Principles of Hash-based Text Retrieval

Benno Stein
Faculty of Media, Media Systems
Bauhaus University Weimar
99421 Weimar, Germany
benno.stein@medien.uni-weimar.de

ABSTRACT

Hash-based similarity search reduces a continuous similarity relation to the binary concept "similar or not similar": two feature vectors are considered as similar if they are mapped on the same hash key. From its runtime performance this principle is unequaled—while being unaffected by dimensionality concerns at the same time. Similarity hashing is applied with great success for near similarity search in large document collections, and it is considered as a key technology for near-duplicate detection and plagiarism analysis.

This papers reveals the design principles behind hash-based search methods and presents them in a unified way. We introduce new stress statistics that are suited to analyze the performance of hash-based search methods, and we explain the rationale of their effectiveness. Based on these insights, we show how optimum hash functions for similarity search can be derived. We also present new results of a comparative study between different hash-based search methods.

Categories and Subject Descriptors

H.3.1 [**INFORMATION STORAGE AND RETRIEVAL**]: Content Analysis and Indexing; H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval; F [**Theory of Computation**]: MISCELLANEOUS

General Terms

Theory, Performance

Keywords

hash-based similarity search, locality-sensitive hashing, dimension reduction

1. INTRODUCTION AND BACKGROUND

This paper contributes to an aspect of similarity search that receives increasing attention in information retrieval: The use of hashing to significantly speed up similarity search. The hash-based search paradigm has been applied with great success for the following tasks:

- *Near-Duplicate Detection.* Given a (very large) corpus D, find all documents whose pairwise similarity is close to 1 [29, 13].
- *Plagiarism Analysis.* Given a candidate document d and a (very large) corpus D, find all documents in D that contain nearly identical passages from d [27].

From the retrieval viewpoint hash-based text retrieval is an incomplete technology. Identical hash keys do not imply high similarity but indicate a high probability of high similarity. This fact suggests the solution strategy for the aforementioned tasks: In a first step a candidate set $D_q \subset D$, $|D_q| \ll |D|$, is constructed by a hash-based retrieval method; in a second step D_q is further investigated by a complete method.

The entire retrieval setting can be formalized as follows. Given are (*i*) a set $D = \{d_1, \ldots, d_n\}$ of documents each of which is described by an m-dimensional feature vector, $\mathbf{x} \in \mathbf{R}^m$, and (*ii*) a similarity measure, $\varphi : \mathbf{R}^m \times \mathbf{R}^m \rightarrow [0; 1]$, with 0 and 1 indicating no and maximum similarity respectively. φ may rely on the l_2 norm or on the angle between two feature vectors. For a query document d_q, represented by feature vector $\mathbf{x}_{d_q}$, and a similarity threshold $\theta \in [0; 1]$, we are interested in the documents of the θ-neighborhood $D_q \subseteq D$ of d_q, which is defined by the following condition:

$$d \in D_q \quad \Leftrightarrow \quad \varphi(\mathbf{x}_{d_q}, \mathbf{x}_d) > \theta,$$

where $\mathbf{x}_d$ denotes the feature vector of d. Within information retrieval applications the documents are represented as high-dimensional term vectors with $m > 10^4$, typically under the vector space model. We distinguish between the real documents, $d \in D$, and their representations as feature vectors, since one and the same document may be analyzed under different models, different representations, and different similarity measures, as will be the case in this paper.

In low-dimensional applications, say, $m < 10$, the retrieval problem can be efficiently solved with space-partitioning methods like grid-files, KD-trees, or quad-trees, as well as with data-partitioning index trees such as R-trees, Rf-trees, or X-trees. For significantly larger m the construction of D_q cannot be done better than by a linear scan in $O(|D|)$ [28]. However, if one accepts a decrease in recall, the search can be dramatically accelerated with similarity hashing. As will be discussed later on, the effectiveness of similarity hashing results from the fact that the recall is controlled in terms of the similarity threshold θ for a given similarity measure φ.

To motivate the underlying ideas consider an m-dimensional document representation under the vector space model with a tf-weighting scheme. An—admittedly—very simple similarity hash function h_φ, $h_\varphi : \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \to \mathbf{N}$, could map each term vector $\mathbf{x}$ on a single number $h_\varphi(\mathbf{x})$ that totals the number of those terms in $\mathbf{x}$ starting with the letter "a". If $h_\varphi(\mathbf{x}_{d_1}) = h_\varphi(\mathbf{x}_{d_2})$ it is assumed that d_1 and d_2 are similar.

Though this example is simple, it illustrates the principle and the problems of hash-based similarity search:

- If h_φ is too generic it will allegedly claim very dissimilar documents as similar, say, it will return a large number of false positives.
- If h_φ is too specific the understanding of similarity will become too narrow. Take MD5-hashing as an example, which can only be used to model the similarity threshold $\theta = 1$.

If h_φ is purposefully designed and captures the gist of the feature vectors, search queries can be answered in virtually constant time, independent of the dimension of $\mathbf{x}$.

1.1 Perfect Similarity Sensitive Hashing

First, we want to point out that hash-based similarity search is a space partitioning method. Second, it is interesting to note that, at least in theory, for a document set D and a similarity threshold θ a perfect space partitioning for hash-based search can be stated. To make this plausible we have formulated hash-based similarity search as a set covering problem. This generic view differs from the computation-centric descriptions found in the relevant literature.

Consider for this purpose the $\mathbf{R}^m$ being partitioned into overlapping regions such that the similarity of any two points of the same region is above θ, where each region is characterized by a unique key $\kappa \in \mathbf{N}$. Moreover, consider a multivalued hash function, $h^*_\varphi : \mathbf{R}^m \to \mathcal{P}(\mathbf{N})$, which is "perfectly similarity sensitive" with regard to threshold θ.[1] $\forall d_1, d_2 \in D$:

$$\underbrace{\left(h^*_\varphi(\mathbf{x}_{d_1}) \cap h^*_\varphi(\mathbf{x}_{d_2}) \right) \neq \emptyset}_{\alpha} \quad \Leftrightarrow \quad \underbrace{\varphi(\mathbf{x}_{d_1}, \mathbf{x}_{d_2}) > \theta}_{\beta} \qquad (1)$$

Rationale and Utilization. h^*_φ assigns each feature vector $\mathbf{x}_d$ a membership set $N_d \in \mathcal{P}(\mathbf{N})$ of region keys, whereas two sets, N_{d_1}, N_{d_2}, share a key iff $\mathbf{x}_{d_1}$ and $\mathbf{x}_{d_2}$ have a region in common. Figure 1, which is used later on in a different connection, serves as an illustration.

Based on h^*_φ we can organize the mapping between all region keys K, $K := \bigcup_{d \in D} N_d$, and documents with the same region key as a hash table h, $h : K \to \mathcal{P}(D)$. Based on h the θ-neighborhood D_q of d_q can be constructed in $O(|D_q|)$ runtime: [2]

$$D_q = \bigcup_{\kappa \in h^*_\varphi(\mathbf{x}_{d_q})} h(\kappa) \qquad (2)$$

Observe that h^*_φ operationalizes both perfect precision and perfect recall. For a set D that is completely known and time-invariant such a function may be found. However, in most cases the equivalence relation of Equation (1), $\alpha \Leftrightarrow \beta$, cannot be guaranteed:

$\not\Rightarrow$ If β is not a conclusion of α, D_q contains documents that do not belong to the θ-neighborhood of d_q: the precision is < 1.

$\not\Leftarrow$ If α is not a conclusion of β, D_q does not contain all documents from the θ-neighborhood of d_q: the recall is < 1.

[1] For the time being only the existence of such a partitioning along with a hash function is assumed, not its construction.

[2] In most practical applications $O(|D_q|)$ is bound by a small constant since $|D_q| \ll |D|$. The cost of a hash table access $h(\kappa)$ is assessed with $O(1)$; experience shows that for a given application hash functions with this property can always be stated [7].

Current research is concerned with the development of similarity hash functions that are robust in their behavior, efficient to be computed, and, most importantly, that provide an adjustable trade-off between precision and recall.

1.2 Contributions of the Paper

Our contributions relate to retrieval technology in general; they have been developed and analyzed with focus on text retrieval tasks under arbitrary classes of vector space models. In detail:

- The construction principles that form the basis of most hash-based search methods are revealed, exemplified, and related to the statistical concepts of precision and recall (Section 2).
- The relation between hash-based search methods and optimum embeddings is analyzed. New stress statistics are presented that give both qualitative and quantitative insights into the effectiveness of similarity hashing (Subsection 3.1 and 3.2).
- Based on a manipulation of the original similarity matrix it is shown how optimum methods for hash-based similarity search can be derived in closed retrieval situations (Subsection 3.3).
- New results of a comparative study between different hash-based search methods are presented (Section 4). This analysis supports the theoretical considerations and the usefulness of the new stress statistics developed in Section 3.

2. HASH-BASED SEARCH METHODS

Despite the use of sophisticated data structures nearest neighbor search in D degrades to a linear search if the dimension of the feature vectors is around 10 or higher. If one sacrifices exactness, that is to say, if one accepts values below 1 for precision and recall, the runtime bottleneck can be avoided by using hash-based search methods. These are specifically designed techniques to approximate near(est) neighbor search within sublinear runtime in the collection size $|D|$.

2.1 Related Work

Only few hash-based search methods have been developed so far, in particular random projection, locality-sensitive hashing, and fuzzy-fingerprinting [20, 18, 11, 26]; they are discussed in greater detail in Subsection 2.3 and 2.4.

As will be argued in Subsection 2.2, hash-based search methods operationalize—apparently or hidden—a means for embedding high-dimensional vectors into a low-dimensional space. The intended purpose is dimension reduction while retaining as much as possible of the similarity information. In information retrieval embedding technology has been developed for the discovery of hidden semantic structures: a high-dimensional term representation of a document is embedded into a low-dimensional concept space. Known transformation techniques include latent semantic indexing and its variants, probabilistic latent semantic analysis, iterative residual rescaling, or principal component analysis. The concept representation shall provide a better recall in terms of the semantic expansion of queries projected into the concept space.

Embedding is a vital step within the construction of a similarity hash function h_φ. Unfortunately, the mentioned semantic embedding technology cannot be applied in this connection, which is rooted in the nature of the use case. Hash-based search focuses on what is called here "open" retrieval situations, while semantic embedding implies a "closed" or "semi-closed" retrieval situation.

	Open retrieval situation	(Semi-)closed retrieval
Un-biased	Locality sensitive hashing [11]	MDS with cosine similarity (latent semantic indexing) [9]
	p-stable LSH [8]	Non-metric MDS [21]
	LSH forest [3]	PCA [19]
Biased	Fuzzy-fingerprinting [26]	Probabilistic LSI [16]
	Vector approximation [28]	Autoencoder NN [15]
	Shingling [4]	Iterative residual rescaling [1]
		Locality preserv. ind., LPI [12]
		Orthogonal LPI [5]

Table 1: Classification of embedding paradigms used for indexing in information retrieval.

This distinction pertains to the knowledge that is compiled into the retrieval function $\rho : \mathbf{Q} \times \mathbf{D} \rightarrow \mathbf{R}$, where $\mathbf{Q}$ and $\mathbf{D}$ designate the computer representations of the sets Q and D of queries and documents respectively.

- *Open Retrieval Situation.* Q and D are *un*known in advance. ρ relies on generic language concepts such as term distribution, term frequency, or sentence length. An example is the vector space model along with the cosine similarity measure.
- *Closed Retrieval Situation.* Q and D, and hence $\mathbf{Q}$ and $\mathbf{D}$ are known in advance. ρ models semantic dependencies found in $\mathbf{D}$ with respect to $\mathbf{Q}$. An example is an autoencoder neural network applied for category identification in D [15].
- *Semi-Closed Retrieval Situation.* Q is unknown and D is known in advance. ρ models semantic dependencies of $\mathbf{D}$ and expands a query $q \in Q$ with respect to the found structure. An example is PLSI.

We propose the scheme in Table 1 to classify embedding methods used in information retrieval. The scheme distinguishes also whether or not domain knowledge is exploited within the embedding procedure (unbiased versus biased). E. g., locality sensitive hashing works on arbitrary data, while fuzzy-fingerprinting as well as shingling exploit the fact that the embedded data is text. A similar argumentation applies to MDS and probabilistic LSI.

Aside from their restriction to (semi-)closed retrieval most of the embedding methods in the right column of Table 1 cannot be scaled up for large collections: they employ some form of spectral decomposition, which is computationally expensive.

2.2 Generic Construction Principle of h_φ

We developed a unified view on hash-based search methods by interpreting them as instances of a generic construction principle, which comprises following steps:

1. *Embedding.* The m-dimensional feature vectors of the documents in D are embedded in a low-dimensional space, striving for minimum distortion. The resulting k-dimensional feature vectors shall resemble the distance ratios, at least the order of the pairwise inter-document distances, as close as possible.[3]
2. *Quantization.* The real-valued components of the embedded feature vectors are mapped onto a small number of values.
3. *Encoding.* From the k quantized components a single number is computed, which serves as hash code.

[3]Against the analysis presented in Section 3, the concept of optimality implied here must be seen more differentiated.

Some or all of these steps may be repeated for one and the same original feature vector $\mathbf{x}$ in order to obtain a *set* of hash codes for $\mathbf{x}$. The next subsection exemplifies this construction principle for two hash-based search methods: locality-sensitive hashing and fuzzy-fingerprinting. Subsection 2.4 explains the properties of hash-based search methods in terms of the precision and recall semantics.

2.3 A Unified View to Locality-Sensitive Hashing and Fuzzy-Fingerprinting

Locality-sensitive hashing (LSH) is a generic framework for the construction of hash-based search methods. To realize the embedding, a locality-sensitive hash function h_φ employs a family $\mathcal{H}_\varphi$ of simple hash functions h, $h : \mathbf{R}^m \rightarrow \mathbf{N}$. From $\mathcal{H}_\varphi$ a set of k functions is chosen by an independent and uniformly distributed random choice, where each function is used to compute one component of the embedding $\mathbf{y}$ of an original vector $\mathbf{x}$. Several hash families $\mathcal{H}_\varphi$ that are applicable for text-based information retrieval have been proposed [6, 8, 3]. Our focus is on the approach of Datar et. al. [8], which maps a feature vector $\mathbf{x}$ to a real number by computing the dot product $\mathbf{a}^T \cdot \mathbf{x}$. $\mathbf{a}$ is a random vector whose components are chosen from an α-stable probability distribution.[4]

Quantization is achieved by dividing the real number line into equidistant intervals of width r each of which having assigned a unique natural number. The result of the dot product is identified with the number of its enclosing interval.

Encoding can happen in different ways and is typically done by summation; the computation of $h_\varphi^{(\rho)}$ for a set ρ of random vectors $\mathbf{a}_1, \ldots, \mathbf{a}_k$ reads as follows:

$$h_\varphi^{(\rho)}(\mathbf{x}) = \sum_{i=1}^{k} \left\lfloor \frac{\mathbf{a}_i^T \cdot \mathbf{x} + c}{r} \right\rfloor,$$

where $c \in [0, r]$ is a randomly chosen offset of the real number line. A multivalued hash function repeats the outlined steps for different sets $\rho_1, \ldots, \rho_l$ of random vectors.

Fuzzy-fingerprinting (FF) is a hash-based search method specifically designed for text-based information retrieval. Its underlying embedding procedure can be understood as an abstraction of the vector space model and happens by "condensing" an m-dimensional term vector $\mathbf{x}$ toward k prefix classes. A prefix class comprises all terms with the same prefix; the components of the embedded feature vector $\mathbf{y}$ quantify the normalized expected deviations of the k chosen prefix classes.[5]

Quantization is achieved by applying a fuzzification scheme, ρ, which projects the exact deviations $y_1, \ldots, y_k$ on r deviation intervals: $\rho : \mathbf{R} \rightarrow \{0, \ldots, r-1\}$

Encoding is done by computing the smallest number in radix r notation from the fuzzified deviations; the computation of $h_\varphi^{(\rho)}$ for a particular fuzzification scheme ρ reads as follows:

$$h_\varphi^{(\rho)}(\mathbf{x}) = \sum_{i=1}^{k} \rho(y_i) \cdot r^{i-1},$$

where y_i is the normalized expected deviation of the i-th prefix class in the original term vector $\mathbf{x}$. Similar to LSH, a multivalued hash function repeats the quantization and encoding steps for different fuzzification schemes, $\rho_1, \ldots, \rho_l$.

[4]α-stability guarantees locality sensitivity [17, 23]. An example for an α-stable distribution is the Gaussian distribution.

[5]For the normalization the British National Corpus is used as reference. The BNC is a 100 million word collection of written and spoken language from a wide range of sources, designed to represent a wide cross-section of current British English [2].

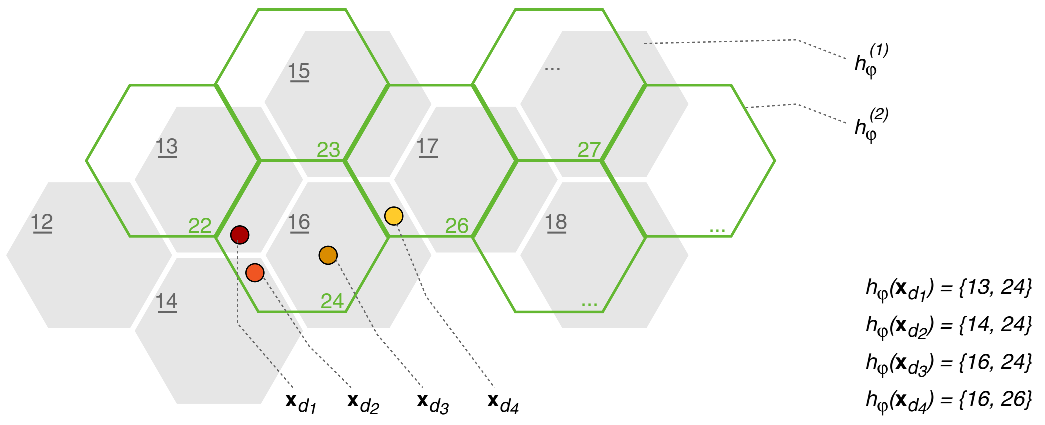
15
...
13
23
17
27
12
22
16
26
18
...
14
24
...
$h_\varphi^{(1)}$
$h_\varphi^{(2)}$
x_{d1}
x_{d2}
x_{d3}
x_{d4}
$h_\varphi(x_{d1}) = \{13, 24\}$
$h_\varphi(x_{d2}) = \{14, 24\}$
$h_\varphi(x_{d3}) = \{16, 24\}$
$h_\varphi(x_{d4}) = \{16, 26\}$

Figure 1: A space partitioned into overlapping regions, hinted as two grids of shaded and outlined hexagons. Each region is characterized by a unique key; points in the same region have a similarity of at least θ. A similarity hash function h_φ at level θ assigns a set of region keys to a feature vector $\mathbf{x}_d$, implying the following semantics: If and only if two feature vectors share a region key they are considered having a similarity of at least θ. In the example $h_\varphi(\mathbf{x}) = \{h_\varphi^{(1)}(\mathbf{x}), h_\varphi^{(2)}(\mathbf{x})\}$ operationalizes both a precision and a recall of 1. For readability purposes the keys of the shaded regions are shown underlined.

2.4 Controlling Retrieval Properties

The most salient property of hash-based search is the simplification of a continuous similarity function φ to the binary concept "similar or not similar": two feature vectors are considered as similar if their hash keys are equal; otherwise they are considered as not similar. This implication is generalized in Equation (1) at the outset; the generalization pertains to two aspects: (*i*) the equivalence relation refers to a similarity threshold θ, and (*ii*) the hash function h_φ is multivalued.

With the background of the presented hash-based search methods we now continue the discussion of precision and recall from Subsection 1.1. Observe that the probability of a hash collision for two vectors $\mathbf{x}_{d_1}, \mathbf{x}_{d_2}$ decreases if the number k of simple hash functions (LSH) or prefix classes (FF) is increased. Each hash function or each prefix class captures additional knowledge of $\mathbf{x}$ and hence raises the similarity threshold θ. This can be broken down to the following formula, termed Property 1:

"Code length controls precision."

Being multivalued is a necessary condition for h_φ to achieve a recall of 1. A scalar-valued hash function computes one key for one feature vector $\mathbf{x}$ at a time, and hence it defines a rigorous partitioning of the feature vector space. Figure 1 illustrates this connection: The scalar-valued hash function $h_\varphi^{(1)}$ responsible for the shaded partitioning assigns different keys to the vectors $\mathbf{x}_{d_1}$ and $\mathbf{x}_{d_2}$, despite their high similarity (low distance). With the multivalued hash function, $h_\varphi = \{h_\varphi^{(1)}, h_\varphi^{(2)}\}$, which also considers the outlined partitioning, the intersection $h_\varphi(\mathbf{x}_{d_1}) \cap h_\varphi(\mathbf{x}_{d_2})$ is not empty, giving raise to infer that $\varphi(\mathbf{x}_{d_1}, \mathbf{x}_{d_2}) > \theta$. In fact, there is a monotonic relationship between the number of hash codes and the achieved recall, which can be broken down to the following formula, termed Property 2:

"Code multiplicity controls recall".

However, there is no free lunch, the improved recall is bought with a decrease in precision.

3. OPTIMALITY AND EMBEDDING

The embedding of the vector space model into a low-dimensional space is inevitably bound up with information loss. The smaller the embedding error is, the better are precision and recall of the constructed hash function, because the affine transformation in Step 2 and 3 (cf. Subsection 2.2), which maps an embedded vector onto a hash code, is distance-preserving.

The section starts with a derivation of the globally optimum embedding under the cosine similarity measure, and then uncovers the inferiority of this embedding compared to the prefix class embedding of fuzzy-fingerprinting (Subsection 3.2). This observation is explained by the idea of *threshold-centered* embeddings, for which we introduce the formal underpinning in the form of new error statistics, called *precision stress* and *recall stress* at a given similarity threshold θ. By extending the idea toward thresholded similarity matrices we show how optimum embeddings for similarity hashing in closed retrieval situations can be developed (Subsection 3.3).

3.1 Globally Optimum Embeddings

Multidimensional scaling (MDS) designates a class of techniques for embedding a set of objects into a low-dimensional real-valued space, called embedding space here. The embedding error, also called "stress", is computed from the deviations between the original inter-object similarities and the new inter-object similarities in the embedding space.

Given n objects, the related similarity matrix, $\mathbf{S}$, is a symmetric $n \times n$ matrix of positive real numbers, whose (i, j)-th entry quantifies the similarity between object i and object j. Let each object be described by an m-dimensional feature vector $\mathbf{x} \in \mathbf{R}^m$, and let $\mathbf{X}$ be the $m \times n$ matrix comprised of these vectors. [6]

Without loss of generality we assume each feature vector $\mathbf{x}$ being normalized according to the l_2-norm, i. e., $||\mathbf{x}||_2 = 1$. Then, under the cosine similarity measure, $\mathbf{S}$ is defined by the identity $\mathbf{S} = \mathbf{X}^T\mathbf{X}$, where $\mathbf{X}^T$ designates the matrix transpose of $\mathbf{X}$.

An important property of the cosine similarity measure is that under the Frobenius norm an optimum embedding of $\mathbf{X}$ can be directly constructed from its singular value decomposition (SVD). With SVD an arbitrary matrix $\mathbf{X}$ can be uniquely represented as the product of three matrices:[7]

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

$\mathbf{U}$ is a column orthonormal $m \times r$ matrix, $\mathbf{\Sigma}$ is an $r \times r$ diagonal matrix with the singular values of $\mathbf{X}$, and $\mathbf{V}$ is an $n \times r$ matrix. I. e.,

[6]In IR applications $\mathbf{X}$ is the term-document-matrix. For applying an MDS only $\mathbf{S}$ must be given.

[7]Unique up to rearrangement of columns and subspace rotations.

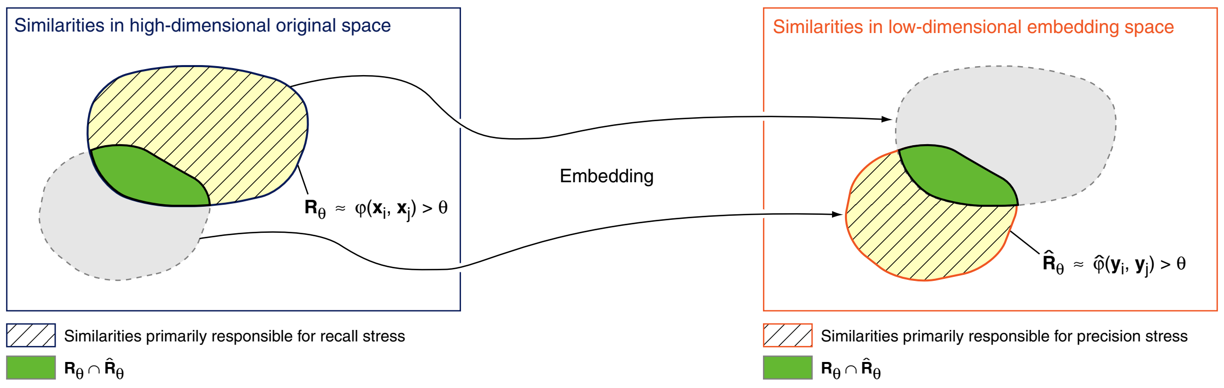
Similarities in high-dimensional original space
Similarities in low-dimensional embedding space
Embedding
$\mathbf{R}_\theta \approx \varphi(\mathbf{x}_i, \mathbf{x}_j) > \theta$
$\hat{\mathbf{R}}_\theta \approx \hat{\varphi}(\mathbf{y}_i, \mathbf{y}_j) > \theta$
▨ Similarities primarily responsible for recall stress
🟩 $\mathbf{R}_\theta \cap \hat{\mathbf{R}}_\theta$
▨ Similarities primarily responsible for precision stress
🟩 $\mathbf{R}_\theta \cap \hat{\mathbf{R}}_\theta$

Figure 2: If the original document representations, X, are embedded into a low-dimensional space, the resulting document representations Y resemble the original similarities only imperfectly. Given a particular threshold θ, similarities of the original space may be shifted from above θ to below θ (hatched area left), from below θ to above θ (hatched area right), or still remain in the interval $[\theta; 1]$ (green area). The similarities in the hatched areas are responsible for the major part of the embedding stress.

$\mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{V}\mathbf{V}^T = \mathbf{I}$ where $\mathbf{I}$ designates the identity matrix. Using these properties the matrix $\mathbf{S}$ can be rewritten under both the viewpoint of its singular value decomposition and the viewpoint of similarity computation:

$$\begin{aligned} \mathbf{S} &= \mathbf{X}^T\mathbf{X} = (\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T)^T\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T \\ &= \underbrace{\mathbf{V}\boldsymbol{\Sigma}^2\mathbf{V}^T}_{\text{SVD}} = \underbrace{(\boldsymbol{\Sigma}\mathbf{V}^T)^T(\boldsymbol{\Sigma}\mathbf{V}^T)}_{\text{Similarity computation}} \end{aligned}$$

$\boldsymbol{\Sigma}\mathbf{V}^T$ represents a set of points with the same inter-object similarities as the original vectors $\mathbf{X}$. The nature of the cosine similarity measure implies the direct construction of $\mathbf{S}$ and, in particular, the identities $rank(\mathbf{S}) = rank(\mathbf{X}) = rank(\boldsymbol{\Sigma}\mathbf{V}^T)$. Conversely, if we restrict the dimensionality of the embedding space to k, the resulting similarity matrix $\hat{\mathbf{S}}$ is also of rank k. According to the Eckart-Young Theorem the optimum rank-k approximation $\hat{\mathbf{S}}^*$ of $\mathbf{S}$ under the Frobenius norm can be obtained from the SVD of $\mathbf{S}$, by restricting the matrix product to the k largest singular values [10]:

$$\begin{aligned} \hat{\mathbf{S}}^* &= \mathbf{V}_k\boldsymbol{\Sigma}_k^2\mathbf{V}_k^T = (\boldsymbol{\Sigma}_k\mathbf{V}_k^T)^T(\boldsymbol{\Sigma}_k\mathbf{V}_k^T) \\ \Rightarrow \quad \boldsymbol{\Sigma}_k\mathbf{V}_k^T &= \underset{\{\mathbf{Y} \mid \substack{columns(\mathbf{Y})=n, \\ rank(\mathbf{Y})=k}\}}{\text{argmin}} ||\mathbf{S} - \mathbf{Y}^T\mathbf{Y}||_F \end{aligned}$$

In the information retrieval community the embedding $\mathbf{Y}_{SVD} := \boldsymbol{\Sigma}_k\mathbf{V}_k^T$ of document vectors $\mathbf{X}$ is known as representation in the so-called latent semantic space, spanned by k concepts. The embedding process became popular under the name of latent semantic indexing (LSI) [9].

Remark 1. A common misconception is that LSI projects the document vectors into a subspace in order to represent semantic similarity. Rather, LSI constructs new features to approximate the original document representations. And, if the dimension of the embedding space is properly chosen then, due to the reduction of noise and the elimination of weak dependencies, this embedding is able to address retrieval problems deriving from the use of synonymous words. As a consequence the retrieval performance may be improved in semi-closed retrieval applications. Hofmann argues similarly [16]: the superposition principle underlying LSI is unable to handle polysemy.

3.2 The Rationale of Hash-Based Search: Threshold-Centered Embeddings

Though the embedding $\mathbf{Y}_{SVD}$ minimizes the embedding error of $\mathbf{X}$, it is not the best starting point for constructing similarity-sensitive hash codes. The main reason is that an MDS strives for a global stress minimization, while hash-based search methods concentrate on the high similarities in $\mathbf{S}$ in first place.[8] The nature of this property is captured by the following definition, which relates the threshold-specific stress of an embedding to the statistical concepts of precision and recall. Figure 2 illustrates the definition.

Definition 1 (precision stress, recall stress) *Let D be a set of objects and let $\mathbf{X}$ and $\mathbf{Y}$ be their representations in the n-dimensional and the k-dimensional space respectively, $k < n$. Moreover, let $\varphi : \mathbf{X} \times \mathbf{X} \to [0; 1]$ and $\hat{\varphi} : \mathbf{Y} \times \mathbf{Y} \to [0; 1]$ be two similarity measures, and let $\theta \in [0; 1]$ be a similarity threshold.*

θ defines two result sets, $\mathbf{R}_\theta$ and $\hat{\mathbf{R}}_\theta$, which are comprised of those pairs $\{x_i, x_j\}$, $x_i, x_j \in D$, whose respective representations in $\mathbf{X}$ and $\mathbf{Y}$ are above the similarity threshold θ:

$$\{x_i, x_j\} \in \mathbf{R}_\theta \;\Leftrightarrow\; \varphi(\mathbf{x}_i, \mathbf{x}_j) > \theta,$$

$$\text{and likewise: } \{x_i, x_j\} \in \hat{\mathbf{R}}_\theta \;\Leftrightarrow\; \hat{\varphi}(\mathbf{y}_i, \mathbf{y}_j) > \theta$$

Then the set of returned pairs from the embedding space, $\hat{\mathbf{R}}_\theta$, defines the precision stress at similarity threshold θ, e_{p_θ}:

$$e_{p_\theta} = \frac{\sum\limits_{\{x_i,x_j\} \in \hat{\mathbf{R}}_\theta} \left(\varphi(\mathbf{x}_i, \mathbf{x}_j) - \hat{\varphi}(\mathbf{y}_i, \mathbf{y}_j)\right)^2}{\sum\limits_{\{x_i,x_j\} \in \hat{\mathbf{R}}_\theta} \left(\varphi(\mathbf{x}_i, \mathbf{x}_j)\right)^2}$$

Likewise, the set of similar pairs in the original space, $\mathbf{R}_\theta$, defines the recall stress at similarity threshold θ, e_{r_θ}:

$$e_{r_\theta} = \frac{\sum\limits_{\{x_i,x_j\} \in \mathbf{R}_\theta} \left(\varphi(\mathbf{x}_i, \mathbf{x}_j) - \hat{\varphi}(\mathbf{y}_i, \mathbf{y}_j)\right)^2}{\sum\limits_{\{x_i,x_j\} \in \mathbf{R}_\theta} \left(\varphi(\mathbf{x}_i, \mathbf{x}_j)\right)^2}$$

Remark 2. The precision stress and the recall stress of an embedding $\mathbf{Y}$ are statistics that tell us something about the maximum precision and recall that can be achieved with similarity hash codes constructed from $\mathbf{Y}$. The larger the precision stress is the higher is the probability that two embedded vectors, $\mathbf{y}_i, \mathbf{y}_j$, are claimed being similar though their similarity in the original space,

[8] *"The similarity threshold controls the effective embedding error."* This property complements the two properties of hash-based search methods stated in Subsection 2.4.

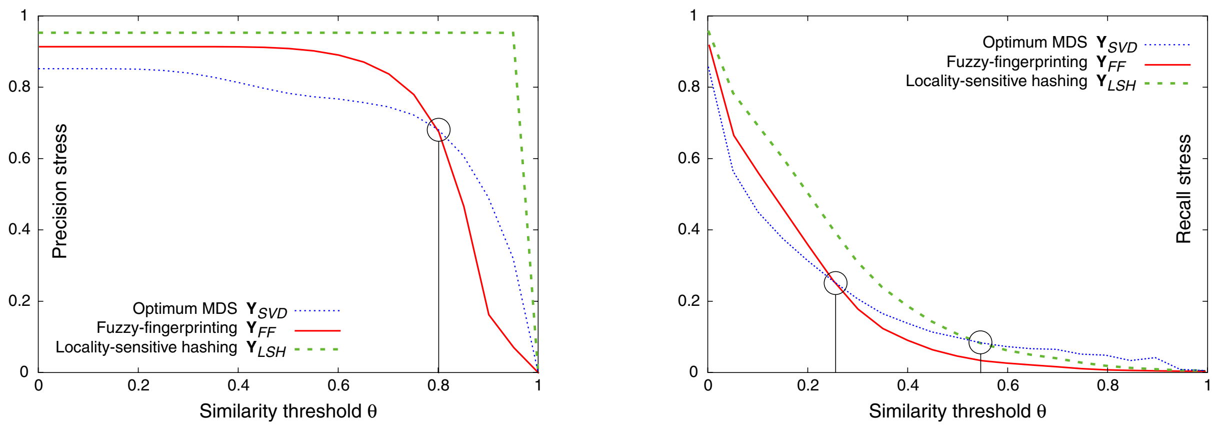

Figure 3: Evolution of the embedding stress against the similarity threshold θ (lower stress is better). The left plot takes the embedded vectors as basis, the right plot the original vectors, corresponding to the precision stress, e_p, and the recall stress, e_r, respectively. At some threshold the embedding of fuzzy-fingerprinting, $\mathbf{Y}_{FF}$, outperforms the optimum MDS embedding, $\mathbf{Y}_{SVD}$.

$s_{ij} = \varphi(\mathbf{x}_i, \mathbf{x}_j)$, is low. Likewise, the larger the recall stress is the higher is the probability that two vectors in the original space, $\mathbf{x}_i, \mathbf{x}_j$, are mapped onto different codes though their similarity, s_{ij}, is high.

For the three embeddings, $\mathbf{Y}_{SVD}$, $\mathbf{Y}_{FF}$, and $\mathbf{Y}_{LSH}$, obtained from optimum MDS, fuzzy-fingerprinting, and LSH respectively, we have analyzed the precision stress and the recall stress at various similarity thresholds and with different corpora. The results reflect the predicted behavior:

1. Because of its generality (domain independence) the LSH embedding is consistently worse than the prefix class embedding of fuzzy-fingerprinting.
2. At some break-even point the retrieval performance of prefix class embedding outperforms the optimum MDS embedding.

Figure 3 illustrates this behavior for a sample of 2000 documents drawn from the Reuters Corpus Volume 1 (RCV1) [24]. With other corpora and other parameter settings for the hash-based search methods this characteristic is observed as well. We analyzed in this connection also specifically compiled corpora whose similarity distribution is significantly skewed towards high similarities: Figure 4 contrasts the similarity distribution in the original Reuters Corpus (hatched light) and in the special corpora (solid dark).

Remark 3. For most retrieval tasks an—even high—precision stress can be accepted, since the necessary subsequent exact similarity analysis needs to be performed only for a very small fraction $|D_q|/|D|$ of all documents. Remember that the construction methods for the hash-based search methods provide sufficient means to fine-tune the trade-off between the precision stress, e_p, and the recall stress, e_r.

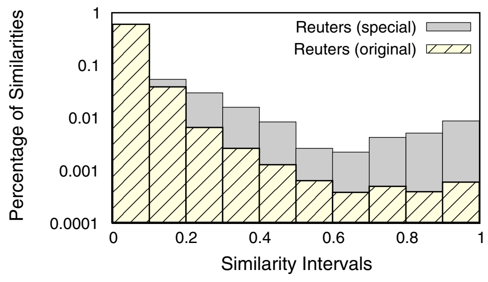

Figure 4: Similarity distribution in the original Reuters Corpus and in the special compilations with increased high similarities.

3.3 Threshold-Optimum Embeddings in Closed Retrieval Situations

Threshold-centered embeddings are tailored document models for special retrieval tasks such as near duplicate detection or high similarity search. They tolerate a large embedding error in the low similarity interval $[0, \theta]$ and strive for a high fidelity of similarities from the interval $[\theta, 1]$. This principle forms the rationale of hash-based search.

With $\mathbf{Y}_{SVD}$, obtained by optimally solving an MDS, an embedding that minimizes the accumulated error over all similarities is at hand. We now introduce a threshold-optimum embedding, $\mathbf{Y}^*$, which minimizes the accumulated error with respect to the interval $[\theta, 1]$. The presented ideas address the closed retrieval situation in first place—for open retrieval situations the construction of an optimum embedding requires a-priori knowledge about the term distribution in the collection D.[9] Though the typical use case for hash-based search are open retrieval situations, the derivation is useful because (*i*) it provides additional theoretical insights and (*ii*) it forms a basis to reason about performance bounds.

The θ-specific retrieval analysis of the preceding subsection suggests the construction principle of $\mathbf{Y}^*$. Instead of approximating the original similarity matrix $\mathbf{S}$ a “thresholded” similarity matrix $\mathbf{S}_\theta$ is taken as basis, introducing this way the binary nature of similarity hashing into the approximation process. For a given threshold θ the matrix $\mathbf{S}_\theta$ is defined as follows:

$$\mathbf{S}_\theta := \begin{pmatrix} f_\theta(s_{11}) & f_\theta(s_{12}) & \dots & f_\theta(s_{1n}) \\ \vdots & \vdots & \ddots & \vdots \\ f_\theta(s_{n1}) & f_\theta(s_{n2}) & \dots & f_\theta(s_{nn}) \end{pmatrix},$$

where $f_\theta(s)$ is a combination of two sigmoidal functions that define an upper threshold θ and a lower threshold ϑ respectively. Similarity values from $[\theta; 1]$ are amplified toward 1, similarity values from $[0; \theta)$ are moved toward ϑ. The following rationale reveals the underlying trade-off: with increasing difference $\theta - \vartheta$ the amplification above θ improves the robustness in the encoding step (cf. Subsection 2.2), with increasing ϑ the contraction toward ϑ reduces the error in the embedding step and hence allows for shorter codes. f_θ can be realized in different ways; within our analyses two consecutive $\tanh$-approximations with the thresholds $\vartheta = 0.1$ and $\theta = 0.8$ were employed.

[9]Remember that $\mathbf{Y}_{FF}$ is a domain-specific embedding which exploits knowledge about document models and term distributions.

		Precision			
Embedding	Dim.	$(0.8; 0.9]$	$(0.85; 1.0]$	$(0.9; 1.0]$	$(0.95; 1.0]$
$\mathbf{Y}^*$	50	0.58	0.71	0.84	0.95
$\mathbf{Y}_{FF}$	50	0.17	0.45	0.69	0.85
$\mathbf{Y}_{SVD}$	50	0.35	0.45	0.57	0.73
$\mathbf{Y}^*$	25	0.29	0.38	0.51	0.74
$\mathbf{Y}_{FF}$	25	0.01	0.02	0.09	0.59
$\mathbf{Y}_{SVD}$	25	0.16	0.22	0.34	0.56

Table 2: Results of a near-duplicate retrieval analysis, based on RCV1 and the experimental setup like before. The precision achieved with $\mathbf{Y}^*$ outperforms even the $\mathbf{Y}_{FF}$ embedding.

Since $\mathbf{S}_\theta$ is a symmetric matrix it is normal, and hence its Schur decomposition yields a spectral decomposition:

$$\mathbf{S}_\theta = \mathbf{Z}\mathbf{\Lambda}\mathbf{Z}^T$$

$\mathbf{Z}$ is an orthogonal matrix comprising the eigenvectors of $\mathbf{S}_\theta$, and $\mathbf{\Lambda}$ is a diagonal matrix with the eigenvalues of $\mathbf{S}_\theta$. If $\mathbf{S}_\theta$ is positive definite its unique Cholesky decomposition exists:

$$\mathbf{S}_\theta = \mathbf{Z}\mathbf{Z}^T$$

$\overline{\mathbf{X}} := \mathbf{Z}^T$ can directly be interpreted as matrix of thresholded document representations. As was shown in Subsection 3.1, the dimension of the embedding space, k, prescribes the rank of the approximation $\hat{\mathbf{S}}_\theta$ of $\mathbf{S}_\theta$. Its optimum rank-k-approximation, $\hat{\mathbf{S}}^*_\theta$, is obtained by an SVD of $\mathbf{S}_\theta$, which can be expressed in the factors of the rank-k-approximated SVD of $\overline{\mathbf{X}}$. Let $\mathbf{O}\mathbf{\Delta}\mathbf{Q}^T$ be the SVD of $\overline{\mathbf{X}}$ and hence $\mathbf{O}_k^T\mathbf{O}_k = \mathbf{I}$. Then holds:

$$\begin{aligned}\hat{\mathbf{S}}^*_\theta &= \overline{\mathbf{X}}_k^T\,\overline{\mathbf{X}}_k = (\mathbf{O}_k\mathbf{\Delta}_k\mathbf{Q}_k^T)^T\,\mathbf{O}_k\mathbf{\Delta}_k\mathbf{Q}_k^T\\ &= (\mathbf{\Delta}_k\mathbf{Q}_k^T)^T\,\mathbf{\Delta}_k\mathbf{Q}_k^T = \mathbf{Y}^{*T}\,\mathbf{Y}^*\end{aligned}$$

Remark 4. $\mathbf{Y}^* := \mathbf{\Delta}_k\mathbf{Q}_k^T$ is an embedding of $\mathbf{X}$ optimized for similarity hashing. Due to construction, $\mathbf{Y}^*$ is primarily suited to answer binary similarity questions at the a-priori chosen threshold θ. Since $\mathbf{S}_\theta$ is derived from $\mathbf{S}$ by sigmoidal thresholding, the document representations in $\mathbf{Y}$ are insusceptible with respect to a rank-k-approximation. This renders $\mathbf{Y}^*$ robust for similarity comparisons under the following interpretation of similarity:

$$\begin{aligned}&\text{If}\quad \langle \mathbf{y}^*_i, \mathbf{y}^*_j\rangle > 0.5 \quad \text{assume}\quad \langle \mathbf{x}_i, \mathbf{x}_j\rangle > \theta\\ &\text{If}\quad \langle \mathbf{y}^*_i, \mathbf{y}^*_j\rangle \le 0.5 \quad \text{assume}\quad \langle \mathbf{x}_i, \mathbf{x}_j\rangle \le \theta\end{aligned}$$

where $\langle\,,\,\rangle$ denotes the scalar product. Table 2 illustrates the superiority of $\mathbf{Y}^*$: For the interesting similarity interval $[\theta, 1]$ it outperforms the classical embedding as well as the embedding strategies of sophisticated hash-based search methods.

Remark 5. To obtain for a new n-dimensional vector $\mathbf{x}$ its optimum k-dimensional representation $\mathbf{y}^*$ at similarity threshold θ, a $k \times n$ projection matrix $\mathbf{P}$ can be stated:

$$\mathbf{y}^* = \mathbf{P}\mathbf{x}, \quad \text{where } \mathbf{P} \text{ is computed from } \quad \mathbf{X}^T\mathbf{P}^T = \mathbf{Y}^{*T}$$

Remark 6. The transformations imply the thresholded similarity matrix $\mathbf{S}_\theta$ being positive definite. An efficient and robust test for positive definiteness was recently proposed by Rumb [25]. If $\mathbf{S}_\theta$ is not positive definite it can be approximated by a positive definite matrix $\mathbf{S}_{\theta+}$, which should be the nearest symmetric positive definite matrix under the Frobenius norm. As shown by Higham, $\mathbf{S}_{\theta+}$ is given by following identity [14]:

$$\mathbf{S}_{\theta+} = \frac{\mathbf{G}+\mathbf{H}}{2} \quad \text{with} \quad \mathbf{G} = \frac{\mathbf{S}_\theta + \mathbf{S}_\theta^T}{2},$$

where $\mathbf{H}$ is the symmetric polar factor of $\mathbf{G}$.

4. HASH-BASED RETRIEVAL AT WORK

Finally, this section demonstrates the efficiency of locality-sensitive hashing, fuzzy-fingerprinting, and hash-based search in general. We report results from a large-scale experiment on near-duplicate detection and plagiarism analysis, using a collection of $100,000$ documents compiled with Yahoo, Google, and AltaVista by performing a focused search on specific topics. To compile the collection a small number of seed documents about a topic was chosen from which 100 keywords were extracted with a co-occurrence analysis [22]. Afterward, search engine queries were generated by choosing up to five keywords, and the highest ranked search results were downloaded and their text content extracted.

To render retrieval results comparable the two hash functions were parameterized in such a way that, on average, small and equally-sized document sets were returned for a query. As described in Section 2.4, this relates to adjusting the recall of the hash functions, which is done with the number of fuzzification schemes and random vector sets respectively: two or three different fuzzification schemes were employed for fuzzy-fingerprinting; between 10 and 20 different random vector sets were employed for locality-sensitive hashing. The precision of fuzzy-fingerprinting is controlled by the number k of prefix classes and the number r of deviation intervals per fuzzification scheme. To improve the precision performance either of them or both can be raised. Note that k is application-dependent; typical values for r range from 2 to 4. The precision of locality-sensitive hashing is controlled by the number k of combined hash functions. For instance, when using the hash family proposed by Datar et al., k corresponds to the number of random vectors per hash function [8]; typical values for k range from 20 to 100.

The plots in Figure 5 contrasts performance results. With respect to recall either approach is excellent at high similarity thresholds (> 0.8) compared to a linear search using a cosine measure. However, high recall values at low similarity thresholds are achieved by chance only. With respect to precision fuzzy-fingerprinting is significantly better than locality-sensitive hashing—a fact which directly affects the runtime performance. With respect to runtime performance both hashing approaches perform orders of magnitude faster than a linear search. For reasonably high thresholds θ the similarity distribution (Figure 4) along with the precision stress (Figure 3, left) determine a sublinear increase of the result set size $|D_q|$ for a document query d_q (Equation 2).

Remark 7. The computation of the baseline relies on a non-reduced vector space, defined by the dictionary underlying D. Note that a pruned document representation or a cluster-based preprocessing of D, for example, may have exhibited a slower—but yet linear growth. Moreover, the use of such specialized retrieval models makes the analysis results difficult to interpreted.

5. CONCLUSION AND CURRENT WORK

The paper analyzed the retrieval performance and explained the retrieval rationale of hash-based search methods. The starting point was the development of a unified view on these methods, along with the formulation of three properties that capture their design principles. We pointed out the selective nature of hash-based search and introduced new stress statistics to quantify this characteristic.

The concept of tolerating a large embedding error for small similarities while striving for a high fidelity at high similarities can be used to reformulate the original similarity matrix and thus to derive tailored embeddings in closed retrieval situations.

The presented ideas open new possibilities to derive theoretical bounds for the performance of hash-based search methods.

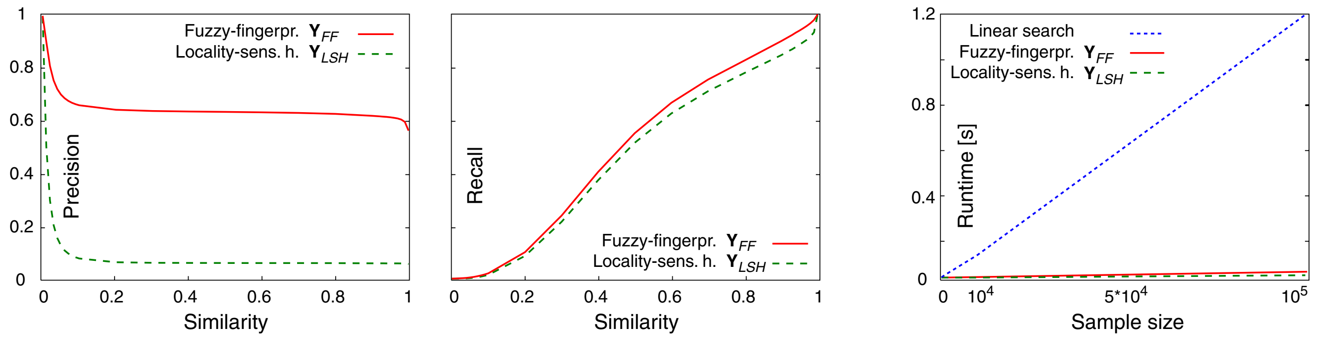

Figure 5: Near-duplicate detection and plagiarism analysis with hash-based search technology. The plots shows recall-at-similarity, precision-at-similarity, and runtime-at-sample-sizes, using fuzzy-fingerprinting (FF) and locality-sensitive hashing (LSH).

Whether they can be used to develop better search methods is subject of our research: by construction, $\mathbf{Y}^*$ outperforms other embeddings. It is unclear to which extent this property can be utilized in similarity search methods designed for *open* retrieval situations. The theoretical analysis of the trade-off between θ and ϑ as well as the Remarks 5 and 6 provide interesting links to follow.

6. REFERENCES

[1] R. Ando and L. Lee. Iterative Residual Rescaling: An Analysis and Generalization of LSI. In *Proc. 24th conference on research and development in IR*, 2001.

[2] G. Aston and L. Burnard. The BNC Handbook. `http://www.natcorp.ox.ac.uk/what/`, 1998.

[3] M. Bawa, T. Condie, and P. Ganesan. LSH Forest: Self-Tuning Indexes for Similarity Search. In *WWW'05: Proc. of the 14th int. conference on World Wide Web*, 2005.

[4] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic Clustering of the Web. In *Selected papers from the sixth int. conference on World Wide Web*, 1997.

[5] D. Cai and X. Hee. Orthogonal Locality Preserving Indexing. In *Proc. of the 28th conference on Research and development in IR*, 2005.

[6] M. S. Charikar. Similarity Estimation Techniques from Rounding Algorithms. In *STOC'02: Proc. of the thirty-fourth ACM symposium on theory of computing*, 2002.

[7] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge. 1990.

[8] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In *SCG'04: Proc. of the twentieth symposium on computational geometry*, 2004.

[9] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman. Indexing by Latent Semantic Analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[10] C. Eckart and G. Young. The Approximation of one Matrix by Another of Lower Rank. *Psychometrika*, 1:211–218, 1936.

[11] A. Gionis, P. Indyk, and R. Motwani. Similarity Search in High Dimensions via Hashing. In *The VLDB Journal*, 1999.

[12] X. He, D. Cai, H. Liu, and W.-Y. Ma. Locality Preserving Indexing for Document Representation. In *Proc. of the 27th conference on research and development in IR*, 2001.

[13] M. Henzinger. Finding Near-Duplicate Web Pages: a Large-Scale Evaluation of Algorithms. In *Proc. of the 29th conference on research and development in IR*, 2006.

[14] N. Higham. Computing a Nearest Symmetric Positive Semidefinite Matrix. *Linear Algebra and its App.*, 1988.

[15] G. Hinton and R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313:504–507, 2006.

[16] T. Hofmann. Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, 42:177–196, 2001.

[17] P. Indyk. Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation. In *FOCS'00: Proc. of the 41st symposium on foundations of computer science*, 2000. IEEE Computer Society.

[18] P. Indyk and R. Motwani. Approximate Nearest Neighbor – Towards Removing the Curse of Dimensionality. In *Proc. of the 30th symposium on theory of computing*, 1998.

[19] I. Jolliffe. *Principal Component Analysis*. Springer, 1996.

[20] J. Kleinberg. Two Algorithms for Nearest-Neighbor Search in High Dimensions. In *STOC'97: Proc. of the twenty-ninth ACM symposium on theory of computing*, 1997.

[21] J. Kruskal. Multidimensional Scaling by Optimizing Goodness of Fit to a Nonmetric Hypothesis. *Psychometrika*, 29(1), 1964.

[22] Y. Matsuo and M. Ishizuka. Keyword Extraction from a Single Document using Word Co-ocurrence Statistical Information. *Int. Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.

[23] J. Nolan. Stable Distributions—Models for Heavy Tailed Data. http://academic2.american.edu/~jpnolan/stable/, 2005.

[24] T. Rose, M. Stevenson, and M. Whitehead. The Reuters Corpus Volume 1. From Yesterday's News to Tomorrow's Language Resources. In *Proc. of the third int. conference on language resources and evaluation*, 2002.

[25] S. Rump. Verification of Positive Definiteness. *BIT Numerical Mathematics*, 46:433–452, 2006.

[26] B. Stein. Fuzzy-Fingerprints for Text-Based IR. In *Proc. of the 5th Int. Conference on Knowledge Management, Graz*, Journal of Universal Computer Science, 2005.

[27] B. Stein and S. Meyer zu Eißen. Near Similarity Search and Plagiarism Analysis. In *From Data and Information Analysis to Knowledge Engineering*. Springer, 2006.

[28] R. Weber, H. Schek, and S. Blott. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-dimensional Spaces. In *Proc. of the 24th VLDB conference*, 1998.

[29] H. Yang and J. Callan. Near-Duplicate Detection by Instance-level Constrained Clustering. In *Proc. of the 29th conference on research and development in IR*, 2006.