Learning Hash Codes for Efficient Content Reuse Detection

Qi Zhang, Yan Wu, Zhuoye Ding, Xuanjing Huang
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, P.R.China
{qi_zhang, 10210240075, 09110240024, xjhuang}@fudan.edu.cn

ABSTRACT

Content reuse is extremely common in user generated mediums. Reuse detection serves as be the basis for many applications. However, along with the explosion of Internet and continuously growing uses of user generated mediums, the task becomes more critical and difficult. In this paper, we present a novel efficient and scalable approach to detect content reuse. We propose a new signature generation algorithm, which is based on learned hash functions for words. In order to deal with tens of billions of documents, we implement the detection approach on graphical processing units (GPUs). The experimental comparison in this paper involves studies of efficiency and effectiveness of the proposed approach in different types of document collections, including ClueWeb09, Tweets2011, and so on. Experimental results show that the proposed approach can achieve the same detection rates with state-of-the-art systems while uses significantly less execution time than them (from 400X to 1500X speedup).

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Information Search and Retrieval; H.3.7 [**Digital Libraries**]: Collection, Systems Issues

Keywords

Content Reuse Detection, GPUs, Learning to Hash

1. INTRODUCTION

There is a quick expansion in the popularity of user generated content in forums, microblogging sites, blogs, and other medius in recent years. These broadcast mediums provide opportunities for users to exchange content. According to the statistics, users in Twitter send 230 million tweets per day[8]. Technorati's State of the Blogosphere Report also showed that there are about 126 million blogs on the Internet in 2010. The development of these platforms has resulted in the number of user generated content (UGC) rapidly growing during the last few years.

While the increasing of UGC, *content reuse*, which is the practice of using existing content components, occurs frequently in these mediums. It contains various forms including duplicate, near-duplicate, and partial-duplicate. Exact duplicate documents can be easily identified by standard checksumming techniques. Near-duplicate web pages contain identical core content but are different in framing, navigation bar, advertisements, footer, or other non-content parts. Partial-duplicate, which is a more difficult problem, contains quoted phrases, sentences, or passages from other documents.

Duplicate or near-duplicate detection can help search engine to reduce storage costs and improve the quality of search indexes. It may also avoid users to see redundant documents in search results. Applications including plagiarism detection, information flow tracking, opinion mining, and so on may benefit from the partial-duplicate detection or involve it as the basis. Along with the increasing requirements, content reuse detection has received much attention in recent years. Many efficient and effective algorithms have been proposed [11, 15, 19, 23, 24, 25, 28, 33].

The challenges of content reuse detection include: 1) reuse may happen at different levels; 2) massive documents should be efficiently processed. Partial-duplicate detection may require different algorithms to the approaches proposed for resolving near-duplicate document detection. One of the main reason for this is that only a small part of a document is taken from others. Content reuse detection algorithms have to face with enormous documents, due to the rapid growth of the web. Thus, it is essential that content reuse detection methods should be efficient and scalable.

In this paper, we investigate a novel approach to detect sentence level content reuse by mapping sentence to a signature space. Signature of a sentence is created by taking the *bitwise-or* of all signatures of words occurs in the sentence. Rather than using traditional hash functions, which do not consider statistics of words or characters, to assign hash code for each word/character, we analyze the requirements of what the good codes should satisfy and formalize it as a constraint optimization problem. Since the task of finding optimal codes is NP hard, we relax the optimization problem and introduce a efficiency method to calculate the codes. With the signature generation method, sentences whose reuse scores are predicted to be less than a given threshold are eliminated. Experimental results show that the codes generated by the proposed method outperform

SIGIR'12, August 12–16, 2012, Portland, Oregon, USA.

state-of-the-art approaches. In order to handle millions of documents, graphical processing units (GPUs) are used to implement the detection algorithm.

The contributions of this work are as follows: 1) We outline and discuss what makes good codes of words for content reuse detection. 2) Efficient optimization method is proposed based on several relaxation. 3) We provide parallel algorithm and its GPU implementation. 4) Evaluations on six large web collections in both English and Chinese are used to measure the efficiency and effectiveness.

The remaining of the paper is organized as follows: In section 2, we review a number of related work and the state-of-the-art approaches in related areas. Section 3 presents the proposed method. Experimental results in test collections and analyses are shown in section 4. Section 5 concludes this paper.

2. RELATED WORK

Our approach relates to three research areas: content reuse detection, learning to hash and parallel algorithms based on GPUs. In this section, we discuss the related work on these areas.

2.1 Content Reuse Detection

Content reuse detection has received much attention in the past several years. Previous studies on content reuse detection can be roughly divided into two research directions: representation and efficiency. The first one focuses on representing text in different levels with or without linguistic knowledge. With the growth of digital documents, efficiency, has also received much more attentions.

Shingling, which was proposed by Broder [4], uses contiguous subsequences to represent documents. It does not rely on any linguistic knowledge. If sets of shingles extracted from different documents are appreciably overlap, these documents are considered exceedingly similar, which are usually measured by Jaccard similarity. In order to reduce the complexity of shingling, meta-sketches was proposed to handle the efficiency problem [5].

In order to improve the robustness of shingle-like signatures, Theobald et al. introduced a method, SpotSigs. It provides more semantic pre-selection of shingles for extracting characteristic signatures from Web documents [28]. SpotSigs combines stopword antecedents with short chains of adjacent content terms. The aim of it is to filter natural-language text passages out of noisy Web page components. They also proposed several pruning conditions based on the upper bounds of Jaccard similarity.

Chowdhury et al. proposed I-Match [7], which filters the input document based on collection statistics and compute a single hash value for the remainder text. If the documents with same hash value, they are considered as duplicates. It hinges on the premise that removal of very infrequent terms and very common terms results good document representations for the near-duplicate detection task. Since I-Match signatures is respect to small modifications, Kołcz et al. [16] proposed the solution of several I-Match signatures, all derived from randomized versions of the original lexicon.

Different from the methods focused on document level, partial-duplicate detection was proposed by Zhang et al. [33]. They converted the task into two subtasks: sentence level near-duplicate detection and sequence matching. Except for the similarities between documents, the method can simultaneously output the positions where the duplicated parts occur. In order to handle the efficiency problem, they implement their method using three MapReduce jobs.

Local text reuse detection focus on identifying the reused and modified sentences, facts or passages, rather than whole documents. Seo and Croft [23] analyzed the task and defined six categories of text reuse. They proposed a general framework for text reuse detection. Several fingerprinting techniques for the framework were evaluated under the framework.

The most similar work to ours was proposed by Kim et al. [15]. They mapped sentences into a point in a high dimensional space and leveraged range searches in this space. However different with us, they simply use MD5 hash function for each word to generate signature file. In this paper, we outline and discuss what makes a good code for content reuse detection, and propose to use learned hash codes to capture the relations between words/characters to reduce the false matches.

2.2 Learning to Hash

Extensive research on similarity search have been proposed in recent years. Among them hash-based methods were received more attention due to its ability of solving similarity search in high dimensional space. Recently, several researches attempted to find good data-aware hash functions through machine learning.

Hinton and Salakhutdinov proposed to train a multilayer neural network with a small central layer to convert high-dimensional input vectors into low-dimensional codes [13]. They used a two-layer network called a Restricted Boltzmann machine(RBM) [14] to do it. Experimental results showed that it could accelerate document retrieval.

Spectral hashing [30] was defined to seek compact binary codes in order to preserve the semantic similarity between codewords. Weiss et al. defined the criterion for a good code which is related to graph partitioning and used a spectral relaxation to obtain a solution.

Norouzi and Fleet [20] introduced a method for learning similarity-preserving hash functions, which is based on latent structural SVM framework. They designed a specific loss function taking Hamming distance and binary quantization into account.

Zhang et al. introduced Self-Taught Hashing (STH) approach to semantic hashing [32]. They divided the problem of finding small codes into two stages. Firstly, they used unsupervised method, binarised-LapEig, to optimal l-bit binary codes for all documents in the given corpus. The classifiers were trained to predict the l-bit code for unseen documents.

Almost all the current methods for similarity-preserving hash functions attempt to map the high dimensional data, which represents the whole document or sentence, onto binary codes. In this paper, we seek good binary codes for words under the content reuse detection framework.

2.3 GPU-based Algorithms

Graphics programming units is designed for single instruction multiple data(SIMD) paradigm, which is different from general purpose microprocessors. Due to its advantages on massive parallel, high memory bandwidth, and powerful computing capacity, GPUs have been successfully used in numerical algorithms.

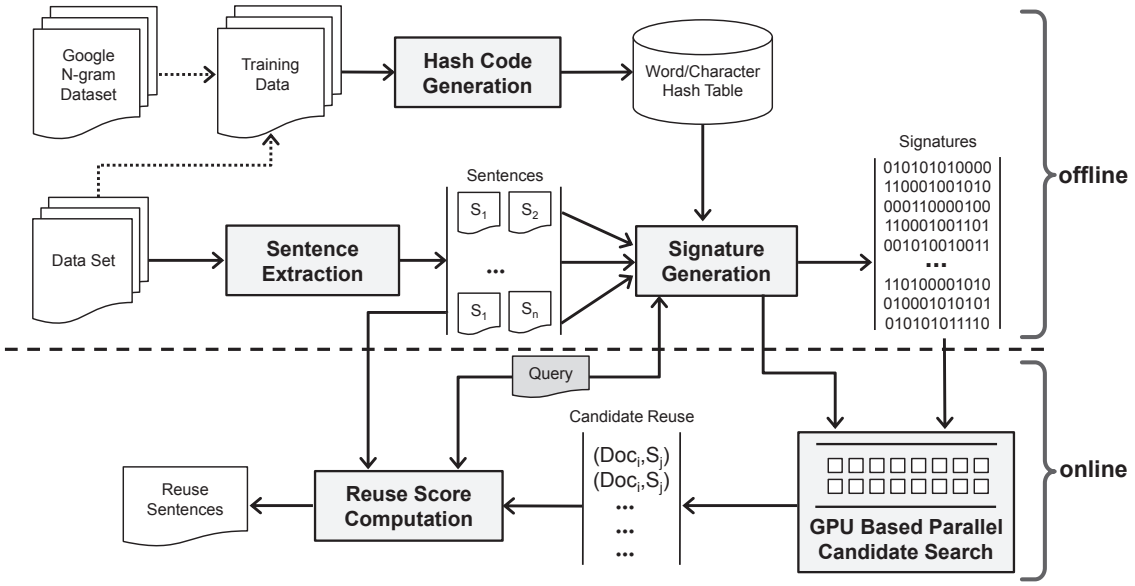


Figure 1: An overview of the proposed content reuse detection approach

Owens et al. [21] introduced their works on implementing three applications (protein folding simulation, scalable molecular dynamics, and calculating electrostatic potential maps). Through these examples, they demonstrated the potential of the GPU for delivering performance gains on real problems.

The emergence of the NVIDIA CUDA programming model speed up the trend of using GPUs to accelerate algorithms. Edelkamp et al. [9] introduced their work on accelerating state space search using GPUs. Linear algebra operators were also implemented to build blocks for more complex numerical algorithms [17]. The results of researches in sort [12], search [6], linear algebra [10], partial differential equations (PDEs) [3], and many other applications have demonstrated the performance and capabilities of GPUs.

F

3. OUR APPROACH

The processing flow of the proposed content reuse detection approach is shown in Figure 1. It consists of two distinct stages:*offline* and *online*. Given a collection of documents, the *sentence extraction* step splits the documents into sentences. *Hash code generation* step takes the training data calculated based on the given collections or open domain data set to generate data-aware hash codes for words or characters. *Signature generation* step uses the hash table to calculate hash code of sentences from both document corpus and given queries. The key algorithm of the online stage is *candidate searching*, which tries to filter the sentences whose reuse scores with the given query are guaranteed not bigger than a given threshold. The reuse sentences and their corresponding reuse scores are calculate in the final step.

Many similarity metrics have been proposed for content reuse detection. Jaccard similarity has been used in various works [4, 33]. Metzler et al. proposed to use weighted word overlap similarity to measure reuse score [18]. There are also a number of works to study the effect of cosine similarity [2, 31]. However, directly use these similarity metrics to detect content reuse in large collections would be very expensive. Because of this, in recent years, hash-based methods have been carefully studied and have demonstrated their advantageous for near similarity search in large document collections [27].

In this paper, we follow the method proposed by Kim et al. [15], which map sentence signature into a point in a high dimensional space. Within this method, each word (or character in Chinese corpus) is assigned a fixed-width bit string. The sentence signature is generated by taking *bitwise-or* of all signature of words in the sentence. Figure 2 shows an example of sentence signature generation process. $h(\cdot)$ represents hash function for words. In [15], two bits were set for each word using MD5 hash function [22] for 32-bit signatures.

Suppose $S_i = b_{i1}b_{i2}b_{i3}...b_{im}$, which consists of m-bits, represents a signature extracted from a sentence. We can map it into a point, $p_i = (b_{i1}, b_{i2}, b_{i3}, ..., b_{im})$, in m-dimensional space. Then the candidate sentences can be selected based on Euclidean distance between sentences, which can be calculated as follow:

$$Distance(p_i, p_j) = \sqrt{\sum_{k=1}^{m} (b_{ik} - b_{jk})^2}.$$

In other words, it also represents the number of bit different between the two sentences. Given a threshold $\sqrt{d}$, the points whose distances lie in the range are extracted as candidates.

3.1 Learning Hash Code

As mentioned above, signatures of sentences are generated based on the hash codes of words. Therefore, how to select hash codes for words has become one of the key problems in this task. In the following parts of this section, we discuss and describe our proposed method.

Sentence: Apple announces iPhone 4s.

h(Apple)	= 0100 0010
h(announce)	= 1100 0010
h(iPhone)	= 1001 0000
h(4s)	= 0101 0011
Signature of Sentence (*bitwise-or*)	= 1101 0011

Figure 2: Example of sentence signature

3.1.1 What good code should satisfy?

The proposed approach tries to find as many as possible reuses under the given upper bound of false matches. In other words, it aims to maximize the recall rate, ρ_{rec}, on the given false positives rate ρ_{fp}. Under the same conditions, the number of bits, m, and the number, l, of bits set to 1 in the signature effect the detection recall and precision. In particular, m, l, and vocabulary size should follow the equation $\binom{m}{l} \geq w$ [15], and l is selected as the smallest value among all candidates. Hence, in order to save the code space, words which usually occur together should have the same hash code.

Based on the descriptions above, we seek hash codes which should satisfy the following properties: (1) the number of bits set to 1 in the word hash code is low; (2) the number of bits to code the vocabulary should be small; (3) words which usually occur together should have same hash codes. In this paper, we formalize the good code seeking task as a constraint optimization problem.

Let $\{y_i\}_{i=1}^n$ be the list of hash codes for n words. $y_i \in \{0,1\}^m$ represents m-bits binary vector. $\#(w_i)$ represents the number of sentences containing word w_i in the given corpus, $\#(w_i, w_j)$ indicates the number of sentences containing both word w_i and word w_j. s_{ij} measures the similarity between word w_i and w_j, which is formulated as following:

$$s_{ij} = \frac{\#(w_i, w_j)}{\sqrt{\#(w_i)}\sqrt{\#(w_j)}}.$$

s_{ij} equals to binary cosine similarity in which a dimension receives a score of 1 when the word appears in the sentence and 0 when it does not appear. The parameter l defines the number of bits set to 1 in the hash code. s_{ij} can be unsupervised generated based on given corpus. By incorporating all the constraints together, we obtain the following problem:

$$\begin{aligned} \text{minimize:} \quad & \sum_{ij} s_{ij} \left(\sum_{k=1}^{m} (y_{ik} - y_{jk})^2 \right) \\ \text{subject to:} \quad & y_i \in \{0,1\}^m, 1 \leq i \leq n \\ & \sum_{k=1}^{m} y_{ik} = l, 1 \leq i \leq n \end{aligned} \tag{1}$$

The property (1) is satisfied by $\sum_{k=1}^{m} y_{ik} = l$, where l is usually set to a small number (in this work, we set $l = \frac{1}{16}m$). The number of bits in hash code is predefined by m to follow the property (2). The property (3) is implemented as the objective function. If two words usually occur together in sentences and have different hash code, it would give negative impact of the objective function.

3.1.2 Optimization

The solving of equation (1) is 0-1 integer programming program, which is a special case of integer programming. It is known to be NP-hard. There is no easy solution for directly optimizing it. Since the number of bits required by equation (1) does not follow the restrictive assumption that the bits are uniformly distributed, the spectrum of the similarity matrix of the data can not be directly used to get the hash codes as spectral hashing [30]. Relaxation and approximation methods should be used to solve the large-scale problem.

First of all, the constraint $y_i \in \{0,1\}^m$ is replaced by $0 \leq y_i \leq 1$. By ignoring the integer constraints, the objective function in (1) is differentiable. In other words, the problem in (1) is relaxed as:

$$\begin{aligned} \text{minimize:} \quad & \sum_{ij} s_{ij} \left(\sum_{k=1}^{m} (y_{ik} - y_{jk})^2 \right) \\ \text{subject to:} \quad & 0 \leq y_i \leq 1, 1 \leq i \leq n \\ & \sum_{k=1}^{m} y_{ik} = l, 1 \leq i \leq n \end{aligned} \tag{2}$$

Since there are usually thousands of y_i and m is also bigger than 32 in practice, the number of parameters tend to be extremely large. The number of constraints is also linear in the size of vocabulary. Because of these, the problem in (2) can not be directly solved within acceptable time either.

In this work, we use an interior-point nonlinear programming algorithm based on a filter line search to solve the problem [29]. Based on it, the inequality constraints are converted to barrier functions which are combined with objective function. We combine the following barrier function with objective function to replace the constraint $0 \leq y_i \leq 1, 1 \leq i \leq n$:

$$-\mu \sum_{i}^{n} \sum_{k}^{m} (\ln(y_{ik}) + \ln(1 - y_{ik})) \tag{3}$$

,where μ is barrier parameter, which is decreased at each optimization iteration. The problem in equation 1 can now be formulated as a sequence of approximate maximization :

$$\begin{aligned} \text{minimize:} \quad & \sum_{ij} s_{ij} \left(\sum_{k=1}^{m} (y_{ik} - y_{jk})^2 \right) \\ & -\mu \sum_{i}^{n} \sum_{k}^{m} (\ln(y_{ik}) + \ln(1 - y_{ik})) \\ \text{subject to:} \quad & \sum_{k=1}^{m} y_{ik} - l = 0, 1 \leq i \leq n \end{aligned} \tag{4}$$

3.2 Reuse Detection

Algorithm 1 presents the pseudo-code for the detection part (marked as *online* in the figure 1) of the proposed reuse detection method. *SigCol* represents a collection of sentence signatures extracted from a document collection. Given *SigCol* and a query sentence, q, the *candidate search* step tries to identify a set of candidate sentences, *CSet*. The *candidate search* step selects sentences by searching the points whose distances between p_q are less than the given threshold $\sqrt{d}$. After that, if the candidate sentences set *CSet* is not empty, the *reuse score computation* step calculates the reuse scores between query sentence with each

Algorithm 1 Pseudo-code of the Reuse Detection

INPUT: a query sentence, q, a distance threshold $\sqrt{d}$, and signatures of document collection, $SigCol$
OUTPUT: a set of detected reuse sentences, O_q

Candidate Selection:

1: Generate signature S_q for the given query q,
2: **for all** $S_i \in SigCol$ **do**
3: **if** $Distance(S_q, S_i) < \sqrt{d}$ **then**
4: $CSet = CSet \bigcup i$
5: **end if**
6: **end for**
7: **return** $CSet$

Reuse Score Computing:

1: **for all** $i \in CSet$ **do**
2: $Sim_{q,i} = Jaccard(S_q, S_i)$
3: **if** $Sim_{q,i} > \theta$ **then**
4: $O_q = O_q \bigcup < i, Sim_{q,i} >$
5: **end if**
6: **end for**
7: **return** Q_q

candidate based on Jaccard similarity. If the reuse score of a sentence is greater than the pre-defined threshold, the sentence and its corresponding document is added in the final list.

From analyzing the calculation consumption of algorithm 1, we observe that the most time consuming part is spent on step 3 of the candidate selection step. Since the number of sentences in $SigCol$ is usually tens of millions, $Distance(S_q, S_i)$ takes the most computing time of the whole algorithm. Fortunately, GPUs offered us an opportunity to accelerate the performance of the algorithm. Modern GPUs are massively parallel processors with extremely high memory bandwidth. Many operations can be performed in parallel. The distance calculation part is suitable for implementation on GPUs as it is fairly simple and consumes a huge part of computing resources.

The function executed GPUs in parallel is called *kernel*, which is driven by *threads* and grouped together in *blocks* and *grids*. In this work, we implement the step 3, 4, and 5 in *candidate selection* part as a kernel function. Based on the thread index and block index, different sentence signature S_i will be justified in different threads. Since signature file of the whole corpus is small enough (302MB for 10 million sentences), it can be easily loaded into the global memory of modern GPUs entirely at once.

The implement of the *candidate selection* is shown in figure 3. The first step is to copy signatures extracted from a collection into global memory. Different threads would parallel process different parts of signatures in stream processors. The step 2 and step 3 in the figure are iterated for each query. Because the bottleneck of this task on GPUs is the memory access rather than processing time, if multiple queries were processed at the same time, the acceleration ratio to CPU implementation would be further improved.

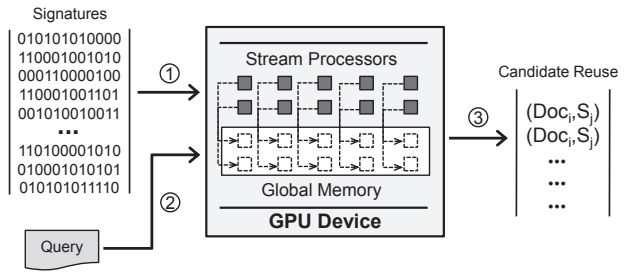


Figure 3: Implementation of GPU Based Parallel Candidate Search

4. EXPERIMENTS

4.1 Collections

We evaluate the proposed method with six corpora TIPSTER (Volume 1-3)[1], ClueWeb09-T09B[2], Tweets2011 Twitter [3], SogouT 2.0[4], Baidu Zhidao[5], Sina Weibo[6]. Table 1 shows the statistics of the six collections. TIPSTER collection contains news articles, discourse passages extracted from Associated Press (AP), Wall Street Journal (WSJ) and so on. It has been used for evaluations of information retrieval, entity extraction, and many other tasks. Tweets2011 Twitter collection is used by Trec 2011 microblog track, which contains about 16 million tweets sampled between January 23rd and February 8th, 2011 [7]. TREC Category B dataset (ClueWeb09-T09B), which is a subset of the ClueWeb09, contains 50 million English pages and has been used in various TREC tracks. Since the proposed approach is language independent, we also evaluate the proposed approach on three Chinese collections. Baidu Zhidao is one of the most popular community Q&A site in China. We crawled a portion of question and answer pairs of all categories from it, resulting in a local archive of about 33.5 million questions. Sina Weibo is the most popular and the largest Twitter-like micro-blog site in China. Messages or comments from approximate 1.78 million users are used in this work.

4.2 Implementation and Setup

We set the signature lengths(m) to 32 in this work. Following the parameters used in [15], the l is set 2 for 32 bits signature. For English collections, words are used as the basic units to assign hash codes. The basic units of Chinese collections are characters. We re-implemented the baseline `qSign` algorithm [15] using the MD5 [22] as hash function to assign hash codes for all the words/characters. It is labeled as "*MD5*" in the following tables and figures. To verify the proposed properties that good hash code should satisfy, we construct another hash code generation baseline, which set the same hash code for the words which often oc-

[1]http://www.ldc.upenn.edu/
[2]http://boston.lti.cs.cmu.edu/Data/clueweb09/
[3]http://trec.nist.gov/data/tweets/
[4]http://www.sogou.com/labs/dl/t.html
[5]http://zhidao.baidu.com
[6]http://www.weibo.com
[7]Since the data is crawled by ourselves with the tools and tweet lists provided by NIST, about 5.8% tweets are missed in our collection due to the user name change or other reasons.

Table 2: Detailed impact of different hash code generation methods. The reuse threshold θ is set to 0.8.

bits diff.	MD5			NER			OPT		
	#can.	P	R	#can.	P	R	#can.	P	R
$d = 0$	19,087	0.239	0.896	8,881	0.514	0.896	4,704	0.965	0.891
$d = 1$	17,2464	0.027	0.928	64,361	0.073	0.929	7,262	0.648	0.924
$d = 2$	992,403	0.005	0.965	416,552	0.012	0.968	27,060	0.182	0.966
$d = 3$	3,955,643	0.001	0.992	1,889,988	0.002	0.994	139,808	0.036	0.988
$d = 4$	12,049,613	0.000	1.000	6,472,842	0.001	1.000	632,570	0.008	0.998
$d = 5$	29,990,264	0.000	1.000	17,857,693	0.000	1.000	2,358,244	0.002	1.000

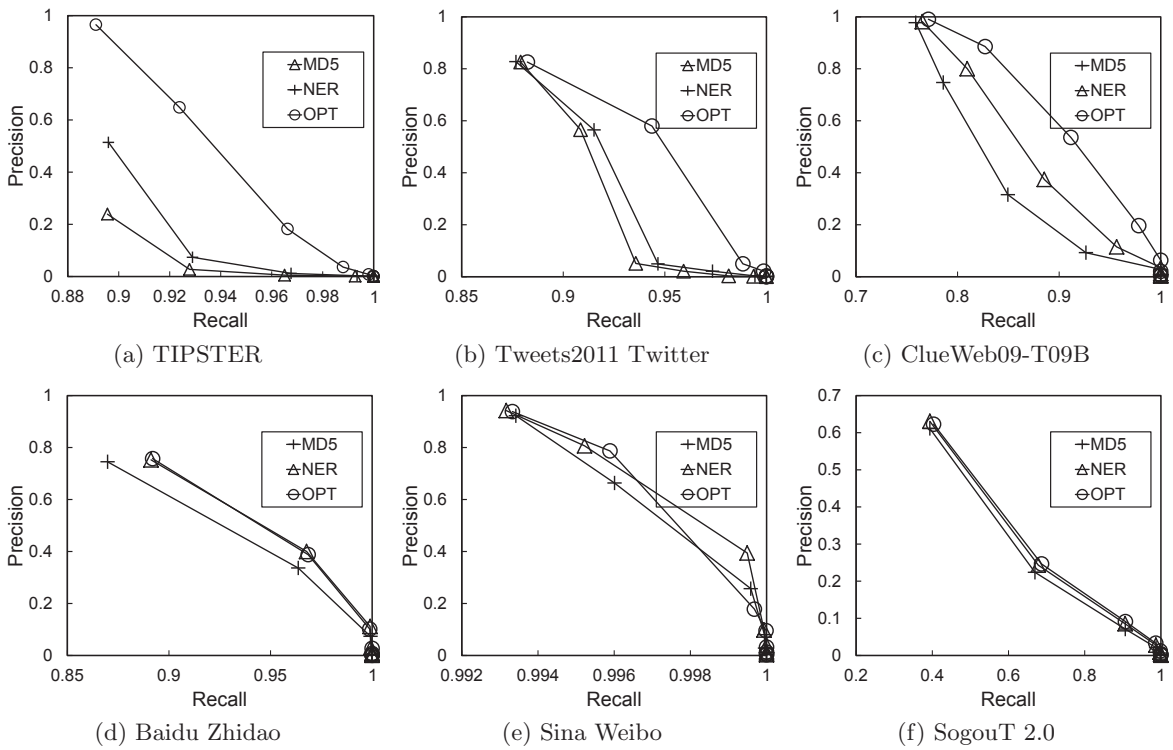
(a) TIPSTER

The total number of ground truth reuse sentences for 2,000 queries is 5,095.

bits diff.	MD5			NER			OPT		
	#can.	P	R	#can.	P	R	#can.	P	R
$d = 0$	15,686	0.611	0.394	15,205	0.631	0.394	15,773	0.623	0.403
$d = 1$	72,832	0.224	0.669	67,964	0.243	0.678	67,846	0.247	0.687
$d = 2$	310,708	0.071	0.907	258,559	0.085	0.906	242,141	0.091	0.907
$d = 3$	1,146,414	0.021	0.987	862,317	0.027	0.987	727,063	0.033	0.986
$d = 4$	3,775,472	0.006	0.998	2,654,901	0.009	0.998	2,001,083	0.012	0.998
$d = 5$	11,025,379	0.002	0.999	7,574,689	0.003	0.999	5,306,237	0.004	0.999

(b) SogouT 2.0

The total number of ground truth reuse sentences for 2,000 queries is 24,359.



(a) TIPSTER (b) Tweets2011 Twitter (c) ClueWeb09-T09B

(d) Baidu Zhidao (e) Sina Weibo (f) SogouT 2.0

Figure 4: The precision-recall curves of candidate searching based on different hash code generation methods in all six corpora. The reuse threshold θ is set to 0.8.

Table 1: Statistics of the evaluation document collections

Corpus	Language	#Docs	Size
TIPSTER	English	1,078,925	3.25GB
Tweets2011 Twitter	English	15,204,939	2.13GB
ClueWeb09-T09B	English	50,220,423	490.4GB
Baidu Zhidao	Chinese	33,497,107	22.8GB
Sina Weibo	Chinese	267,612,493	418.6GB
SogouT 2.0	Chinese	37,205,218	558.0GB

cur together. "*NER*" is used to represent this method. For the proposed approach, although we have proposed several methods to reduce the computing consumption of the optimization problem, there would also be too many variables needed to optimize. In this work, we select top 3,000 words/characters to optimize according to their frequency. The similarity matrix is calculated based on 300,000 sentences extracted from each collection. The hash codes of the other words are generated based on MD5. We use "*OPT*" to represent this method in the following.

All the experiments were evaluated on a workstation with a 2.13G Intel Xeon quad-core processor, 4GB memory, and an NVIDIA Quadro 4000 graphics card with 2GB global memory and 256 stream processors. CUDA Toolkit version 4.1 is used to implement the algorithm. For sentence boundary detection, we used around 50 manually written rules to do it.

4.3 Effectiveness Evaluation

To compare the candidate searching with different hash code generation methods, for each corpus, we randomly selected 1 million sentences as evaluation data sets. As reuse detection queries, 2,000 sentences are randomly selected from them. The similarities between queries and sentences in the data set are calculated using cosine coefficient. The ground truth reuse sentences are obtained by comparing queries with all sentences in corresponding data set.

Table 2 illustrates the impact of candidate searching step with different hash code generation methods. We only list the detailed result in TIPSTER and SogouT 2.0, due to space limitations. Figure 4 summarizes results of all six corpora. In Table 2, d represents the d bits difference between query and reuse sentences. The reuse threshold θ is set to 0.8. "#can." represents the number of candidate sentences, which are selected using different hash code generation methods. From the results, we can observe that candidate searching step really benefits acceleration the reuse detection system. Instead of comparing with all sentences, the step can help reduce more than 90% calculation without losing any correct reuses in most cases.

From Table 2a, we can also observe that different hash code generation methods may highly impact the performance of candidate searching step. Although the baseline method, which is also used by previous work qSign [15], already achieves good results, the simple way which merge the hash code of similar words can further give more than 50% improvement in all bits difference level. Comparing to the baseline methods, hash codes generated based on the proposed learning based method achieve the best results. At the same recall level, the number of selected candidate sentences based the hash codes generated by the proposed methods is only 2.7% to 24.6% of the sentences selected based on MD5 hash code. This indicates that the proposed method can dramatically improve the effectiveness of candidate searching.

Table 3: The average number of selected candidates per sentence at different recall level. The reuse threshold θ is set to 0.8.

Corpus	Recall Level	Gol.	MD5	NER	OPT
TIPSTER	0.89 (d=0)	2.5	9.5	4.4	**2.4**
	1.00 (d=4)	2.5	6024.8	3236.4	**316.3**
Twitter	0.88 (d=0)	1.5	1.6	1.6	**1.6**
	1.00 (d=4)	1.5	488.1	485.1	**477.4**
ClueWeb09	0.77 (d=0)	35.1	27.7	27.3	**26.9**
	1.00 (d=4)	35.1	1253.0	1006.4	**554.1**
Baidu Zhidao	0.75 (d=0)	2.6	3.0	3.0	**2.9**
	1.00 (d=3)	2.6	147.2	108.7	**94.5**
Sina Weibo	0.99 (d=0)	37.5	40.4	**39.5**	39.6
	1.00 (d=2)	37.5	145.8	95.0	**47.4**
SogouT 2.0	0.90 (d=2)	12.2	155.4	129.3	**121.1**
	1.00 (d=4)	12.2	1887.7	1327.5	**1000.5**

The precision-recall curves graph for all six collections are shown in Figure 4. The reuse threshold θ is also set to 0.8 in all the experiments for this figure. In almost all cases, the proposed approach achieves the best result among the three hash code generation methods. These results also demonstrate the observations described in the previous section. Although from the view precision-recall curve graph the precision improvement can not be easily noticed, at a recall level of 0.999, the proposed approach can further reduce around 51.9% sentences in SogouT 2.0, 55.4% in Sina Weibo and 27.7% in Baidu Zhidao over baseline method.

Table 3 shows the detailed performance of candidate searching step at different recall level in all six collections. The "Gol." column represents the average number of ground truth reuses per sentences. Since the recall level can only be controlled by the bits difference threshold, we list the parameter d in the bracket in the third column. From the table, we can observe that quotations are common in Web collections. While news articles also contain a large number of exact quotations. At almost all recall levels, the proposed hash code generation method achieves the best performance. It means that the proposed method can benefit the performance of candidate searching step.

In the above experiments, the documents used to calculate similarities between words for our approach are random selected from each collection. Since in-domain data may not be pre-collected in some cases, in this experiment we evaluate the performance of candidate selection with hash codes

Table 4: The average number of selected candidates per query sentence with in-domain and out-of-domain data. The reuse threshold θ is set to 0.8. The recall level is set to 100%, the corresponding bits difference d are in the bracket. "OPT_IN" represents hash codes generated based on in-domain data. "OPT_OUT" represents hash codes generated based on out-of-domain data.

Corpus	MD5	OPT_IN	OPT_OUT
TIPSTER (d=4)	6024.8	316.3	5482.1
Twitter (d=4)	488.1	477.4	492.1
ClueWeb09 (d=4)	1253.0	554.1	796.4
Baidu Zhidao (d=3)	147.2	94.5	88.1
Sina Weibo (d=2)	145.8	47.4	49.1
SogouT 2.0 (d=4)	1887.7	1005.5	1228.9

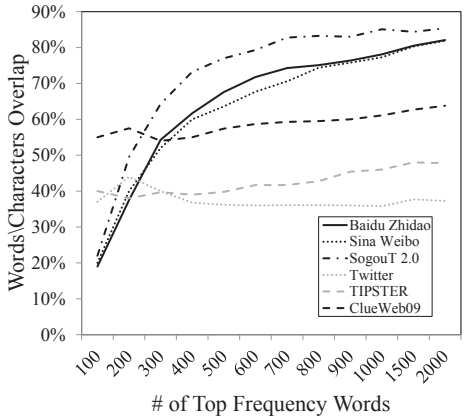


Figure 5: The overlap percentage of top frequency words/characters between corpora and "Web 1T 5-gram Corpus" or "Chinese Web 5-gram".

generated with similarities matrix calculated from open domain data sets. We use "Web 1T 5-gram Corpus[8]" and "Chinese Web 5-gram[9]" as the open domain data, where 5-grams are treated as sentences. They contain English and Chinese word n-grams and their observed frequency counts. Table 4 shows the performance candidate selection with these corpora. From the results, we can observe that in-domain data performs better than out-of-domain's in most cases. Out-of-domain data works better in Chinese than in English. For the corpus Baidu Zhidao, the performance of hash codes generated based on 5-grams is even better than in-domain data. We think that the main reason is that the words distributions in "Chinese Web 5-gram" are similar as Baidu Zhidao, which also contains documents from multiple domains. For English corpora, the performances of out-of-domain are not good as in-domains. In order to find the reason, we analysis the overlap of top frequency words. Figure 5 shows the overlap percentage of top frequency words/characters between corpora and "Web 1T 5-gram Corpus" or "Chinese Web 5-gram". From the statistics, we can observe the reason why out-of-domain data perform worse in English corpora than Chinese corpora. Distributions of words play an important role for the quality of generated hash codes from different domains.

4.4 Efficiency Evaluation

Due to the increasingly growing data, efficiency is another important issue we focused on in this paper. In this subsection, we compare the running time of our approach with state-of-the-art systems. We note that the running time of our approach composes of two steps: candidate selection and post-processing. Candidate selection step can be further divided into two steps: sentence feature generation and range searching. Because feature generation time for sentences are equal in different methods and the calculation consumption is small, we only evaluate the time of different range searching methods. To evaluate the impact of the number of selected candidates with different hash codes, we also evaluate the post-processing time.

Table 5 shows the running time of three different range searching methods. Brute force, which directly calculates all the distances between query and reference sentences, is implemented using a single thread CPU implementation and GPU implementation. We also adopt PM-Tree[10] [26] which is an indexing technique for efficient similarity searching. From the results, we can observe that indexing technique can improve searching efficiency. However, brute force method with GPU implementation can even achieve more than 1500 times speedup. Further more, the brute force methods do not need the index construction time. We think that the high memory bandwidth of GPU and naturally parallel algorithm are the main reason of the success of GPU implementation. Figure 6 shows the running time of GPU based candidate searching. We can observe that the processing time along the y axis increases as a linear function of the size of collection.

Figure 7 shows the post processing time at different recall level. We select two corpora "TIPSTER" and "SogouT 2.0" to evaluate the time using different hash codes. Figure 8 shows the post processing time with different hash codes generation methods. From these results, we can observe that the proposed hash code generation method can benefit the execution time of post processing. The number of selected candidates through different hash code generation methods impacts the execution time. Since our proposed method can filter more negative candidates than other methods, the running time are reduced.

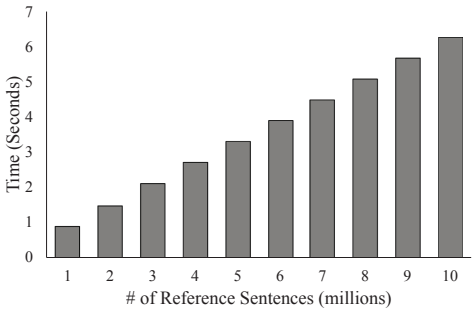


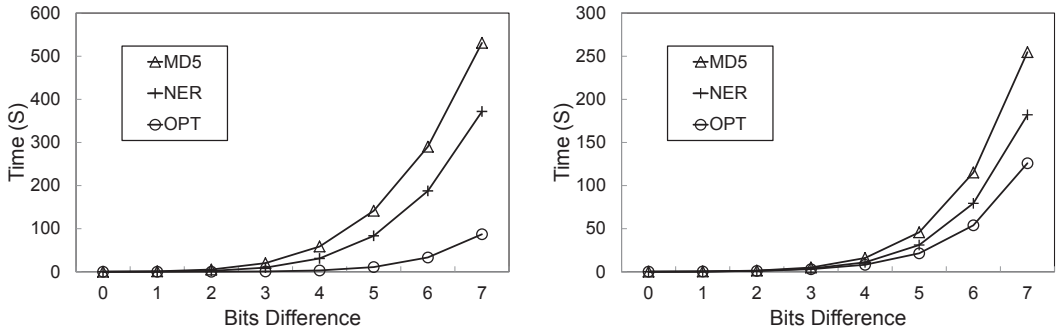
Figure 6: Total execution time of candidate selection implemented with GPU for 2,000 queries.

[8]LDC Catalog No. LDC2006T13
[9]LDC Catalog No. LDC2010T06

[10]We use the source code provided by Tomás Skopal. We set *pivot* to 20, and *pageSize* to 2048 in the experiments.

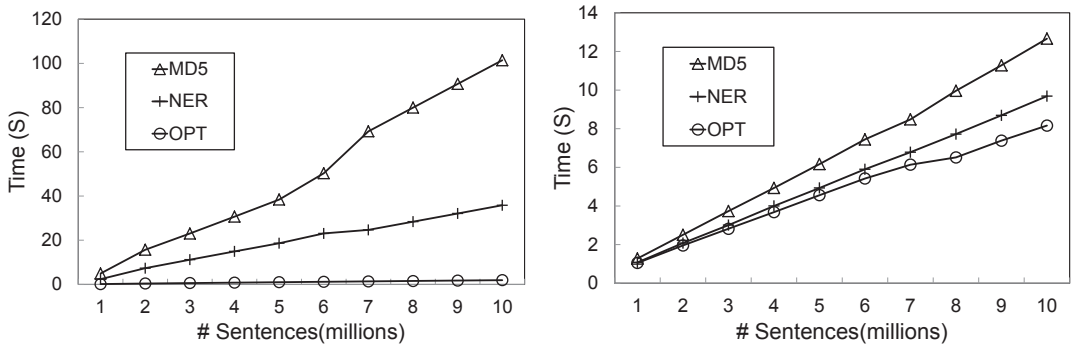
Table 5: Candidate selection time (seconds) for 2,000 queries using different range searching methods.

	0.1M	0.2M	0.3M	0.4M	0.5M	0.6M	0.7M	0.8M	0.9M	1M
Brute force (CPU)	468	939	1406	1873	2339	2807	3279	3749	4216	4685
PM-Tree (CPU)	152	297	433	569	704	852	979	1124	1258	1401
Brute force (GPU)	0.39	0.44	0.49	0.55	0.61	0.67	0.73	0.79	0.85	0.91



(a) TIPSTER (b) SogouT 2.0

Figure 7: The post processing time with different hash codes generation methods varies with recall level. The reuse threshold θ is set to 0.8. Each corpus contains 1 million sentences.



(a) TIPSTER (b) SogouT 2.0

Figure 8: The post processing time with different hash codes generation methods varies with corpus size. The reuse threshold θ is set to 0.8. The number of bits difference is set to 3.

5. CONCLUSIONS

In this work, we propose a novel approach which improves the efficiency and effectiveness of content reuse detection in two aspects. We introduce learning to hash method for generating hash codes of words/characters. We also propose to use GPU implementation to speedup the range searching task, which is the most time consuming part in candidate selection step. We evaluate the proposed approach in six different kinds of documents collections. Experimental results show that our method can significantly improve the efficiency of content reuse detection and would not impact the recall any more.

6. ACKNOWLEDGEMENT

The author wishes to thank the anonymous reviewers for their helpful comments. This work was partially funded by 973 Program (2010CB327900), National Natural Science Foundation of China (61003092, 61073069), Shanghai Leading Academic Discipline Project (B114), and "Chen Guang" project supported by Shanghai Municipal Education Commission and Shanghai Education Development Foundation (11CG05).

7. REFERENCES

[1] J. Aberdeen, J. Burger, D. Day, L. Hirschman, P. Robinson, and M. Vilain. Mitre: description of the alembic system used for muc-6. In *Proceedings of MUC6*, pages 141–155, Morristown, NJ, USA, 1995.

[2] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 131–140, 2007.

[3] J. Bolz, I. Farmer, E. Grinspun, and P. Schröoder. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. *ACM Trans. Graph.*, 22:917–924, July 2003.

[4] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of SEQUENCES 1997*, page 21, Washington, DC, USA, 1997. IEEE Computer Society.

[5] A. Z. Broder. Identifying and filtering near-duplicate documents. In *Proceedings of COM 2000*, pages 1–10, London, UK, 2000. Springer-Verlag.

[6] B. Bustos, O. Deussen, S. Hiller, and D. Keim. A graphics hardware accelerated algorithm for nearest neighbor search. In V. Alexandrov, G. van Albada, P. Sloot, and J. Dongarra, editors, *Computational Science ÍC ICCS 2006*, volume 3994 of *Lecture Notes in Computer Science*, pages 196–199. Springer Berlin / Heidelberg, 2006.

[7] A. Chowdhury, O. Frieder, D. Grossman, and M. C. McCabe. Collection statistics for fast duplicate document detection. *ACM Trans. Inf. Syst.*, 20(2):171–191, 2002.

[8] L. Dugan. 230 million tweets per day, 50 million daily users and other twitter stats. *WWW.MEDIABISTRO.COM*, 2011.

[9] S. Edelkamp, D. Sulewski, and C. YÍźcel. Perfect hashing for state space exploration on the gpu. In *ICAPS*, pages 57–64. AAAI, 2010.

[10] K. Fatahalian, J. Sugerman, and P. Hanrahan. Understanding the efficiency of gpu algorithms for matrix-matrix multiplication. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, HWWS '04, pages 133–137, 2004.

[11] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB '99*, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[12] N. K. Govindaraju, M. Henson, M. C. Lin, and D. Manocha. Interactive visibility ordering and transparency computations among geometric primitives in complex environments. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*, I3D '05, pages 49–56, 2005.

[13] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

[14] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18:1527–1554, July 2006.

[15] J. W. Kim, K. S. Candan, and J. Tatemura. Efficient overlap and content reuse detection in blogs and online news articles. In *Proceedings of the 18th international conference on World wide web*, WWW '09, pages 81–90, 2009.

[16] A. Kołcz, A. Chowdhury, and J. Alspector. Improved robustness of signature-based near-replica detection via lexicon randomization. In *Proceedings of SIGKDD 2004*, pages 605–610, 2004.

[17] J. Krüger and R. Westermann. Linear algebra operators for gpu implementation of numerical algorithms. In *ACM SIGGRAPH 2005 Courses*, SIGGRAPH '05, 2005.

[18] D. Metzler, Y. Bernstein, W. B. Croft, A. Moffat, and J. Zobel. Similarity measures for tracking information flow. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 517–524, 2005.

[19] K. Muthmann, W. M. Barczyński, F. Brauer, and A. Löser. Near-duplicate detection for web-forums. In *IDEAS '09*, pages 142–151, New York, NY, USA, 2009. ACM.

[20] M. Norouzi and D. Fleet. Minimal loss hashing for compact binary codes. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 353–360, June 2011.

[21] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips. GPU Computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

[22] R. Rivest. The md5 message-digest algorithm, 1992.

[23] J. Seo and W. B. Croft. Local text reuse detection. In *SIGIR '08*, pages 571–578, New York, NY, USA, 2008.

[24] N. Shivakumar and H. Garcia-Molina. Scam: A copy detection mechanism for digital documents. In *Digitial Library*, 1995.

[25] N. Shivakumar and H. Garcia-Molina. Finding near-replicas of documents and servers on the web. In *Proceedings of WebDB 1998*, pages 204–212, London, UK, 1999. Springer-Verlag.

[26] T. Skopal, J. Pokorný, and V. Snásel. Pm-tree: Pivoting metric tree for similarity search in multimedia databases. In *ADBIS (Local Proceedings)*, 2004.

[27] B. Stein. Principles of hash-based text retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 527–534, 2007.

[28] M. Theobald, J. Siddharth, and A. Paepcke. Spotsigs: robust and efficient near duplicate detection in large web collections. In *SIGIR '08*, pages 563–570, New York, NY, USA, 2008. ACM.

[29] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, Mar. 2006.

[30] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 1753–1760. MIT Press, 2008.

[31] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *Proceeding of the 17th international conference on World Wide Web*, WWW '08, pages 131–140, 2008.

[32] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 18–25, 2010.

[33] Q. Zhang, Y. Zhang, H. Yu, and X. Huang. Efficient partial-duplicate detection based on sequence matching. In *Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '10, pages 675–682, 2010.