Continuous Word Embeddings for Detecting Local Text Reuses at the Semantic Level

Qi Zhang, Jihua Kang, Jin Qian, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, P.R.China
{qz, 12201240059, 12110240030, xjhuang}@fudan.edu.cn

ABSTRACT

Text reuse is a common phenomenon in a variety of user-generated content. Along with the quick expansion of social media, reuses of local text are occurring much more frequently than ever before. The task of detecting these local reuses serves as an essential step for many applications. It has attracted extensive attention in recent years. However, semantic level similarities have not received consideration in most previous works. In this paper, we introduce a novel method to efficiently detect local reuses at the semantic level for large scale problems. We propose to use continuous vector representations of words to capture the semantic level similarities between short text segments. In order to handle tens of billions of documents, methods based on information geometry and hashing methods are introduced to aggregate and map text segments presented by word embeddings to binary hash codes. Experimental results demonstrate that the proposed methods achieve significantly better performance than state-of-the-art approaches in all six document collections belonging to four different categories. At some recall levels, the precisions of the proposed method are even 10 times higher than previous methods. Moreover, the efficiency of the proposed method is comparable to or better than that of some other hashing methods.

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

Local Text Reuse, Word Embedding, Fisher Vector

1. INTRODUCTION

With the rapid expansion of the World Wide Web, digital information has become much easier to access, modify, and duplicate. Text reuse, which is the practice of using existing content without modification or with few modifications, has become a common phenomenon on the Web. Rather than reusing a whole document, sentences, facts or passages were also often reused and modified, especially in social media [28]. *Local text reuse* is usually used to refer to this kind of phenomenon. The task of detecting local text reuses is important for many applications such as information flow tracking [20], plagiarism detection [29], partial-duplicate detection [39], and so on.

Along with the increasing requirements, the task has received considerable attention in recent years [28, 39, 34, 38]. Existing works on local text reuse detection have been conducted from different perspectives. However, we argue that most of them worked on the lexical level. The semantic similarities between text segments were not considered. Let us consider the following examples:

S1. The company **announced** two directors.
S2. The company **appointed** two directors.
S3. The company **fired** two directors.

From these examples, we can observe that although only one word differs between the three sentences, sentence *S3* should not be considered as the local text reuses of sentence *S1* and sentence *S2*. The meaning expressed by *S3* is almost opposite to *S1* and *S2*'s. Since existing methods are usually based on lexical level similarities, this kind of issue cannot be well addressed by these methods.

Inspired by the success of continuous space word representations in capturing the semantic similarities in various natural language processing tasks, we propose in this work to incorporate continuous space word representations, Fisher kernel framework, and hashing methods to generate hash codes for short text segments, while preserving semantic level similarities. The generated hash codes can be used to detect local reuses more efficiently and effectively. Due to the ability of continuous space word representations (also called "word embeddings") at capturing syntactic and semantic regularities, we firstly transform words in a text segment into continuous vector representations by looking up tables. These word representations were learned in advance using a feed-forward neural network language model [3], continuous skip-gram model [22], or other continuous space word representation learning methods. Then, the variable sizes of word embedding sets will be aggregated into a fixed length vector, Fisher Vector (FV), based on the Fisher kernel framework [27]. Since FVs are usually high-dimensional and dense, it makes the system less efficient for large-scale applications. Hence, the final step of the proposed framework is to compress FVs to binary hash codes using hashing methods [6, 36, 26]. Through these steps, the proposed method can map text segments into compact binary hash codes, while preserving the similarities at the semantic level. Hence, it can efficiently and effectively achieve the local text reuse detection problem. Through evaluating on six large collection belonging to four different categories, experimental results demonstrate that the proposed framework can achieve better performance than state-of-the-art approaches.

The main contributions of this work are summarized as follows.

- We propose and study the task of detecting local text reuse at the semantic level. To the best of our knowledge, this is the first work to focus on this problem.
- Inspired by the advantages of continuous space word representations, we introduce a novel method to aggregate and compress the variable-size word embedding sets to binary hash codes through Fisher kernel and hashing methods.
- We manually constructed six evaluation datasets of four different kinds categories. Experimental results show that the proposed method are significantly better than existing methods.

2. RELATED WORK

Our approach relates to the following three research areas: text reuse detection, learning to hash, and fisher kernel framework. In this section, we briefly describe the related works on these areas.

2.1 Text Reuse Detection

The task of detecting text reuse has received considerable attentions in recent years. Previous works studied the problem from different aspects such as fingerprint extraction methods with or without linguistic knowledge, hash codes learning methods, different reuse granularity, and so on.

Broder [4] proposed Shingling method, which 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. [32] introduced a method, SpotSigs. It provides more semantic pre-selection of shingles for extracting characteristic signatures from Web documents. 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.

I-Match [7] is one of the methods using hash codes to represent input document. It filters the input document based on collection statistics and compute a single hash value for the remainder text. If the documents have 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 are respect to small modifications, Kołcz et al. [18] proposed the solution of several I-Match signatures, all derived from randomized versions in the original lexicon.

Local text reuse detection focused on identifing the reused and modified sentences, facts or passages, rather than whole documents. Seo and Croft [28] analyzed the task and defined six categories of text reuses. They proposed a general framework for text reuse detection. Several fingerprinting techniques under the framework were evaluated under the framework. Zhang et al. [39] also studied the partial-duplicate detection problem. 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 Map-Reduce jobs. Kim et al. [17] proposed to map sentences into points in a high dimensional space and leveraged range searches in this space. They used MD5 hash function to generate hash code for each word. File signature is then created by taking the bitwise-or of all signatures of words that appear in the file.

Different with these existing methods, in this paper, we propose to use aggregated word embeddings to capture the semantic level similarities to reduce the false matches.

2.2 Learning to Hash

Due to the ability of solving similarity search in high dimensional space, hash-based methods have received much more attention in recent years. Extensive works on similarity search have been proposed to find good data-aware hash functions using machine learning techniques.

Hinton and Salakhutdinov [10] proposed to train a multilayer neural network with a small central layer to convert high-dimensional input vectors into low-dimensional codes. They used a two-layer network called a Restricted Boltzmann machine (RBM) to do it. Spectral hashing [36] was defined to seek compact binary codes in order to preserve the semantic similarity between codewords. They 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 [24] 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 consideration. Self-Taught Hashing (STH) [37] divided the hash codes learning problem into two stages. Firstly, they used unsupervised method, binarised Laplacian Eigenmap, to optimize l-bit binary codes for all documents in the given corpus. The classifiers were trained to predict the l-bit code for unseen documents.

Besides the works on document level, based on the qSign framework [17], Zhang et al. [38] proposed a method to optimize hash codes of words/characters rather than directly use MD5 for sentence level reuse detection. They also proposed to use GPU to accelerate the Hamming distance calculation. Wang et al. [35] proposed to use hashing methods to address tag completion and prediction problem. They used discrete optimization to construct the hash codes for both data examples and tags. The constructed the hash codes for the observed tags are consistent.

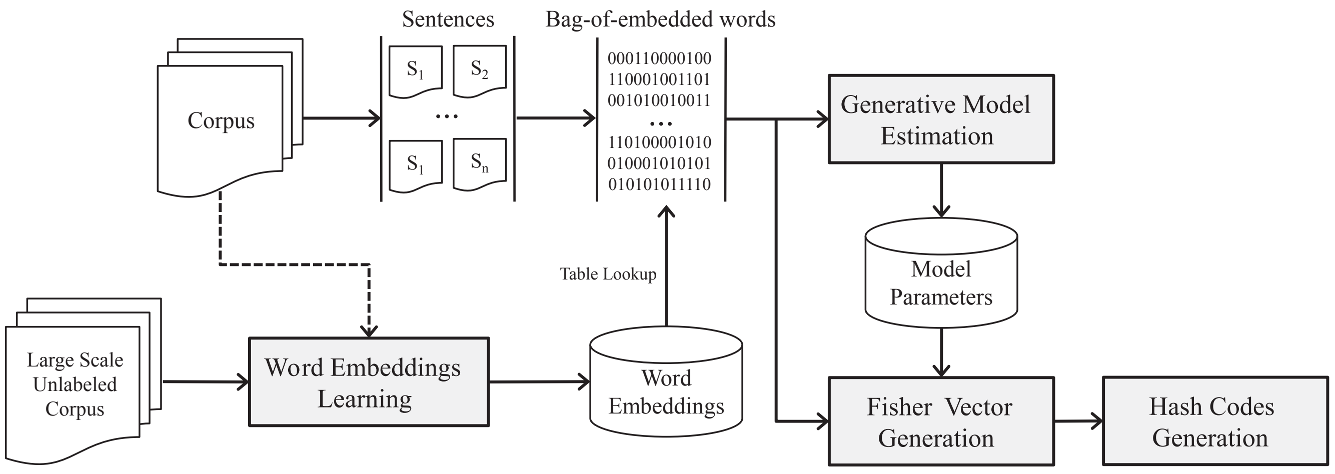


Figure 1: An overview of the proposed cFV-BoEW method for local text reuse detection.

2.3 Fisher Kernel

Fisher Kernel (FK) introduced by Jaakkola and Haussler [15] combines the benefits of generative and discriminative approaches for classification through the kernel derived from a generative probability models. It has been successfully applied to many tasks [14, 23, 12, 31, 8].

Jaakkola et al. [14] proposed to use the derived kernel function from HMMs corresponding to the protein family of interest to detect remote protein homologies. In [23], Moreno et al. explored their work in using Fisher kernel methods for audio classification problem. They proposed to use FK to map audio sequences into a fixed length representation and then use discriminative models to classify the data. Hofmann proposed a FK based method to calculate the similarities between documents. Each document is modeled as a memoryless information source. Fisher kernel is derived from the learned multinomial subfamily. Sun et al. [31] introduced a LDA-based Fisher kernel method to achieve text segmentation task. They proposed to use Latent Dirichlet Allocation to compute words semantic distribution and use Fisher kernel to measure semantic similarities.

The most similar work to ours is the recent work by Clinchant and Perronnin [8]. They proposed to use Fisher kernel to aggregate word embeddings to represent documents. They studied the relationship with LSI, pLSA, and LDA. They also evaluated the representation method through clustering and retrieval. Different with them, in this paper, we study the FK framework in representing the short text segments for the local text reuse detection. To overcome the efficiency problem, we also propose to incorporate hashing methods to map Fisher vectors to binary hash codes.

3. THE PROPOSED METHOD

The processing flow of the proposed method is shown in Figure 1. Given a collection of documents, sentences or short text segments are treated as the basic processing units. Through table lookup, all the words in a sentence are transformed to continuous vectors generated by the *word embeddings learning* step. The dashed arrow between corpus to word embeddings learning represents that in-domain corpus can also be included into the training data. Based on the learned word embeddings, sentences are represented by variable-size sets of word embeddings. *Generative model estimation* step, which is the generative part in the FK framework, tries to estimate the parameters for the models used to describe the data. *Fisher Vector generation* step uses the estimated parameters of generative model and bag-of-embedded words to generate FVs for all the sentences. After that, learning to hash methods are used in the *hash codes generation* step to transfer the dense FVs to hash codes. Semantic level local text reuses can be detected through calculating the hamming distances between these hash codes.

From the framework, we can see that although word embeddings computation and generative model estimation steps are time consuming, they run only once in advance. Meanwhile, the computational requirements of FV generation and hash code generation are limited. Hence, the proposed framework can efficiently achieve the local reuse detection task at the semantic level. In the following of this section, we detail the main steps of the proposed framework.

3.1 Word Embeddings Learning

Representation of words as continuous vectors recently has been shown to benefit performance for a variety of NLP and IR tasks [11, 33, 30]. Similar words tend to be close to each other with the vector representation. Moreover, Mikolov et al. [21] also demonstrated the learned word representations could capture meaningful syntactic and semantic regularities. Hence, in this work, we propose to use word embeddings to capture the semantic level similarities between short text segments.

Fig. 2 shows three architectures used for learning word embeddings. w_i represents the ith words in the given words sequence $\{w_1, w_2, ..., w_T\}$. Fig. 2(a) shows the architecture of the probabilistic neural network language model (NNLM) proposed by Bengio et al. in [3]. It can have either one hidden layer beyond the word features mapping or direct connections from the word features to the output layer. They also proposed to use $softmax$ function for the output layer to guarantee positive probabilities summing to 1. The word vectors and the parameters of that probability function

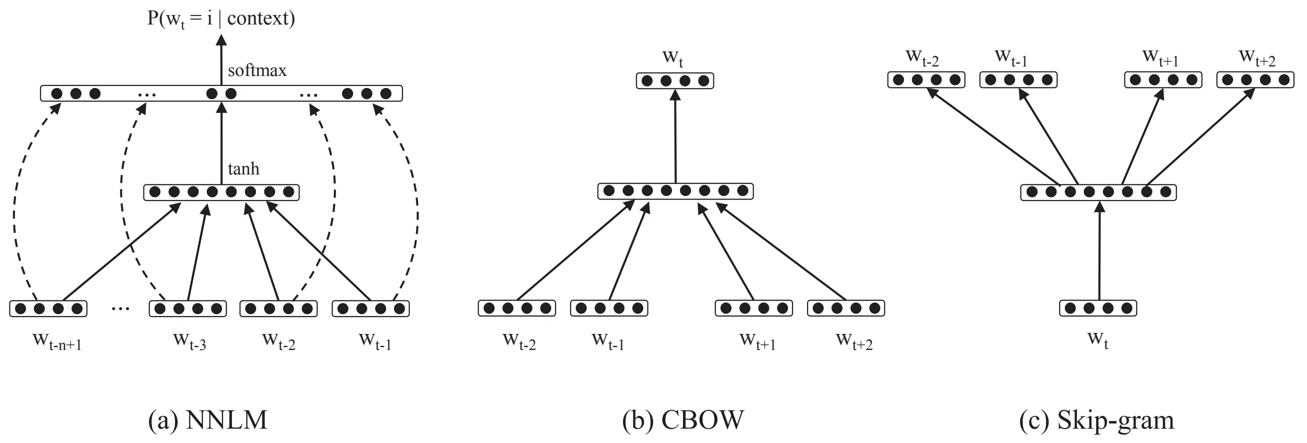


Figure 2: Methods used to learn word embeddings. The NNLM architecture predicates the probability of words based on the existing words [3]. CBOW predicts the current word based on the context [21]. Skip-gram predicts surrounding words given the current word [21].

can be learned simultaneously. In this work, we only use the learned word vectors.

Fig. 2(b) and Fig. 2(c) show the architectures of the methods proposed by Mikolov in [21]. The architecture of CBOW, which is similar to NNLM, is shown in Fig. 2(b). The main differences are that (i) the non-linear hidden layer is removed; (ii) the words from the future are included; (iii) the training criterion is to correctly classify the current (w_t) word. The Skip-gram architecture, which is shown in Fig. 2(c), is similar to CBOW. However, instead of predicting the current word based on the history and future words, it tries to maximize classification accuracy of words within a certain range before and after the current word based on only the current word as input.

Besides the methods mentioned above, there are also a large number of works addressing the task of learning distributed word representations [33, 19, 13]. Most of them can also be used in this work. The proposed framework has no limits in using which of the continuous word representation methods.

3.2 Fisher Vector

Base on the learned word embeddings, short text segments can be represented by variable length sets of word embeddings, which can be viewed as Bag-of-Embedded-Words (BoEW) [8]. Semantic level similarities between text segments represented by BoEW can be captured more accurately than previous BoW methods. However, since BoEWs are variable-size sets of word embeddings and most of the index methods are not suitable for this kinds of issues, BoEWs cannot be directly used for large scale problems. Inspired by the success of Fisher Kernel framework in various image processing techniques, we propose to adopt it to aggregate BoEW to a fixed-length vector.

Given a corpus $D = \{d_i, 1 \le i \le |D|\}$, where d_i is the ith text segment and $|D|$ is the number of text segments in the corpus. The ith text segment d_i is composed by a sequence of text words $w_i = \{w_{ij}, 1 \le j \le N_i\}$, where N_i represents the length of d_i. Through table lookup, the ith text segment d_i can be represented by $ew_i = \{ew_{ij}, 1 \le j \le N_i\}$, where ew_{ij} is the word embedding of w_{ij}. According to the framework of Fisher Kernel, text segments are modeled by a probability density function. In this work,we use Gaussian mixture model (GMM) to do it. We assume that the continuous word embeddings ew_i in the text segment d_i are generated by GMM, which is denoted as u_λ in the following. $\lambda = [\lambda_1, \lambda_2, ..., \lambda_K]'$ denotes the vector of K parameters of u_λ. The ith vector component in GMM is characterized by normal distributions with weights ϕ_i, means μ_i and covariance matrices Σ_i. Hence, in this work, λ includes $\{\omega_i, \mu_i, \Sigma_i, 1 \le i \le K\}$. The parameters λ usually can be estimated through the optimization of Maximum Likelihood (ML) criterion using Expectation Maximization (EM) method. In the following of this section, we follow the notations used in [27].

Based on the learned u_λ, the text segment d_i can be characterized using the following score function:

$$G_\lambda^{d_i} = \nabla_\lambda^{N_i} \log u_\lambda(d_i), \tag{1}$$

where $G_\lambda^{d_i}$ is a vector whose size is only dependent on the number of parameters in λ and not the number of words in the text segment. The gradient describes the contribution of each individual parameters to the generative process.

According to the theory of information geometry [1], $\mathscr{U} = \{u_\lambda, \lambda \in \Lambda\}$, which is a parametric family of distributions, can be regarded as a Riemanninan manifold M_Λ with a local metric given by the Fisher Information Matrix (FIM) $F_\lambda \in \mathbb{R}^{M \times M}$:

$$F_\lambda = E_{d_i \sim u_\lambda}\left[G_\lambda^{d_i} G_\lambda^{d_i'}\right]. \tag{2}$$

Based on this observation, the following equation can be used to measure the similarity between two text segments d_i and d_j using the the Fisher Kernel [15]:

$$K_{FK}(d_i, d_j) = G_\lambda^{d_i'} F_\lambda^{-1} G_\lambda^{d_j}. \tag{3}$$

Since F_λ is symmetric and positive definite, F_λ^{-1} can be transformed to $L_\lambda' L_\lambda$ based on the Cholesky decomposition. Hence, $K_{FK}(d_i, d_j)$ can be rewritten as follows:

$$K_{FK}(d_i, d_j) = \mathscr{G}_\lambda^{d_i'} \mathscr{G}_\lambda^{d_j}, \tag{4}$$

where

$$\mathscr{G}_\lambda^{d_i} = L_\lambda G_\lambda^{d_i} = L_\lambda \nabla_\lambda \log u_\lambda(d_i). \tag{5}$$

Table 1: Statistics of the evaluation document collections

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

Under the assumption that ew_{ij} is sampled independently, in this work, $\mathscr{G}_\lambda^{d_i}$ can be rewritten as follows:

$$\sum_{j=1}^{N_i} L_\lambda \nabla_\lambda \log u_\lambda(ew_{ij}). \tag{6}$$

In [27], $\mathscr{G}_\lambda^{d_i}$ is also referred to as *Fisher Vector* of d_i. The dot product between Fisher vectors can be used to calculate the semantic similarities.

Based on the specific probability density function, GMM, we used in this work, FV of d_i is respect to the mean μ and standard deviation σ of all the mixed Gaussian distributions. Let $\gamma_j(k)$ be the soft assignment of the jth word embedding ew_{ij} in d_i to Gaussian k (u_k):

$$\gamma_j(k) = p(k|ew_{ij}) \frac{\omega_i u_k(ew_{ij})}{\sum_{j=1}^{K} \omega_k u_k(ew_{ij})} \tag{7}$$

Mathematical derivations lead to:

$$\mathscr{G}_{\mu,k}^{d_i} = \frac{1}{N_i\sqrt{\omega_i}} \sum_{j=1}^{N_i} \gamma_j(k) \left(\frac{ew_{ij} - \mu_k}{\sigma_k} \right) \tag{8}$$

$$\mathscr{G}_{\sigma,k}^{d_i} = \frac{1}{N_i\sqrt{2\omega_i}} \sum_{t=1}^{N_i} \gamma_j(k) \left[\frac{(ew_{ij} - \mu_k)^2}{\sigma_k^2} - 1 \right],$$

where N_i is the number of word embeddings in d_i. The division between vectors is as a term-by-term operation. The final gradient vector $\mathscr{G}_\lambda^{d_i}$ is is the concatenation of the $\mathscr{G}_{\mu,k}^{d_i}$ and $\mathscr{G}_{\sigma,k}^{d_i}$ vectors for $k = 1...K$. Let T denotes the dimensionality of the word embeddings. The final gradient vector $\mathscr{G}_\lambda^{d_i}$ is therefore 2KT-dimensional.

3.3 Hash Code Generation

Through the previoussteps, a variable length of text segments can be transferred to a fixed length vector and the semantic similarities between text segments can be preserved. However, Fisher vectors are usually high dimensional and dense. It limits the usages of FVs for large-scale applications, where computational requirement should be studied. In this work, we propose to use hashing methods to address the efficiency problem.

The task of generating hash codes for samples can be formalized as learning a mapping $b(\mathbf{x})$, referred to as a hash function, which can project p-dimensional real-valued inputs $x \in \mathbb{R}^p$ onto q-dimensional binary codes $h \in \mathcal{H} \equiv \{-1, 1\}^q$, while preserving similarities between samples in original spaces and transformed spaces. The mapping $b(\mathbf{x})$ can be parameterized by a real-valued vector $\mathbf{w}$ as:

$$b(\mathbf{x}; \mathbf{w}) = \text{sign}(f(\mathbf{x}; \mathbf{w})), \tag{9}$$

where sign($\cdot$) represents the element-wise sign function, and $f(\mathbf{x}; \mathbf{w})$ denotes a real-valued transformation from $\mathbb{R}^p$ to $\mathbb{R}^q$. In this work, Fisher vectors of text segments are the $\mathbf{x}$ in mapping function $b(\mathbf{x}; \mathbf{w})$. A variety of existing methods have been proposed to achieve this task under this framework using different forms of f and different optimization objectives. Most of the learning to hash methods for dense vectors can be used in this framework. In this work, we evaluated several state-of-the-arts hashing methods, whose performances are shown in the experiment section.

4. EXPERIMENTS

In this section, we firstly describe how we construct the collection. Then we introduce the experiment configurations and baseline methods. Finally, the evaluation results and discussions are given.

4.1 Datasets

We evaluate the proposed method on two different datasets. The first one is used in [38] (*CRD* for short in the following). It contains six collections includes: TIPSTER (Volume 1-3), ClueWeb09-T09B, Tweets2011, SogouT 2.0, Baidu Zhidao, Sina Weibo. Table 1 shows the statistics of the six collections. From the statistics, we can see that two languages (English and Chinese) and four categories (news, web page, microblog, and Q&A community) are included in the dataset. They randomly selected 1 million sentences from each collections as the evaluation dataset and 2,000 sentences as reuse detection queries. All the sentences whose similarities the query in word level are bigger than 0.8 are extracted as golden standards. In this work, we used the same corpus as them to evaluate the proposed framework.

Since the golden standards used in [38] are constructed based on only similarities in word level, semantic level similarities were not taken into consideration. In this work, we manually annotated another dataset, *CRD-S*. We randomly selected 100 queries from the original 2,000 queries in CRD for each of the collections. We calculated the cosine similarities between them and all the sentences. The sentences whose similarities with the query are higher than 0.7 were extracted as candidates for further processing. Three annotator were asked to determine whether an extracted candidate is a reuse of the query or not. To evaluate the quality of corpus, we validated the agreements of human annotations using Cohen's kappa coefficient. The average κ among all annotators is 0.637. It indicates that the annotations of the corpus are reliable. These manually annotated pairs construct the golden standards of CRD-S.

4.2 Experiment Configuration

Following the parameters used in [17] and [38], in this work, we also set the length of hash code to 32 bit. Words for both English and Chinese collections are used as the basic units for learning vector representations. Since Chinese is written without spaces between words, we use FudanNLP [25] to segment sentences into words. For generating Fisher vectors for text segments represented by word embeddings, we use INRIA's Fisher vector implementation [16]. The number of Gaussian densities in GMM is set to 32. Precision (P), Recall (R), and F1-score (F_1) are used as the evaluation metrics.

For comparing the effectiveness, the following state-of-the-art methods were also evaluated on both CRD and CRD-S datasets.

- **Semantic Hashing (SmH)**: It was proposed by Hinton and Salakhutdinov in [10]. Semantic hashing is a multilayer neural network with a small central layer to convert high-dimensional input vectors into low-dimensional codes. We use the toolkit provided by Ruslan Salakhutdinov and Geoff Hinton.[1]
- **Spectral Hashing (SpH)**: It was defined to seek compact binary codes preserving similarity using spectral relaxation to obtain an eigenvector solution [36]. We use the toolkit provided by the authors.[2]
- **Simhash (SimH)**: Charikar [6] proposed to use random projection methods to estimate the cosine similarity measure between two vectors. In this work, we re-implement the method for evaluation.
- **Expanded qSign (E-qSign)**: Zhang et al. [38] extended the qSign framework with learned signatures of words for local text reuse detection. Since we use the same evaluation dataset as them, we directly use the results reported in [38].

For SmH, SpH, and SimH methods, text segments are represented by bag-of-words model. The vocabulary construction is based on the TF·IDF scores of words. We filtered stop words and low frequency words and selected 20,000 words to construct the vocabulary.

As we mentioned in the previous section, most of word embeddings and hashing methods can be used in the proposed framework. We select some of state-of-the-art methods for evaluation. Tabel 2 shows the word representations we used in this work. Due to the high computing cost required by C&W and HLBL, we used the publicly available word embeddings provided by Collobert et al. [9][3] and Turian et al. [33][4] for English words. The word representations trained by GCNLM are also public available for English. [5] CBOW and Skip-gram represent the word embeddings estimated based on the evaluation data sets. CBOW-CL and Skip-gram-CL represent the word embeddings trained based on ClueWeb09. Since the proposed framework has no limitations about using which word embeddings learning methods and hash codes generation methods, we evaluate several combinations of them. *CBOW+SpH* presents the words embeddings learned with CBOW and compressed with spectral hashing. *Skip-gram+SimH* denotes the combination of Skip-gram and similarity hash [6].

[1]http://www.cs.toronto.edu/~hinton/
[2]http://www.cs.huji.ac.il/~yweiss/SpectralHashing/
[3]http://ml.nec-labs.com/senna/
[4]http://metaoptimize.com/projects/wordreprs/
[5]http://www.socher.org/

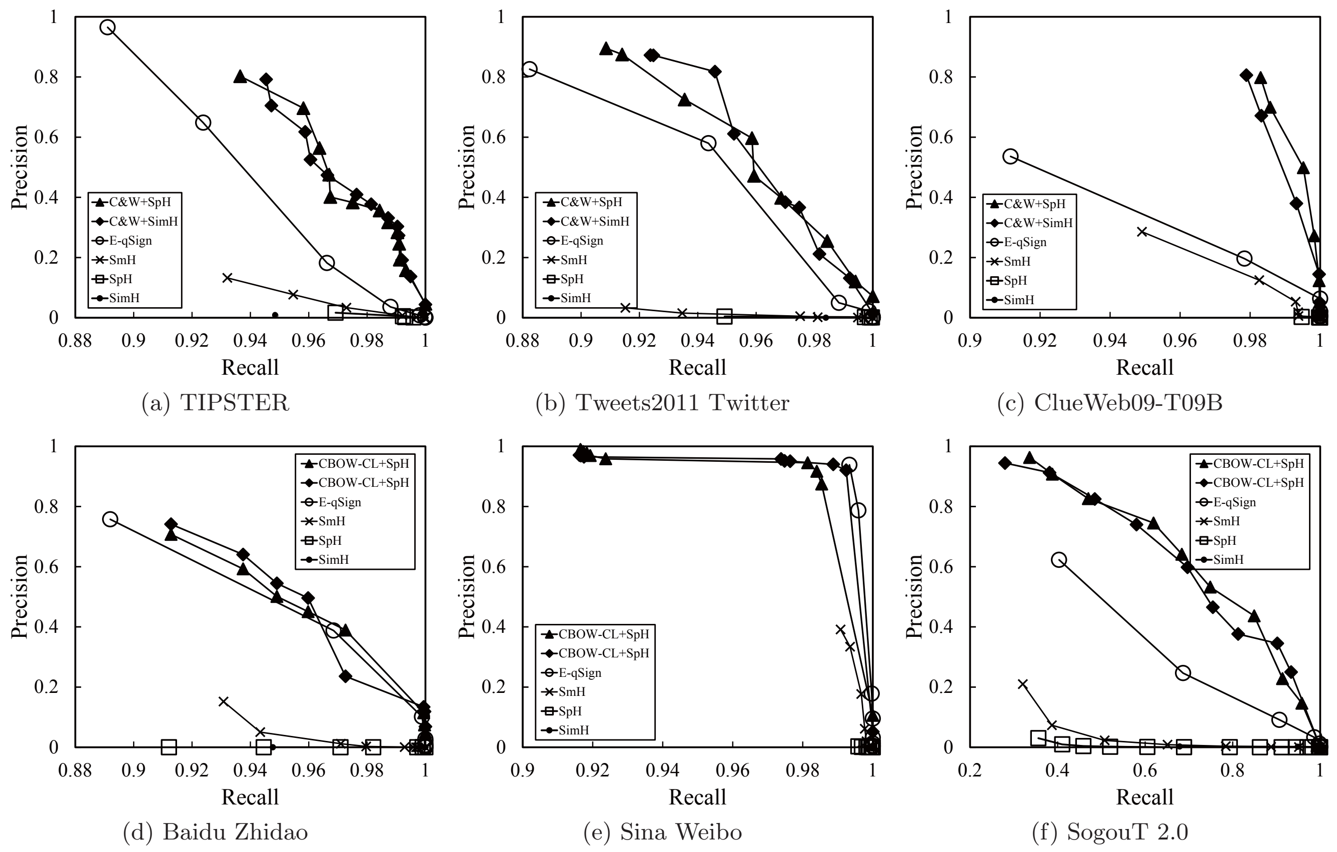
Table 2: The word embeddings used in this work. The top 8 rows of the table are English ones and the others are Chinese word embeddings.

	Dim.	Corpus	# Words
C&W [9]	50	Wikipedia	631M
C&W-T [33]	50	RCV1	63M
HLBL-T [33]	50	RCV1	63M
GCNLM [13]	50	Wikipedia	1M
CBOW [21]	50	In-Domain	–
CBOW-CL	50	Clueweb09	79M
Skip-gram [21]	50	In-Domain	–
Skip-gram-CL	50	Clueweb09	79M
GCNLM [13]	50	Sogou News	4M
CBOW [21]	50	In-Domain	–
CBOW-CL	50	Clueweb09	52M
Skip-gram [21]	50	In-Domain	–
Skip-gram-CL	50	Clueweb09	52M

4.3 Experimental Results

4.3.1 *Effectiveness evaluation*

For comparing the performances of different hash code generation methods, we firstly evaluate the proposed method and state-of-the-art methods in CRD corpus. For English collections, we use the word embeddings provided by Collobert et al. [9]. For Chinese datasets, we use CBOW-CL method, which uses ClueWeb09 as the training corpus, to generate word embeddings. Spectral hash and similarity hash were used to compact Fisher vectors of sentences into hash codes. The precision-recall curves graph of all six collections are shown in Figure 3. From the results we can see that, in all cases, both the proposed methods achieve better performance than the other hash code generation approaches. In all three English datasets, the proposed method achieves significantly better results than existing methods. Among the three Chinese datasets, the proposed methods achieve the highest relative improvement in SogouT dataset. We think that the performance loss provided by Chinese word segmentation method may be one of the main reasons. Since the most of the documents in Baidu Zhidao and Weibo are user generated, the performances of CWS toolkit are worse than in SogouT. The error of CWS may impact the generated Fisher vectors and the final results. In most of the cases, the performance of Simhash and Spectral hash are the worse than others. We think that the main reason is that the vector of short text segment is too sparse with BoW presentation. However, Spectral hash and Simhash can preserve the similarities between FVs well.



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

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

Figure 3: The precision-recall curves of different hash code generation methods in all six collections of CRD.

To detail the performance of our framework using different word embeddings and hashing methods, we select two datasets, SogouT and TISPTER, for evaluating Chinese and English respectively. Figure 4(a) and (b) give the results in TIPSTER with different hashing method to compact FVs. From the results, we can see that the proposed method with different word embeddings and hashing methods can achieve significantly better performance than previous methods in most of the cases. For TIPSTER, the word embeddings C&W generated by [9] obtain the best performance. The training data used by them is also the biggest among other word embeddings used in this work. Comparing the results shown in Figure 4 (a) and (b), we can observe that the performances of Simhash and Spectral hash are similar with each other when the same word embeddings are used. Figure 4(c) gives the results in SogouT. Because of the highly computational requirements, we evaluate GCNLM, CBOW and Skip-gram for SogouT. From the results, we can see that the proposed methods with different word embeddings and hashing method can always achieve better performance than existing methods.

As described in the previous section, in this work, we also manually annotated another dataset CRD-S to evaluate the semantic level duplicates. Table 3 illustrates the results of different methods in CRD-S. The precision and recall of different methods are shown in the table. The first column in all the tables represent the thresholds of different hamming distances. Comparing to the results in CRD, the performances of existing methods based on lexical similarities drop. However, the performances of the proposed methods are even better than it in CRD. We think that it is due to the golden standards used in CRD, which may contain false positive ones. While, testing instances were manually annotated in CRD-S Since the vectors of short text segments represented by BoW model extremely sparse, the performances of Spectral hash and Simhash are much lower than others. Due to space limited, we cannot list all the results for SogouT in Table 3(f). When the threshold of hamming distance is set to 10, the recall and precision are 96.8% and 19.6% respectively. Among all the methods, the proposed methods achieve the best result in all six collections. At the same recall level, the precisions of the proposed method are 2-10 times higher than E-qSign. It also demonstrates that incorporating the word embeddings can bring benefits for calculating the semantic level similarities.

The following hash codes are generated by CBOW-SpH for the sentences illustrated in the Section 1. The numbers

[0xDD60EFE9] S1. The company **announced** two directors.
[0xDD60EFE9] S2. The company **appointed** two directors.
[0xDD20CFE9] S3. The company **fired** two directors.

in square brackets are the generated hash codes for the sentences in the right. From the examples we can see that the hamming distance between the hashcodes of S1 and S3 is two. Although there is one word difference between S1 and S2, the hash codes of S1 and S2 are equal.

Table 3: The performances of different methods in CRD-s.

bits diff.	SmH		SpH		SimH		E-qSign		C&W+SimH		C&W+SpH	
	P	R	P	R	P	R	P	R	P	R	P	R
$d = 0$	0.125	0.962	0.024	0.968	0.017	0.969	0.831	0.907	0.806	0.966	0.806	0.966
$d = 1$	0.046	0.982	0.022	0.989	0.005	0.992	0.801	0.922	0.794	0.974	0.797	0.974
$d = 2$	0.014	0.996	0.016	0.996	0.002	0.993	0.337	0.957	0.696	0.983	0.704	0.983
$d = 3$	0.002	0.999	0.005	0.999	0.001	0.996	0.075	0.986	0.607	0.992	0.593	0.992
$d = 4$	0.001	1.000	0.001	1.000	0.001	0.999	0.016	0.998	0.559	0.993	0.528	0.993

(a) TIPSTER

bits diff.	SmH		SpH		SimH		E-qSign		C&W+SimH		C&W+SpH	
	P	R	P	R	P	R	P	R	P	R	P	R
$d = 0$	0.012	0.954	0.0030	0.949	0.0004	0.949	0.870	0.851	0.903	0.921	0.903	0.922
$d = 1$	0.011	0.982	0.0014	0.953	0.0002	0.987	0.517	0.884	0.899	0.933	0.888	0.936
$d = 2$	0.009	0.993	0.0007	0.984	0.0001	0.990	0.380	0.943	0.818	0.951	0.794	0.952
$d = 3$	0.008	0.997	0.0003	0.990	0.0001	0.999	0.321	0.985	0.814	0.968	0.791	0.970
$d = 4$	0.001	1.000	0.0001	0.998	0.0000	1.000	0.005	0.998	0.716	0.980	0.756	0.975

(b) Twitter

bits diff.	SmH		SpH		SimH		E-qSign		C&W+SimH		C&W+SpH	
	P	R	P	R	P	R	P	R	P	R	P	R
$d = 0$	0.232	0.927	0.071	0.955	0.027	0.974	0.984	0.790	0.963	0.983	0.956	0.984
$d = 1$	0.121	0.945	0.040	0.985	0.011	0.982	0.909	0.887	0.858	0.990	0.866	0.992
$d = 2$	0.076	0.980	0.024	0.991	0.005	0.990	0.744	0.971	0.687	0.997	0.628	0.997
$d = 3$	0.027	0.993	0.015	0.997	0.003	0.999	0.436	0.987	0.446	0.998	0.411	0.998
$d = 4$	0.012	1.000	0.004	1.000	0.002	1.000	0.166	0.999	0.206	1.000	0.183	1.000

(c) ClueWeb09

bits diff.	SmH		SpH		SimH		E-qSign		CBOW+SimH		CBOW+SpH	
	P	R	P	R	P	R	P	R	P	R	P	R
$d = 0$	0.152	0.899	0.015	0.942	0.014	0.948	0.667	0.907	0.753	0.922	0.741	0.922
$d = 1$	0.082	0.967	0.005	0.951	0.005	0.959	0.399	0.928	0.636	0.930	0.639	0.928
$d = 2$	0.043	0.993	0.002	0.965	0.002	0.973	0.179	0.940	0.424	0.932	0.404	0.940
$d = 3$	0.010	0.998	0.001	0.975	0.001	0.977	0.020	0.999	0.260	0.947	0.257	0.950
$d = 4$	0.005	1.000	0.001	0.998	0.000	0.998	0.001	1.000	0.168	0.996	0.162	0.995

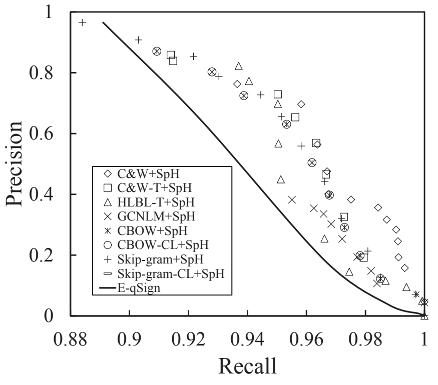
(d) Baidu Zhidao

bits diff.	SmH		SpH		SimH		E-qSign		CBOW+SimH		CBOW+SpH	
	P	R	P	R	P	R	P	R	P	R	P	R
$d = 0$	0.392	0.991	0.030	0.996	0.048	0.162	0.983	0.914	0.990	0.916	0.990	0.916
$d = 1$	0.334	0.994	0.011	0.997	0.026	0.490	0.678	0.936	0.980	0.918	0.978	0.918
$d = 2$	0.177	0.997	0.007	0.998	0.003	0.605	0.362	0.978	0.974	0.978	0.975	0.979
$d = 3$	0.062	0.998	0.004	0.999	0.004	0.692	0.215	0.999	0.969	0.979	0.964	0.983
$d = 4$	0.020	1.000	0.001	1.000	0.001	0.714	0.129	1.000	0.920	0.994	0.902	0.996

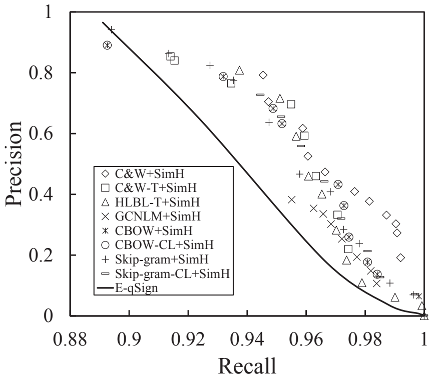
(e) Sina Weibo

bits diff.	SmH		SpH		SimH		E-qSign		CBOW+SimH		CBOW+SpH	
	P	R	P	R	P	R	P	R	P	R	P	R
$d = 0$	0.190	0.362	0.030	0.356	0.030	0.396	0.611	0.398	0.996	0.166	0.991	0.167
$d = 1$	0.087	0.507	0.011	0.411	0.009	0.421	0.228	0.691	0.979	0.387	0.977	0.363
$d = 2$	0.045	0.784	0.004	0.459	0.004	0.452	0.108	0.913	0.913	0.519	0.907	0.418
$d = 3$	0.022	0.882	0.002	0.520	0.002	0.520	0.030	0.983	0.828	0.631	0.824	0.618
$d = 4$	0.016	0.9515	0.001	0.689	0.001	0.605	0.010	0.999	0.734	0.694	0.753	0.687

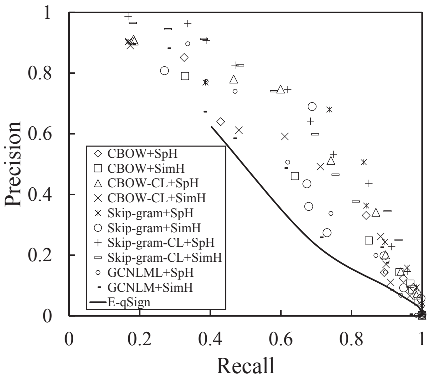
(f) SogouT 2.0



(a) TIPSTER(SpH)



(b) TIPSTER(SimH)



(c) SogouT

Figure 4: The precision-recall curves of different word embeddings in SogouT and Tispter.

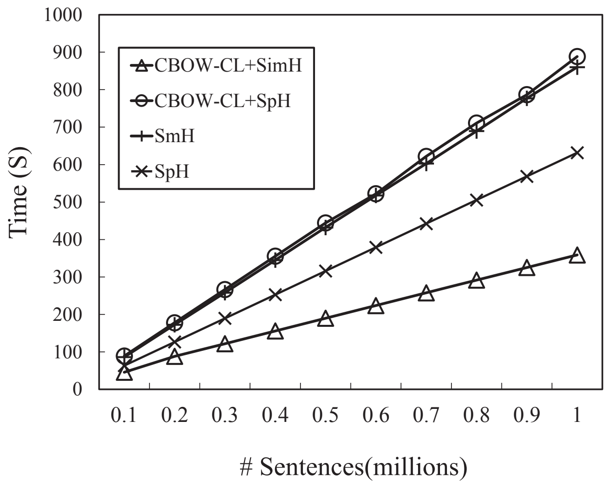


Figure 5: The efficiency comparison of different hashing methods.

It demonstrates that the proposed method can effectively preseve the semantic level similarities.

4.3.2 Efficienc Evaluation

Due to the requirement of processing huge amounts of data, efficiency is also an important issue. In this work, we compare the running time of the proposed approach with other hashing learning methods. Although the offline stage of the proposed framework requires massive computation cost, the computational complexity of online stage is small or comparable to other hashing methods. Figure 5 shows the efficiency comparison of different hashing methods. We implement the all methods to run on single thread in the same machine, which contains Xeon quad core CPUs (2.53GHz) and 32GB RAM. All the methods take the sentences as inputs. The processing time is calculated from receiving the inputs to generating hash codes. For processing out-of-sample extension of spectral hashing, we propose to use the Nystrom method [2] to do it. Since the computational cost of generating Fisher vector for different word embeddings are same, we only list the results of word embeddings CBOW-CL. From the results, we can observe that the computational complexity of the proposed method is comparable with and state-of-the hashing methods. The efficiency of CBOW-CL-SimH is even better than spectral hash and semantic hash. It demonstrates that the proposed method is applicable for large scale applications.

5. CONCLUSIONS

In this work, we study the semantic level local text reuse detection problem and introduce an novel framework to efficiently achieve the task. To capture the semantic level similarities between short text segment, we propose to use continuous vector representations of words to represent short text segments. For processing tens of billions of documents, we propose to incorporate the Fisher vectors to aggregate the variable size BoEW to represent text segments. Moreover, we use hashing methods to compress the dense and high dimensional FVs to binary hash codes. Through experiments on six different collections in both Chinese and English, we demonstrate that the proposed method can achieve better performance than state-of-the-art approaches in all collections. Besides that, the efficiency of the proposed method is comparable with most of hashing methods.

6. ACKNOWLEDGEMENT

The authors wish 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] S. Amari and H. Nagaoka. *Methods of information geometry*, volume 191. AMS Bookstore, 2000.
[2] Y. Bengio, O. Delalleau, N. Le Roux, J.-F. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel pca. *Neural Computation*, 2004.

[3] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3, Mar. 2003.

[4] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of SEQUENCES 1997*, 1997.

[5] A. Z. Broder. Identifying and filtering near-duplicate documents. In *Combinatorial Pattern Matching*, pages 1–10, 2000.

[6] M. S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of STOC '02*, 2002.

[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] S. Clinchant and F. Perronnin. Aggregating continuous word embeddings for information retrieval. August 2013.

[9] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The JMLR*, 2011.

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

[11] G. Hinton and R. Salakhutdinov. Discovering binary codes for documents by learning deep generative models. *Topics in Cognitive Science*, 2010.

[12] T. Hofmann. Learning the similarity of documents: An information-geometric approach to document retrieval and categorization. 2000.

[13] E. H. Huang, R. Socher, C. D. Manning, and A. Y. Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL'12*, 2012.

[14] T. Jaakkola, M. Diekhans, and D. Haussler. Using the fisher kernel method to detect remote protein homologies. In *ISMB*, volume 99, pages 149–158, 1999.

[15] T. Jaakkola, D. Haussler, et al. Exploiting generative models in discriminative classifiers. *Proceedings of NIPS*, 1999.

[16] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local image descriptors into compact codes. *IEEE TPAMI*, 2011.

[17] J. W. Kim, K. S. Candan, and J. Tatemura. Efficient overlap and content reuse detection in blogs and online news articles. In *Proceedings of WWW '09*, 2009.

[18] 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.

[19] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of ACL'11*, 2011.

[20] D. Metzler, Y. Bernstein, W. B. Croft, A. Moffat, and J. Zobel. Similarity measures for tracking information flow. In *Proceedings of CIKM '05*, 2005.

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*, 2013.

[22] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*. 2013.

[23] P. J. Moreno and R. Rifkin. Using the fisher kernel method for web audio classification. In *Proceedings of ICASSP'00*, 2000.

[24] M. Norouzi and D. Fleet. Minimal loss hashing for compact binary codes. In *Proceedings of ICML '11*.

[25] X. Qiu, Q. Zhang, and X. Huang. Fudannlp: A toolkit for chinese natural language processing. In *Proceedings of ACL'13*, 2013.

[26] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[27] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International Journal of Computer Vision*, pages 1–24, 2013.

[28] J. Seo and W. B. Croft. Local text reuse detection. In *Proceedings of SIGIR '08*, 2008.

[29] A. Si, H. V. Leong, and R. W. Lau. Check: a document plagiarism detection system. In *Proceedings of the 1997 ACM symposium on Applied computing*, 1997.

[30] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, 2011.

[31] Q. Sun, R. Li, D. Luo, and X. Wu. Text segmentation with lda-based fisher kernel. In *Proceedings of ACL'08*, 2008.

[32] M. Theobald, J. Siddharth, and A. Paepcke. Spotsigs: robust and efficient near duplicate detection in large web collections. In *Proceedings of SIGIR '08*, 2008.

[33] J. Turian, L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL'10*, 2010.

[34] E. Varol, F. Can, C. Aykanat, and O. Kaya. Codet: Sentence-based containment detection in news corpora. In *Proceedings of CIKM'11*, 2011.

[35] Q. Wang, L. Ruan, Z. Zhang, and L. Si. Learning compact hashing codes for efficient tag completion and prediction. In *Proceedings of CIKM '13*, 2013.

[36] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Processings of NIPS*, 2008.

[37] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proceeding of SIGIR '10*, 2010.

[38] Q. Zhang, Y. Wu, Z. Ding, and X. Huang. Learning hash codes for efficient content reuse detection. In *Proceedings of SIGIR'12*, 2012.

[39] Q. Zhang, Y. Zhang, H. Yu, and X. Huang. Efficient partial-duplicate detection based on sequence matching. In *Proceedings of SIGIR '10*, 2010.