suspicious document against all the documents containing this

term partitioning

term. Here it is more useful to have the list of all these documents on a single node.

3 Thesis Goals

3.1 Objectives and Expected Results

The main aim of the proposed Ph.D. thesis is to explore those approaches for discovering similar documents, which can be implemented in a production system with a medium to large set of the source documents. The main focus of the thesis is using the system for finding document similarities in an anti-plagiarism system, but I will discuss the other possible use cases (such as in the full-text search engine) as well. In particular, the thesis will cover the following areas:

Forms of plagiarism

I will describe the existing forms of plagiarism, and the means of computer-assisted detection of them.

Overview of document overlap detection approaches

I will try to summarize the available techniques, from using chunks of the text (such as in [FZMS02]) to using the text characteristics and information-retrieval techniques and other approaches. So far the chunking approach seems to be the most promising, so I will focus on it in the rest of the thesis.

Parametrization of the chunk-based algorithm

There is a wide range of possibilities for tuning the chunk-based algorithm parameters—from the tokenization process (use only chunks from the single sentence or without considering the end of a sentence a boundary, use stop-words, lemmatization/stemming, etc.), to the handling the chunks themselves (how many words in a chunk, ignoring the most-frequent chunks, etc). I will discuss the influence of these parameters on the different types of plagiarism (copying the whole text passages versus translation and other means of obfuscating the original text).

Document similarity in a full-text search engine

In July 2009, I started to work on a full-text search engine of the Masaryk University Information System, and I plan to explore further the possibility of using the anti-plagiarism system in IS MU also for removing duplicate and similar documents from the search results (an example of such similar documents can be the home page of the same lecture from different years and/or semesters). It may well be that the best fit set of parameters for the similarity algorithm will be different for the different use cases (anti-plagiarism system versus a full-text search engine).

Distributed implementation

I will further discuss the distributed aspects of the algorithm outlined in [KBB09], including whether to add more data (such as chunk/word positions) to the later stages of the computation.

Valid intervals

I plan to investigate further the idea of valid intervals as described in section 2.11, possibly refining the definition of the valid interval and

evaluating the influence of the valid interval parameters to the overall performance (recall, precision, and granularity) of the algorithm.

Further development

I will investigate the possible further improvements of the system for finding similar documents (such as crawling the WWW or using a general full-text search engine API for finding other possible similar documents outside the existing source documents base).

3.2 Schedule

September 2009-January 2010

Implementing the new version of the anti-plagiarism system. Exploring the possibilities of using the system also for the IS MU fulltext search engine. Research on the fine tuning the algorithm (such as ignoring the most common chunks or using different chunking techniques).

January 2010

Defense of this proposal, doctoral exam.

January 2010-August 2010

Writing the Ph.D. thesis.

September 2010

Submitting the thesis.

4 Results of Study

4.1 On Topic of the Ph.D. Thesis

I have designed and implemented several versions of the anti-plagiarism software used in the Masaryk University Information System. The system has been in production use since 2008, replacing the prototype system implemented in the Oracle database, which has been in use since August 2006.

I have proposed to use the hash values of text chunks as a basis of the overlap detection, which reduced the total time of recomputing the data for the whole set of documents from about 56 hours in the prototype system to about 7 hours on an SMP system. I have also discussed the influence of the hash function collisions to the accuracy of the computed results. I have published my findings in the article [KBKŠ08], extended version of which is available as a technical report FIMU-RS-2008-04.

The further development was exploring the possibilities of moving from a SMP system to a loosely-coupled cluster of computers. I have designed a distributed algorithm based on the above approach and on the chunk ID-based partitioning of the problem space (as opposed to the document ID-based partitioning). The implementation further lowered the total run time to about three hours, with incremental runs taking about 10 to 30 minutes, depending on the system load. The results have been published in [KBB09].

The IS MU anti-plagiarism system has been used as a basis of the software with which we have participated in the 1st International Competition on Plagiarism Detection [PAN09]. We have finished in the second place overall. During modifying the software for the requirements of the competition I have discovered that some of the steps in the computation can be further shortened by carrying out more data, namely the matching chunk positions. We have described those findings as well as our approach in [KBK09].

PAN'09
competition

4.2 Academic Work

I have been an advisor of several bachelor and master theses so far. I name only some of them from the last three years:

David Hrachový: *Applications for the Maemo platform*, bachelor thesis, 2009

Milan Zázrivec: *Algorithms for creating binary patches*, master thesis, 2009

Tomáš Šedovič: *Automatically generated playlist*, bachelor thesis, 2009

Lukáš Nový: *Comparison of virtualization technologies*, bachelor thesis, 2008

Rostislav Beneš: *UTF-8 for Midnight Commander*, bachelor thesis, 2008

Libor Vaněk: *Snapshot of a file system*, master thesis, 2007

Jan Horák: *Czech documentation for SELinux*, bachelor thesis, 2007

Libor Kyncl: *Browsing the source code on WWW*, bachelor thesis, 2007

Ivan Novotný: *Systems for voice over IP*, bachelor thesis, 2007

UNIX I also teach the courses focused on the *UNIX* system architecture: *UNIX—Programming and System Administration I*, *UNIX—Programming and System Administration II*, and *UNIX—Seminar from the System Administration*.

4.3 Other Results

Linux I frequently give a Linux- and UNIX-related lectures in various local workshops and conferences. From the recent years I mention the following papers and presentations: [Kas07], [Kas08], [Kas09], [Kas01].

5 Summary/Souhrn

5.1 Summary

With the wider availability of the electronic texts in the recent years, it has also become easier to use work of other people without the appropriate citation. Fortunately, recent developments in the area of detecting document overlap (and in general, discovery of similar documents), can also make it easier to discover the plagiarized work. The algorithms for discovering similar documents have also other uses, especially in the area of full-text search engines: either for removing duplicate documents altogether, or for preventing a subset of important but similar documents to occupy the whole first page of the search results. This proposed Ph.D. thesis will evaluate the approaches for the discovery of similar documents, especially by detecting document overlap, and verify which of them are suitable for large sets of documents. It will also focus on aspects of practical implementation on a distributed cluster of standalone computers, and usage in a production environment of the Masaryk University Information System.

5.2 Souhrn

S širší dostupností elektronických textů v poslední době se také stalo jednodušším používat práci jiných bez korektní citace. Naštěstí nedávný vývoj v oblasti detekce překryvu dokumentů (a obecně v oblasti objevování podobných dokumentů) může usnadnit také objevení plagiátů. Algoritmy pro nalezení podobných dokumentů mají také další způsoby užití, zejména v oblasti fulltextových vyhledávačů: jednak pro odstranění úplných duplikátů, a jednak pro zabránění několika důležitým ale podobným dokumentům v tom, aby zabraly pro sebe celou první stranu vyhledaných výsledků. Navrhovaná disertační práce vyhodnotí přístupy k objevování podobných dokumentů, zejména pomocí detekce překryvu dokumentů, a ověří, které z nich jsou vhodné pro velké množiny dokumentů. Také se zaměří na praktické aspekty implementace na clusteru samostatných počítačů, a na použití v produkčním prostředí Informačního systému Masarykovy univerzity.

6 References

- [BDGM95] Sergey Brin, James Davis, and Hector Garcia-Molina. Copy detection mechanisms for digital documents. In *Proceedings of the ACM SIGMOD Annual Conference*, pages 398–409, 1995.
- [Bri09] Encyclopædia Britannica. Plagiarism. retrieved 2009-08-24 from <http://www.britannica.com/EBchecked/topic/462640/plagiarism>, 2009.
- [BZ04] Y Bernstein and J Zobel. A Scalable System for Identifying Co-derivative Documents. In *String Processing and Information Retrieval, Proceedings*, volume 3246 of *Lecture Notes in Computer Science*, pages 55–67. Springer-Verlag Berlin, 2004.
- [Cop09] Copyscape. <http://copyscape.com/>, retrieved 2009-08-26, 2009.
- [Cox02] Alan Cox. Alan Cox talks about laws... and Linux. retrieved 2009-08-27 from <http://interviews.slashdot.org/article.pl?sid=02/05/20/1314214>, 2002.
- [CT94] William B. Cavnar and John M. Trenkle. N-gram-based text categorization. In *In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1994.
- [Doc09] Doccop. <http://doccop.com/>, retrieved 2009-08-26, 2009.
- [FZMS02] Raphael A. Finkel, Arkady Zaslavsky, Krisztián Monostori, and Heinz Schmidt. Signature extraction for overlap detection in documents. In *ACSC '02: Proceedings of the twenty-fifth Australasian conference on Computer science*, pages 59–64, Darlinghurst, Australia, 2002. Australian Computer Society, Inc.
- [Hei96] Nevin Heintze. Scalable document fingerprinting. In *In Proc. USENIX Workshop on Electronic Commerce*, 1996.
- [iDn08] iDnes.CZ. Zlínského děkana usvědčili z plagiátorství. retrieved 2009-08-25 from http://zpravy.idnes.cz/studium.asp?c=A080709_085836_studium_bar, 2008.
- [IS 09] Masaryk University Information System. <http://is.muni.cz/>, 1999–2009.
- [Kas01] Jan Kasprzak. Clusterová řešení pod Linuxem. In *SLT 2001: Proceedings of the 2nd Seminar on Linux and TEX*, pages 161–168. Konvoj, Brno, 2001.
- [Kas07] Jan Kasprzak. Desktop a jádro Linuxu. In *Proceedings of the XXXI EurOpen.CZ Conference*, pages 45–60. EurOpen.CZ, Plzeň, 2007.

- [Kas08] Jan Kasprzak. Co umí souborové systémy. In *Proceedings of the XXXII EurOpen.CZ Conference*, pages 105–118. EurOpen.CZ, Plzeň, 2008.
- [Kas09] Jan Kasprzak. Git aneb správa verzí trochu jinak. In *Proceedings of the XXXIV EurOpen.CZ Conference*, pages 107–118. EurOpen.CZ, Plzeň, 2009.
- [KBB09] Jan Kasprzak, Michal Brandejs, and Jitka Brandejsová. Distributed aspects of the system for discovering similar documents. In *ITA 09: Proceedings of the Third International Conference on Internet Technology and Applications*, 2009.
- [KBK09] Jan Kasprzak, Michal Brandejs, and Miroslav Křipač. Finding plagiarism by evaluating document similarities. In *SEPLN'09: The 25th edition of the Annual Conference of the Spanish Society for Natural Language Processing*, 2009.
- [KBKŠ08] Jan Kasprzak, Michal Brandejs, Miroslav Křipač, and Pavel Šmerk. Distributed system for discovering similar documents. In *ICEIS 2008: Proceedings of the Tenth International Conference on Enterprise Information Systems, Vol. DISI—Databases and Informations Systems Integration*, pages 437–440. INSTICC (Institute for Systems and Technologies of Information, Control and Communication), Setúbal, Portugal, 2008.
- [MFZ⁺02] Krisztián Monostori, Raphael A. Finkel, Arkady B. Zaslavsky, Gábor Hodász, and Máté Pataki. Comparison of overlap detection techniques. In *ICCS '02: Proceedings of the International Conference on Computational Science-Part I*, pages 51–60, London, UK, 2002. Springer-Verlag.
- [MM90] Udi Manber and Gene Myers. Suffix arrays: a new method for online string searches. In *SODA '90: Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 319–327, Philadelphia, PA, USA, 1990. Society for Industrial and Applied Mathematics.
- [MU09] Masaryk university: Full-text search. retrieved 2009-08-25 from <http://www.muni.cz/general/search>, 2009.
- [Ode09] Odevzdej—the system for collecting seminar works. <http://odevzdej.cz/>, 2009.
- [PAN09] 1st international competition on plagiarism detection. <http://www.uni-weimar.de/medien/webis/research/workshopseries/pan-09/competition.html>, retrieved 2009-08-26, 2009.

- [PR08] Jan Pomikálek and Pavel Rychlý. Detecting co-derivative documents in large text collections. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, pages 132–135, Marrakech, Morocco, 2008.
- [Riv92] Rivest, R. RFC1321: The MD5 Message-Digest Algorithm. 1992. <http://www.rfc-editor.org/rfc/rfc1321.txt>.
- [SGM95] Narayanan Shivakumar and Hector Garcia-Molina. Scam: A copy detection mechanism for digital documents. In *Proceedings of the Second Annual Conference on the Theory and Practice of Digital Libraries*, 1995.
- [The09] Czech National Archive of Graduate Theses. <http://theses.cz/>, 2008–2009.
- [Tur09] Turnitin. <http://turnitin.com/>, retrieved 2009-08-26, 2009.
- [zES06] Sven Meyer zu Eissen and Benno Stein. Intrinsic plagiarism detection. In Mounia Lalmas, Andy MacFarlane, Stefan M. Rüger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors, *ECIR*, volume 3936 of *Lecture Notes in Computer Science*, pages 565–569. Springer, 2006.