Arno Mittelbach and Lasse Lehmann and Christoph Rensing and Ralf Steinmetz:
Automatic Detection of Local Reuse. In: Martin Wolpers, Paul A. Kirschner, Maren Scheffel, Stefanie Lindstaedt, Vania Dimitrova: Sustaining TEL: From Innovation to Learning and Practice - Proceedings of the 5th European Conference on Technology Enhanced Learning, EC-TEL 2010, no. LNCS 6383, p. 229-244, Springer Verlag, September 2010. ISBN 3-642-16019-0.

Automatic Detection of Local Reuse

Arno Mittelbach, Lasse Lehmann, Christoph Rensing, and Ralf Steinmetz

KOM - Multimedia Communications Lab, Technische Universität Darmstadt,
Rundeturmstr. 10, 64283 Darmstadt,
{firstname.lastname}@kom.tu-darmstadt.de

Abstract. Local reuse detection is a prerequisite for a multitude of tasks ranging from document management and information retrieval to web search or plagiarism detection. Its results can be used to support authors in creating new learning resources or learners in finding existing ones by providing accurate suggestions for related documents. While the detection of local text reuse, i.e. reuse of parts of documents, is covered by various approaches, reuse detection for object-based documents has been hardly considered yet. In this paper we propose a new fingerprinting technique for local reuse detection for both text-based and object-based documents which is based on the contiguity of documents. This additional information, which is generally disregarded by existing approaches, allows the creation of shorter and more flexible fingerprints. Evaluations performed on different corpora have shown that it performs better than existing approaches while maintaining a significantly lower storage consumption.

Key words: Local Reuse Detection, Fingerprinting, Overlap Detection

1 Introduction and Motivation

Detection of reuse is a prerequisite for a multitude of tasks. Duplicate or near duplicate detection plays an important role in web search and retrieval where it is, for example, used to filter result lists. However, there are several tasks that require not only the detection of near duplicate documents but the detection of local reuse (meaning reuse of smaller parts of documents). Detection of plagiarism is an important use case [1] but also in the fields of web search or local desktop search several scenarios require techniques for detecting local reuse. Examples are retrieval of related documents or tracking of information flow for news stories [10, 16], over different blogs [10, 9] or in newsgroups or forums. Even in the field of Technology Enhanced Learning (TEL), the obtained information can be used to support authors as well as users of learning content (see Section 2). Generally, algorithms that efficiently handle the near-duplicate detection problem [7, 4, 5] do not work well for the detection of local reuse [19], while standard string matching approaches like Greedy String Tiling [22] or Local Alignment [6] usually suffer from bad runtime behavior or high storage consumption. There are so-called fingerprinting techniques that allow for the detection of local reuse while maintaining an acceptable runtime and storage behavior. However, most

existing approaches concentrate on the detection of text reuse. Many documents contain not just text but also images, audio, video or other content types such as geometric forms or structures. Especially in the domain of Technology Enhanced Learning such object-based formats are commonly used. Presentation slides are a good example. They often contain many shapes and diagrams but comparably little text. Even documents generated by modern word processing tools usually contain objects like images or charts, which are completely neglected by text-based reuse detection methods.

In this paper we propose a flexible fingerprinting technique for the detection of local reuse based on the contiguity of a document's content (Section 4) that outperforms existing fingerprinting approaches (Section 3) in most aspects on the corpora used for evaluation (Section 5) including an annotated subset of the TREC newswire corpus and the LIS.KOM corpus created during the evaluation of the LIS.KOM framework [13]. In the following Section we define local reuse and show how local reuse detection can be applied to support different scenarios in the area of Technology Enhanced Learning.

2 Application of Local Reuse Detection to support TEL

We define local reuse extending the definition for local text reuse by Seo and Croft [19] as follows: *Local reuse occurs when content is copied from one document to another.* In this definition, content can be anything from a part of a sentence to an image or an audio file. The reused content can be modified and may only make up a small fraction of the new document. It is apparent that local reuse relations are not transitive (a property often assumed for near-duplicate documents [7]), meaning that document A may reuse content from document B and B content from document C without A and C having anything in common.

The main use case for local reuse detection is to discover documents which overlap and thus are related to each other. This information can be used to support retrieval and generation of learning content. The resulting document relations can e.g. be used to generate recommendations, enrich given search results, or track the flow of information.

If one document is part of a search result or relevant for a user, e.g. because he is learning it, related documents might be potentially interesting too. Documents that are near-duplicates, are not useful for recommendation since the user already knows most of the content. They are thus usually filtered out by search engines like Google. However, this is not the case for documents that overlap in parts only. When a document A is relevant for a user and document B partially overlaps with document A, the content that document B contains in extent to the overlapping passages is probably relevant for a user or learner. Authors of learning content, like e.g. lecture slides or WBTs, can be provided with information about how a document is connected with other documents. This is especially helpful for authors who are not the creator of named document. If the document itself does not contain the content they need, one of the related

documents might. Local reuse detection allows to find such relations and to give authors access to potentially relevant information.

Another way to support authors is to enable them to track their content. This applies mainly to commercial scenarios where an author has financial interest in his documents not being reused without permission. However even in open content scenarios authors might want to know where their content goes and who reuses it.

3 Fingerprinting Approaches for Local Reuse Detection

The given section covers the methodology for fingerprint generation and gives an overview on existing approaches for local reuse detection for text-based as well as for object-based documents.

3.1 Fingerprint Generation

While near-duplicate fingerprinting techniques generally produce one fingerprint per document, techniques for the detection of local reuse usually generate several fingerprints, i.e. hash values representing a document. How these fingerprints are put together depends on the approach used. Usually the following steps are performed during fingerprint creation: Initially a document is preprocessed. Common preprocessing steps for text include filtering of special characters, stopword filtering or stemming. Secondly chunks are extracted from the document. These chunks are then turned into numeric values using hash functions like MD5 [17]. In the final phase most approaches select a subset of the hashes resulting from the second step. This subset is then called fingerprint (or set of fingerprints) of the document. Most of the approaches described in this section use shingles as chunks. Shingles, as described by Broder [3], are sequences of k consecutive terms that are generated by extracting the contents of a sliding window with fixed size k running stepwise over the text.

3.2 Containment Calculation

To quantify the reuse between two documents, most approaches use an asymmetric measure as such a measure reflects differences in length of the documents. The containment of document A in document B is defined by Broder [3] as:

$$C(A, B) = \frac{|F_A \cap F_B|}{|F_A|} \tag{1}$$

F_A and F_B are the sets of fingerprints of documents A and B respectively and are usually treated as sets in the mathematical sense (e.g. by [8], [18] and [19]) meaning that every fingerprint only occurs once in the selected set of fingerprints, even if it has multiple occurrences in the document. However, taking duplicate fingerprints into account and thus treating F_A and F_B as multisets [20] can be beneficial in some scenarios.

3.3 Approaches for Text-based Documents

K-gram. K-gram is the first and simplest of several approaches that use shingles as chunks. It serves as the basis for most of the following fingerprint schemes. It is based on the n-gram overlap approach [14] and simply uses all shingles of a document generated by a sliding window with size k as fingerprints. Therefore the number of fingerprints generated by k-gram for a document with n tokens is calculated as $N_{k-gram} = n - k + 1$. This, however, only applies when a multiset approach is used. When using k-gram in a set-wise fashion, as it is usually done, the given formula represents an upper bound for the number of fingerprints. The following four approaches take the fingerprints generated by k-gram and reduce their number by selecting a representative subset. Since a subset of k-gram's fingerprints may never provide as accurate results as the whole set, k-gram is usually used as a benchmark to evaluate these approaches.

0 mod p. For a selection algorithm to work properly, it is required to select the identical subset of fingerprints for identical documents. A random selection algorithm would not fulfill this requirement. 0 mod p [15] selects every hash value that is dividable by p out of the hashes generated by k-gram. Thus the average number of fingerprints 0 mod p selects is $N_{0 \ mod \ p} = N_{k-gram}/p$. One drawback of this approach is that if very common shingles are dividable by p the results can be effected negatively. Moreover it is not guaranteed that if two documents contain identical chunks that these chunks will be selected in both documents. Therefore, reuse may not be detected. The guarantee given by 0 mod p is that if a chunk is selected in one document it will be selected in every other document as well.

Winnowing. Winnowing [18] uses a second window with size w sliding over the shingles resulting from k-gram. In each winnowing window the lowest hash value is selected as fingerprint. If there is more than one lowest hash value, the rightmost one within the window is chosen. Schleimer et al. have shown that winnowing yields slightly better results than 0 mod p and give a lower bound for the number of fingerprints chosen as $N_{winnowing} \geq {}^{2}/_{(w+1)} \cdot N_{k-gram}$. While the selection 0 mod p performs is global, winnowing's selection of fingerprints depends on the other hash values within the same winnowing window. This means that even if two documents share a common shingle, winnowing does not necessarily select that shingle's fingerprint in both documents. However, if two documents share a chunk that is at least as large as the winnowing window w, at least one common shingle out of that chunk is selected in both documents.

Other Approaches: Hailstorm [9] combines the results of winnowing with a better and more global coverage of the document. Each token (i.e. word) of a document is hashed separately before applying k-gram to the document. Each shingle is then examined whether the token with the lowest hash value is the leftmost or rightmost token within the shingle. If this is the case, the

shingle is selected as fingerprint. It is necessary to use a large window size to reach high compression rates with Hailstorm. However, the quality of k-gram for detection of local reuse usually decreases for big window sizes (e.g. $k > 6$), since the sensitivity of k-gram for small changes increases with an increasing window size. Originally, Hailstorm has been developed for origin detection. In that use case content is often cited and fully copied from document to document, with relatively long parts of text remaining unchanged. In such a scenario greater values for k - like 8, as proposed in [9] - work well, which is not the case for detection of local reuse.

Hash-breaking [2] is not based on k-gram and thus does not use shingles. A document is divided into non-overlapping chunks. Every word of the document is hashed and if a resulting hash value is dividable by a given parameter p, the text is divided at this point. The resulting text chunks are then hashed and used as fingerprints of the document. While a chunk of text selected by hash-breaking is on average p tokens long, it might be very short or very long in practice, depending on the distribution of hash values. Specifically, when a sequence is very short and consists of very common words, the results Hash-breaking provides are influenced badly. Seo and Croft have modified hash-breaking such that it only selects chunks that are at least p tokens long (revised hash-breaking) [19].

Since the chunks hash-breaking selects can be very long, the approach is very sensitive to changes. If only one character in a chunk is changed, the hash values and thus the outcome of the approach is completely different. Seo and Croft use **Discrete Cosine Transformation** (DCT) to make it more robust against small changes [19].

3.4 Approaches for Object-based Documents

Object-based documents are documents whose main content is non-textual. Typical examples for object-based documents are presentations, web based trainings, documents from modeling tools (like e.g. Visio or ARIS), project charts or the output of various design tools (like Adobe InDesign or MS Publisher). While there are many existing approaches for the detection of reuse for text-based documents, very few approaches for detecting reuse between object-based documents exist or have been researched.

In [11] Klerkx et al. evaluate the reuse of PowerPoint presentations in the ALOCOM repository [21]. To detect reuse, Klerkx et al. utilize the cosine measure for textual content, hash functions for images and a mix of both for the content of slides. No specifics are given and, since different techniques are used for text and images, no common fingerprint can be generated.

Basic approaches like k-gram or the selection algorithms based on k-gram can be adapted for object-based documents. If it is possible to extract ordered features from a document it is possible to extract fingerprints from it. For instance, to generate k-gram fingerprints for a PowerPoint presentation, the objects need to be extracted in the order they occur in the document. To be able to process both objects and textual content at the same time it is necessary to generate individual hash values for each of them and then apply the k-gram window to the

resulting bit string. Hence, the preprocessing used on an object-based document is different, but the principle mechanics of the approach are the same as in the text-based case.

4 MiLe Approach

Documents possess an inherent order. In text documents, for example, the order in which words occur is (for most languages) an eminent aspect of the text's semantics. That is, if the word order is changed, either a nonsense sentence is created or semantics are changed: E.g. *Tom protects Lisa* has a different meaning than *Lisa protects Tom* or *protects Tom Lisa*.

Existing fingerprinting approaches do not consider documents as a contiguous entity but as a set of individual chunks (or features) which forms the basis for the fingerprint generation. While it is possible to store additional offset information for each shingle or chunk (see e.g. [9]), utilizing this information requires additional storage and computational effort, e.g. reconstructing the order of a document's k-gram fingerprints would approximately take $O(n \log n)$ steps, depending on the algorithm used.

We propose a fingerprinting algorithm called MiLe that utilizes the contiguity of documents. The main idea behind MiLe is to not break up documents into chunks and select a subset to calculate a resemblance measure but to transform the resemblance estimation problem into a string-matching problem, thereby benefiting from the additional information taken from the contiguous nature of documents.

The comparison of documents using document fingerprints usually consists of two steps: First a fingerprint is generated for every document, which is then used to estimate the resemblance. This is also the case for MiLe. However, while all local reuse detection approaches described in Section 3.3 generate a set of fingerprints, MiLe creates only one fingerprint per document.

4.1 Fingerprint Generation

While the generation of fingerprints for text-based and object-based documents is different, the calculation of containment is not, i.e. once a MiLe fingerprint has been created it can theoretically be compared to any MiLe fingerprint regardless of the underlying document type. Therefore we first describe the two different methods of fingerprint generation before proceeding with the general calculation of containment measures.

Generating MiLe Fingerprints for Text-based Documents. The resulting bit sequence of a MiLe fingerprint is created from a text-based input document by performing the following steps:

1. The document's text content is extracted and preprocessed. Preprocessing can, for example, involve case folding, stop-wording or stemming. For the

evaluation (Section 5) we mainly used case folding (i.e., we converted all characters to lower case).
2. The preprocessed text is tokenized at word level.
3. Every token is projected onto a hash-value consisting of very few bits (e.g., 4 bits). A simple algorithm could calculate the MD5 value [17] for each token while only using the first n bits. Of course, custom and hence more efficient projection methods could be used when runtime is critical.
4. The generated bit values are appended to a bit string, in the order their corresponding token appears in the document. This bit string is the document's MiLe fingerprint.

MiLe fingerprints are relatively short and can be seen as a lossy compressed version of the document as each token of the input text is mapped to a part of the resulting fingerprint while the order in which tokens occur is preserved. As a result, MiLe fingerprints can be used to locate the exact position of matching parts in the original documents without having to have access to these documents. If stop wording is part of the preprocessing MiLe can still be used to give a good estimate as to where reuse has occurred.

Generating MiLe Fingerprints for Object-based Documents. The generation of a MiLe fingerprint for an object-based document works quite similar as for a text-based document. In the following, we describe the generation of a MiLe fingerprint using a PowerPoint presentation as example. However, the presented algorithm can be applied to arbitrary types of object-based documents as long as they allow their content to be accessed. There are three main differences to the text-based version that need to be addressed when generating MiLe fingerprints for object-based documents:

1. The preprocessing
2. The order of objects
3. The object feature used for fingerprinting

For preprocessing, the object-document is decomposed into modular objects, that means all grouped elements in the presentation are "un-grouped" until they reach a modular state. The text elements are preprocessed like in the text-based version of MiLe.

While the order of slides in a presentation is given by default, objects on a slide have no obvious order. However, since MiLe is order preserving two identical slides need to be processed in the exact same order. There are different possibilities to determine the order of objects including the position of an object (i.e. its coordinates), its layer or - what we make use of - a given ID. This ID is unique for all objects on a slide. During the preprocessing of a PowerPoint document the non-textual objects on a slide are ordered by their ID.

The last difference to the text-based version of MiLe is how objects find entrance into the fingerprint. Texts are usually divided into tokens (i.e. words) and each token is used as feature for the generation of the fingerprint. We do

this as well for the text fragments of object-based documents. However, for a non-text object there are various features that could be used for fingerprinting. Possibilities include its shape, area, width and height, color or combinations of the above. These work well for basic objects like lines, arrows or geometric shapes. However, for images or even more complex objects the named features are not distinctive enough. In those cases we export a scaled image of the object as feature.

The features are then processed in the given order the same way as in the text-based version of MiLe to generate the fingerprint of an object-based document.

4.2 Containment Calculation.

Once the fingerprints of two documents have been created, reuse between these documents can be detected, measured and localized without further access to the documents' contents.

To determine the resemblance of documents A and B, their MiLe fingerprints are treated as strings and a basic string matching algorithm is applied:

1. Shingles are extracted from B's fingerprint and stored in a lookup table. Here, a token corresponds to the bit length chosen during fingerprint creation. The shingle length corresponds to the minimal length m a match has to have at least. This parameter also greatly affects the probability of collisions: With a bit length of four and a shingle size of five the size of a comparison unit will be 20 bits.
2. An integer used for counting matches is initialized.
3. Shingles (using the same shingle length as above) are extracted from A's fingerprint. Each shingle is, in the order of appearance, compared to the list of shingles extracted for B.
4. If B contains the shingle, the match counter is incremented with regard to the previous shingle. The counter is incremented by one, if B did contain the previous shingle, as this means that two consecutive shingles and hence $m+1$ tokens matched. The counter is incremented by m if B did not contain the previous shingle: Here an individual shingle (m tokens) matched. [1]

To obtain the containment of A in B the value of the resulting match counter c has to be divided by the length of A's fingerprint (number of tokens):

$$C(A, B) = \frac{c}{N_A - m + 1} \tag{2}$$

Hence, the calculation is not based on the number of matching shingles but on the number of matching tokens (i.e. words) which improves the accuracy. As each document always consists of roughly as many shingles as tokens the divisor in equation 2 is in both cases almost identical. The dividend on the other hand can differ greatly, depending on the shingle size and the distribution of matches.

[1] For further refinement one can keep track of the shingles that have been matched and delete them from the list of B's shingles.

4.3 Using MiLe on large corpora

MiLe shows a runtime behavior that is similar to state of the art approaches like k-gram for the one-to-one comparison of document fingerprints. However, mostly this is not the main use case. What is usually needed is an efficient way to compare one input document to a large scale corpus. One scenario could be a plagiarism detection system that offers customers the possibility to upload an input document that is then checked against all stored documents. In this section we will describe how to use inverse indexes to store MiLe fingerprints that allow for efficiently processing such queries. A naïve approach would be to build a lookup table using the fingerprint's shingles as keys, thereby generating keys of the size of MiLe's comparison unit (cp. Section 4.2). While in this setting we would still benefit from the more accurate resemblance calculations and the robustness against small changes (see Section 4.4), we now have a storage consumption that is equal to that of k-gram and, if we additionally store the complete MiLe fingerprint, even higher. The easiest way of reducing the storage requirements is to not insert a document pointer for every comparison unit. This is similar to the subset selection approach of 0 mod p, Winnowing or Hailstorm. If we only store a document pointer for every o^{th} shingle (we call o the overleap factor) we reduce the storage requirements by a factor of o. To still benefit from the contiguity of the content we change the resemblance calculation algorithm as follows (m describes the minimal match length, o the overleap factor):

1. Create MiLe fingerprint for the input document that is to be compared to the corpus and initialize a counter c_i for each document D_i in the corpus.
2. For each shingle (i.e. comparison unit) in the document's fingerprint lookup the shingle in the hash-table.
   (a) If it is not found, continue with the next shingle.
   (b) If a match was found for a document D_i and the previous shingle for D_i was not a match then increment c_i by $m + w \cdot (o - 1)$. Here w can be used to parameterize the percentage of how many of the overleaped tokens will be marked as matching (see below).
   (c) If the previous shingle for document D_i was also a match, increment c_i by the overleap factor o.
3. To obtain the containment the resulting match counters c_i have to be divided by the length of the corresponding fingerprints (cp. Section 4.2).

The idea behind these incrementing steps is that if a match was the first in a series there is a certain probability that the previous $o - 1$ tokens are also matches but were not matched because of the overleap. As we cannot quantify this probability we assume that it is 50% therefore setting $w = 0.5$ and hence adding $0.5(o-1)$. If the match is not the first in a series we act on the assumption that the tokens in between are also matches and therefore increment the counter by o. This strategy is comparable with the bridging strategy proposed in [9]. However, Hamid et al. still weigh all shingles equally and thus do not take the contiguity of the underlying texts fully into account. If the overleap factor is less or equal to the minimal match size, the missing steps between two matches

can be easily obtained out of the keys of two consecutive matches. This can be used to refine the containment calculation if necessary. For some use cases, for example, when the exact position of where reuse has taken place is relevant the complete MiLe fingerprint has to be stored alongside the inverted index. However, as the space needed to store the MiLe fingerprint is constant and since usually low bit rates can be used, the overleap factor approach still yields good compression rates as seen in Section 5.

4.4 Properties of MiLe

The two dominant parameters of the MiLe approach are the bit length b for fingerprint creation and the minimal match length m. The size of a MiLe fingerprint for a document A with $|A|$ tokens is $|A| \cdot b/8$ bytes. The size of MiLe fingerprints can therefore directly be controlled by adjusting b.

When working with very low bit sizes (such as $b = 2$) the minimal match length m has to be increased in order to avoid accidental collisions. This can be regarded as a strictness factor, as MiLe ignores all matches with less than m tokens. If a bigger bit size is used, a smaller m can be used accordingly. Since the bit size has to be chosen before creating the fingerprints it cannot be adjusted afterwards[2]. The minimal match length on the other hand can be adjusted for each resemblance calculation, thereby it is possible to adapt MiLe for a specific use case without recalculating fingerprints.

k-gram is often criticized for its lack of flexibility and - mainly for bigger window sizes - sensitivity to small changes. The MiLe fingerprint provides for a certain kind of robustness against small changes. It can be quantified as a function of the ordered common tokens in two compared sequences. The probability that two sequences are matched by MiLe is

$$pr_{match} = \frac{1}{2^{b(m-c)}} \quad 0 \leq c \leq m \tag{3}$$

where c is the number of common tokens of two sequences in the respective order, b the bit size and m the minimal match length. Thus the more tokens in two sequences already match, the higher the probability that these sequences will be matched although they do not overlap completely. The steepness of this increase is determined by the chosen bit size. $c = m$ constitutes a regular match.

How to choose b and m very much depends on the use case at hand. From the evaluations conducted (see Section 5) we can conclude that the size of the comparison unit (i.e. $b * m$) should be in the range of a bit size which is sufficient for algorithms like k-gram or winnowing on the given corpus. In use cases where the basic MiLe algorithm is applicable it is preferable to use very low bit rates (e.g. $b = 3$) as this yields the highest compression rates. However, this means that the containment calculation has to be conducted in a stricter mode (larger m). On the other hand, if used with an inverted index (see Section 4.3) the

[2] Actually, it is possible to downgrade MiLe fingerprints, i.e. create the 4-bit MiLe fingerprint out of the 5-bit fingerprint.

dominant factor in respect to storage consumption is the number of document pointers stored per document. Hence the bit size can be increased to allow for a smaller minimal match size.

A simple statistical analysis, under the assumption that words are equally distributed, suggests that with a comparison unit of 20 bits it is possible to handle a corpus consisting of one million documents each consisting of $10,000$ words. Under these assumptions, when comparing an input document to the entire corpus, we can expect less than 2000 documents where the containment calculation is off by more than 5% due to collisions originating from a too small comparison unit. Note, that in this model the probability for collisions grows linear with the number of documents in the corpus, i.e. if the number of documents in the corpus is doubled, you should expect twice as many documents with an error of more than 5% in the containment calculation.

5 Evaluation

To evaluate MiLe and compare it to existing approaches two corpora have been used. An annotated subset of the TREC newswire corpus created and used by Seo and Croft [19] for text-based documents and a corpus which resulted from the evaluation of the LIS.KOM framework (see [12] and [13] for details) for object-based documents. We compare the results of MiLe with the results of k-gram quality wise. K-gram is generally seen as benchmark and none of the selection algorithms based on k-gram performs equally well. Neither Hash-breaking nor DCT fingerprinting reached the quality of k-gram either. To show that MiLe can compete in terms of storage consumption we additionally evaluated 0 mod p and winnowing with different configurations and compared the fingerprint sizes on the TREC corpus.

5.1 Annotated TREC Corpus

The TREC newswire corpus is a large corpus consisting of news stories of several big American newspapers and news agencies like Wall Street Journal, LA Times, Financial Times or Associated Press. We use a small manually annotated subset of it. We have chosen this corpus as it has been specifically created for the evaluation of reuse detection methods.

Evaluation Setting. The annotated TREC corpus consists of 600 document pairs manually categorized in six categories. The corpus was originally created by Seo and Croft to evaluate their DCT fingerprinting approach against other existing approaches [19]. Three levels of reuse are distinguished, which when applying an asymmetric containment measure, result in six different categories. The categories used are named:

1. Most / Most (C1): Most of the text of document A is reused covering most of the text of document B

2. Most / Considerable (C2): Most of the text from document A is reused in B covering a considerable amount of document B
3. Most / Partial (C3): Most of the text of document A is reused in B, covering only a small part of document B
4. Considerable / Considerable (C4) (as above)
5. Considerable / Partial (C5) (as above)
6. Partial / Partial (C6) (as above)

The thresholds for manual categorization were at 0.80 containment for most, 0.50 for considerable and 0.10 for partial. Since the corpus did not contain document pairs with no reuse and hence the partial threshold would be ineffectual for automatic classification, we added a seventh category with documents that have no resemblance. We have added 100 document pairs for category 7.

The evaluation of a reuse detection approach can now be treated as a categorization problem and thus quality measures like precision and recall can be applied. We decided to use a ten fold cross-validation approach on the given 700 document pairs. We used the Weka framework for our tests and the F1 measure as harmonic mean of precision and recall as measure. We averaged the results over ten runs to reduce statistical variations. We used the two containment measures for each document pair as the only features for Weka's tree based J48 classifier.

Results. We have compared MiLe's performance to that of k-gram, 0 mod p and winnowing. Table 5.1 contains an excerpt from the results. For k-gram, winnowing and 0 mod p 16 bit fingerprints were used, as the performance stalls at that point. However bit sizes down to 13 bits produce more or less acceptable results. The standard deviation of the average F1 value over the ten runs is given in parentheses. MiLe was evaluated in the two modi described in Section 4: the basic algorithm and the extended version for large corpora with a given overleap factor o. We included the best configurations for 0 mod p and winnowing and found that shingle sizes of only 2 and 3 produce the best results. Note that the fingerprint sizes given in table 5.1 assume that the fingerprints are stored directly. When using an inverted index the fingerprint size would depend upon the size used for a document pointer. The tendency suggested by the results can however be converted directly to the number of document pointers needed per document.

With only 2 bits, basic MiLe produces slightly better results than 0 mod 4 while producing a fingerprint that is only half the size. Even when assuming 13 bit fingerprints for 0 mod p (which comes at the cost of quality) MiLe would still outperform 0 mod 4 in terms of storage consumption. As expected, with increasing bit rates MiLe's results increase quality wise and from 4 bits on MiLe's results even get better than k-gram's. Still the storage consumption is comparable to that of 0 mod p and winnowing.

When considering large corpora the overleap factor described in Section 4.3 allows MiLe to use an inverted index. If an overleap factor equal to the minimal match size is used the storage space needed per document is equal to that of a

Table 1. Performance of fingerprinting techniques on annotated TREC newswire collection. MiLe is divided into the basic algorithm (Section 4.2) and MiLe in large corpora settings (Section 4.3). The fingerprint sizes are given in bytes.

Approach	*C1*	*C2*	*C3*	*C4*	*C5*	*C6*	*C7*	*F1 (std)*	*Size*
2-gram	0.954	0.859	0.909	0.833	0.818	0.940	0.994	**0.901** (0.012)	600
3-gram	0.930	0.838	0.880	0.826	0.749	0.895	0.992	**0.873** (0.009)	600
0 mod 4, k=2	0.901	0.716	0.817	0.713	0.628	0.908	0.995	**0.811** (0.010)	150
0 mod 6, k=2	0.902	0.701	0.760	0.673	0.526	0.875	0.986	**0.775** (0.013)	100
winnowing									
k=2, w=4	0.905	0.781	0.871	0.778	0.735	0.909	0.995	**0.853** (0.011)	240
k=3, w=6	0.882	0.758	0.853	0.783	0.682	0.876	0.993	**0.832** (0.010)	171
MiLe									
b=2, m=8	0.915	0.853	0.854	0.781	0.594	0.772	0.959	**0.818** (0.010)	75
b=3, m=5	0.928	0.864	0.837	0.818	0.712	0.886	0.984	**0.861** (0.010)	113
b=4, m=4	0.925	0.890	0.894	0.802	0.734	0.937	0.998	**0.883** (0.014)	150
b=5, m=4	0.926	0.905	0.896	0.871	0.813	0.957	1.0	**0.910** (0.010)	188
MiLe									
b=3, m=5, o=6	0.924	0.796	0.824	0.755	0.554	0.833	0.982	**0.810** (0.019)	94
b=4, m=5, o=4	0.904	0.844	0.900	0.797	0.736	0.896	0.992	**0.867** (0.013)	188
b=4, m=5, o=5	0.913	0.774	0.850	0.751	0.670	0.900	0.995	**0.836** (0.014)	150

basic MiLe fingerprint. The quality decrease on the other hand is still low enough as for MiLe to produce better results with less storage consumption than 0 mod p or winnowing.

5.2 LIS.KOM Corpus

The LIS.KOM framework supports, among others, the automatic capturing of structured relations emerging from reuse processes conducted on documents. During the evaluation of the framework reuse relations between PowerPoint documents have been captured. These relations reflect the reuse of PowerPoint documents and include near-duplicates as well as partial reuse. The collected relations have been manually categorized with regards to their validity. The LIS.KOM framework seldom captures invalid relations. However, while PowerPoint documents are reused because of their contents most of the time, sometimes a document is reused because of its template or structure. These relations are captured by the LIS.KOM framework but are worthless for most - if not all - utilization scenarios. Thus it is desirable to sort these out automatically.

Evaluation Setting. The object-based LIS.KOM corpus consists of 290 relations connecting 367 different PowerPoint documents. 244 of the relations have been categorized as *valid* whereas 46 relations have been categorized as *template* or *invalid* relations. In most cases documents connected by a non-valid relation do not overlap as much as valid relations do. Thus it is feasible to apply fingerprinting techniques to automatically determine the validity of a given relation.

A simple categorization problem with two categories results where precision and recall can be applied. For this evaluation we compared the object-based version of MiLe with a straightforward adaptation of k-gram for object-based documents. We applied a ten fold cross-validation using the containment measures for each relation as features for optimizing the containment thresholds on the training sets. Due to the given distribution of categories a baseline algorithm which categorizes each relation as *valid* reaches a quality of already 84.1%.

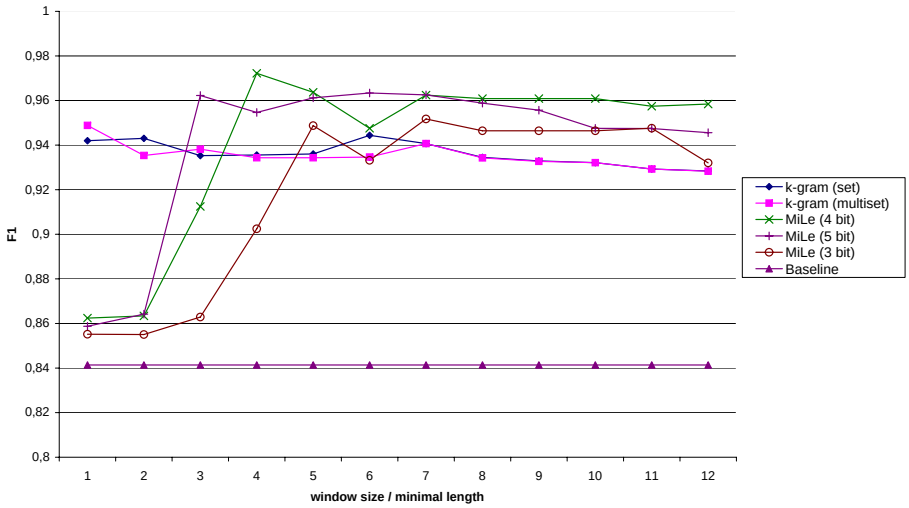

Fig. 1. Comparison of MiLe and k-gram for object-based documents

Results. Figure 1 shows the results for the evaluation of MiLe and k-gram on the LIS.KOM corpus. It is evident, that for bit sizes of 4 or 5, MiLe performs significantly better than the adapted k-gram. Even when using MiLe with a bit size of 3, the algorithm performs better than k-gram for values of $m \geq 7$. The categorization quality of both algorithms is comparatively stable for many different parameterizations. The storage consumption of MiLe is for all bit sizes significantly smaller than that of k-gram (as shown in Section 5.1), especially when using MiLe with 3 bits. Both approaches are able to raise the given baseline considerably. The optimal configuration of MiLe reaches a categorization quality of 97,2% for the F1 value. This is the case for a configuration of MiLe with bit size 4 and minimal match length 4, which is a configuration very similar to the optimal one on the TREC corpus. For both algorithms, the optimal thresholds for the calculated containment reside between 10 and 30 percent in most configurations. However, the stricter the configuration, i.e. the bigger the window size or minimal match length respectively, the smaller the optimal threshold which the containment has to pass for a relation to be counted as valid.

6 Conclusions

In this paper we propose MiLe, a fingerprinting approach for detecting local reuse, which is a current and relevant topic of research and can be applied to support Technology Enhanced Learning in various ways. MiLe can be used to detect text reuse as well as reuse between object-based documents (e.g. PowerPoint presentations). We have evaluated the performance of our approach for both cases, the former on an annotated subset of the TREC newswire collection, which was specifically designed for this purpose, and the latter on the LIS.KOM corpus. Our results show that MiLe is not only able to produce better results than k-gram - which is generally used as benchmark for local reuse detection - but that it can also compete with selection approaches like 0 mod p or winnowing in terms of storage consumption. MiLe's runtime performance is comparable to that of the k-gram based selection algorithms when using an inverted index as described in Section 4.3. As the evaluation has shown, the overleap factor allows for compression rates better than those achieved by 0 mod p or winnowing, while still producing better results.

So far the MiLe approach is promising, as evaluations have shown good results, especially quality wise. We have proposed an inverted index structure that allows MiLe to be used on large corpora. However, so far MiLe has only been evaluated on comparably small corpora, so that future work has to include an evaluation of MiLe on a large corpus. This should also be used to empirically validate the claims made in Section 4.4 on how to choose MiLe's parameters in order not to generate too many collisions. Beyond that, future work will include in-depth research on the exploitation of the flexibility provided by the full length MiLe fingerprints, as this is one of the interesting factors that distinguishes MiLe from other approaches.

References

1. Alberto Barrón-Cede no and Paolo Rosso. On automatic plagiarism detection based on n-grams comparison. In *ECIR '09: Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, pages 696–700, Berlin, Heidelberg, 2009. Springer-Verlag.
2. Sergey Brin, James Davis, and Héctor García-Molina. Copy detection mechanisms for digital documents. In *SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 398–409, New York, NY, USA, 1995. ACM.
3. Andrei Z. Broder. On the resemblance and containment of documents. In *SEQUENCES '97: Proceedings of the Compression and Complexity of Sequences 1997*, page 21, Washington, DC, USA, 1997. IEEE Computer Society.
4. Andrei Z. Broder. Identifying and filtering near-duplicate documents. In *COM '00: Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching*, pages 1–10, London, UK, 2000. Springer-Verlag.
5. Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. In *Proceedings of the Sixth International World Wide Web Conference (WWW6)*, pages 1157–1166, 1997.

6. Steven Burrows, S. M. M. Tahaghoghi, and Justin Zobel. Efficient plagiarism detection for large code repositories. *Softw. Pract. Exper.*, 37(2):151–175, 2007.
7. Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC '02: Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM Press, 2002.
8. Paul Clough, Robert Gaizauskas, Scott S.L. Piao, and Yorick Wilks. METER: MEasuring TExt Reuse. In *Proceedings of the 40th Anniversary Meeting for the Association for Computational Linguistics (ACL-02)*, pages 152–159, Philadelphia, July 2002.
9. Ossama A. Hamid, Behshad Behzadi, Stefan Christoph, and Monika Henzinger. Detecting the origin of text segments efficiently. In *WWW '09: Proceedings of the 18th international conference on World wide web*, pages 61–70, New York, NY, USA, 2009. ACM.
10. Jong Wook Kim, K. Selçuk Candan, and Junichi Tatemura. Efficient overlap and content reuse detection in blogs and online news articles. In *18th International World Wide Web Conference*, April 2009.
11. Joris Klerkx, Katrien Verbert, and Erik Duval. Visualizing reuse: More than meets the eye. In *Proceedings of the 6th International Conference on Knowledge Management (I-KNOW) 2006*, pages 489–497, Graz, Austria, September 2006.
12. Lasse Lehmann, Tomas Hildebrandt, Christoph Rensing, and Ralf Steinmetz. Capture, management and utilization of lifecycle information for learning resources. *IEEE Transactions on Learning Technologies*, 1(1):75–87, Mar 2008.
13. Lasse Lehmann, Arno Mittelbach, Christoph Rensing, and Ralf Steinmetz. Capture of lifecycle information in office applications. *International Journal of Technology Enhanced Learning*, 2:41–57, 2010.
14. Caroline Lyon, James Malcolm, and Bob Dickerson. Detecting short passages of similar text in large document collections. In Lillian Lee and Donna Harman, editors, *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 118–125, Pittsburg, PA USA, 2001.
15. Udi Manber. Finding similar files in a large file system. In *WTEC'94: Proceedings of the USENIX Winter 1994 Technical Conference on USENIX Winter 1994 Technical Conference*, pages 2–2, Berkeley, CA, USA, 1994. USENIX Association.
16. Donald Metzler, Yaniv Bernstein, Bruce W. Croft, Alistair Moffat, and Justin Zobel. Similarity measures for tracking information flow. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 517–524, New York, NY, USA, 2005. ACM.
17. R. Rivest. The md5 message-digest algorithm, 1992.
18. Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. Winnowing: Local algorithms for document fingerprinting. In *Proceedings of SIGMOD 2003*, San Diego,CA, June 2003. ACM, ACM Press.
19. Jangwon Seo and W. Bruce Croft. Local text reuse detection. In *Proceedings of SIGIR '08*, Singapore, July 2008. ACM, ACM Press.
20. Apostolos Syropoulos. Mathematics of multisets. In *WMP '00: Proceedings of the Workshop on Multiset Processing*, pages 347–358, London, UK, 2001. Springer-Verlag.
21. Katrien Verbert, Xavier Ochoa, and Erik Duval. The alocom framework: Towards scalable content reuse. *Journal of Digital Information*, 9, 2008.
22. Michael J. Wise. Running karp-rabin matching and greedy string tiling. Technical report, Basser Department of Computer Science - The University of Sydney, 1993.