Hash-based Indexing: Application, Impact, and Realization Alternatives

Benno Stein and Martin Potthast

Bauhaus University Weimar
Web-Technology and Information Systems
Text-based Information Retrieval (TIR)

Motivation

Consider a set of documents $D$.

Term query—the most common retrieval task: Find all documents $D' \subset D$ containing a set of query terms.

- Best practice: Index $D$ using an inverted file.
- Implemented by well-known web search engines.
Text-based Information Retrieval (TIR)

Motivation

Document query—given a document \(d\):
Find all documents \(D' \subset D\) with a high similarity to \(d\).

→ Use cases: plagiarism analysis, query by example.

→ Naive approach: Compare \(d\) with each \(d' \in D\).

In detail:
Construct document models for \(D\) and \(d\) obtaining \(D\) and \(d\).
Employ a similarity function \(\varphi : D \times D \to [0, 1]\).

Is it possible to be faster than the naive approach?
Background

Nearest Neighbour Search

Given a set $D$ of $m$-dimensional points and a point $d$:
Find the point $d' \in D$ which is nearest to $d$.

Finding $d'$ cannot be done better than in $O(|D|)$ time if $m$ exceeds 10.
[Weber et. al. 1998]

In our case: $1.000 < m < 100.000$
Background

**Approximate Nearest Neighbour Search**

Given a set $D$ of $m$-dimensional points and a point $d$:
Find some points $D' \subset D$ from a certain $\varepsilon$-neighbourhood of $d$.

Finding $D'$ can be done in $O(1)$ time with high probability by means of hashing.

The dimensionality $m$ does not affect the runtime of their algorithm.

[Indyk and Motwani 1998]
Text-based Information Retrieval (TIR)

Nearest Neighbour Search

**Retrieval tasks**

- Index-based retrieval
  - Grouping
  - Similarity search
    - Classification
    - Partial document similarity
    - Complete document similarity
- Categorization
- Near-duplicate detection

**Use cases**

- focused search, efficient search (cluster hypothesis)
- preparation of search results
- plagiarism analysis
- query by example
- directory maintenance

Approximate retrieval results are often acceptable.
Similarity Hashing

Introduction

With standard hash functions collisions occur accidently.

In similarity hashing collisions shall occur purposefully where the purpose is “high similarity”.

Given a similarity function \( \varphi \) a hash function

\[ h_\varphi : D \rightarrow U \quad \text{with} \quad U \subset \mathbb{N} \]

resembles \( \varphi \) if it has the following property [Stein 2005]:

\[ h_\varphi(d) = h_\varphi(d') \Rightarrow \varphi(d, d') \geq 1 - \varepsilon \quad \text{with} \quad d, d' \in D, \quad 0 < \varepsilon \ll 1 \]
Similarity Hashing

Index Construction

Given a similarity hash function \( h_\varphi \) a hash index

\[
\mu_h : D \rightarrow \mathcal{D} \quad \text{width } \mathcal{D} = \mathcal{P}(D)
\]

is constructed using

- a hash table \( \mathcal{T} \)
- a standard hash function \( h : U \rightarrow \{1, \ldots, |\mathcal{T}|\} \)
Similarity Hashing

Index Construction

Given a similarity hash function $h_\phi$ a hash index

$$\mu_h : \mathcal{D} \rightarrow \mathcal{D} \quad \text{width } \mathcal{D} = \mathcal{P}(\mathcal{D})$$

is constructed using

- a hash table $\mathcal{T}$
- a standard hash function $h : U \rightarrow \{1, \ldots, |\mathcal{T}|\}$

To *index* a set of documents $\mathcal{D}$ given their models $\mathcal{D}$,

- compute for each $d \in \mathcal{D}$ its hash value $h_\phi(d)$
- store a reference to $d$ in $\mathcal{T}$ at storage position $h(h_\phi(d))$

To *search* for documents similar to $d$ given its model $d$,

- return the bucket in $\mathcal{T}$ at storage position $h(h_\phi(d))$
Similarity Hash Functions

Fuzzy-Fingerprinting (FF)  [Stein 2005]

All words having the same prefix belong to the same prefix class.
Similarity Hash Functions

Locality-Sensitive Hashing (LSH)
[Indyk and Motwani 1998, Datar et. al. 2004]

Vector space with sample document and random vectors

\[ a_i \cdot d \]

Dot product computation

Real number line

Fingerprint

The results of the $k$ dot products are summed.
Similarity Hash Functions

Adjusting Recall and Precision

Recall:

\[ h_\varphi \]
Similarity Hash Functions

Adjusting Recall and Precision

Recall:

\[ h_{\varphi} \]

(FF) # fuzzy schemes.

(LSH) # random vector sets.

A set of hash values per document is called fingerprint.
Similarity Hash Functions

Adjusting Recall and Precision

Recall:

\( h'_{\varphi} \) # fuzzy schemes.

(FF) # fuzzy schemes.

(LSH) # random vector sets.

A set of hash values per document is called fingerprint.

Precision:

(FF) # prefix classes or

# intervals per fuzzy scheme.

(LSH) # random vectors.
Experimental Setting

Three test collections for three retrieval situations

1. **Web results**: 100,000 documents from a focused search.
   → Documents as Web retrieval systems return them.

2. **RCV1**: 100,000 documents from the Reuters Corpus.
   → Documents as corporations organize them.

3. **Plagiarism corpus**: 3,000 documents with high similarity.
   → Documents as they appear in plagiarism analysis.

Retrieval tasks: Similarity Search, Near-Duplicate Detection

Precision and Recall were recorded for similarity thresholds ranging from 0 to 1.
Introduction
Hash-based Indexing Methods
Comparative Study

Results

Web results
Recall vs. Similarity

RCV1
Recall vs. Similarity

FGIR'06 Oct. 11th, 2006 Stein/Potthast
Results

![Graphs showing recall and precision for Web results and RCV1 datasets with LSH and FF methods.](image)
Results

Recall

Precision

Plagiarism corpus

FF
LSH
Summary

Near-duplicate detection in plagiarism situation:
- FF outperforms LSH in terms of Precision and Recall.

Retrieval tasks in standard corpora:
- FF outperforms LSH in terms of Precision.
- FF and LSH perform equally good in terms of Recall.

Conclusions:
- Both hash-based indexing methods may be applied in TIR.
- FF has an advantage in precision which directly affects runtime performance.

None of the hash-based indexing methods is limited to TIR. The only prerequisite is a reasonable vector representation.
Thank you!