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 similarty to d.

➜ Use cases: plagiarism analysis, query by example.

www.**turing**.org.uk/ - 11k - Cached - Similar pages - Filter

➜ Naive approach: Compare d with each $d' \in D$.

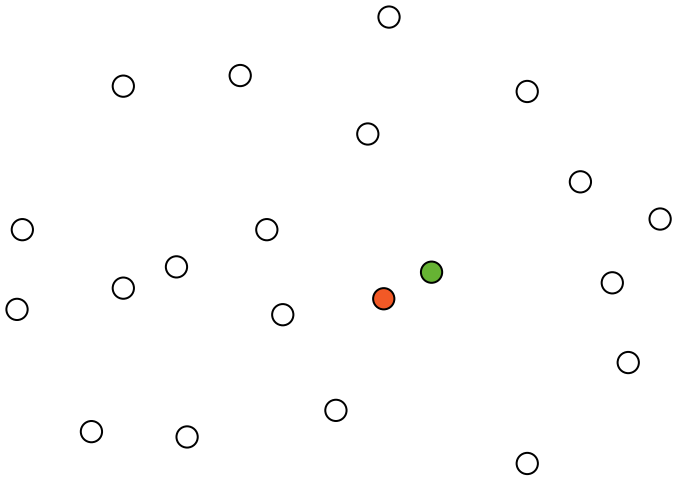
In detail:
Construct document models for D and d obtaining $\mathbf{D}$ and $\mathbf{d}$.
Employ a similarity function $\varphi : \mathbf{D} \times \mathbf{D} \rightarrow [0, 1]$.

Is it possible to be faster than the naive approach?

Background

Nearest Neighbour Search

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



Finding $\mathbf{d}'$ cannot be done better than in $\mathcal{O}(|\mathbf{D}|)$ time if m exceeds 10.

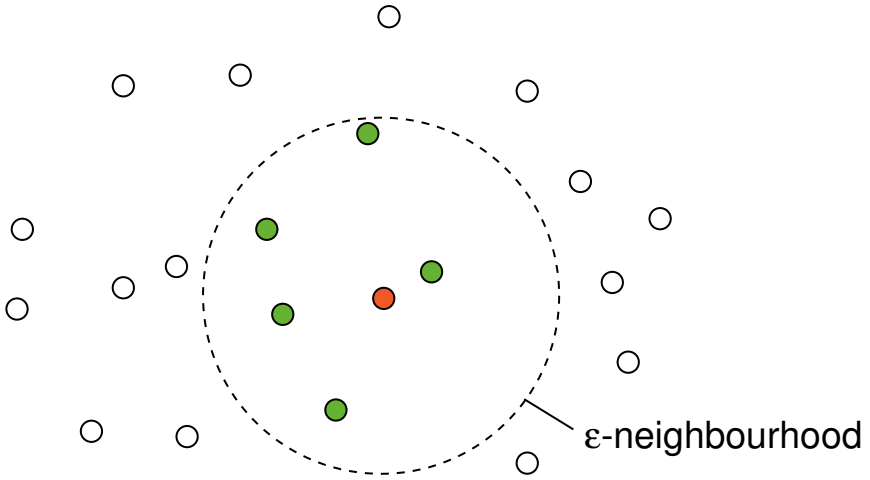
[Weber *et. al.* 1998]

In our case: $1.000 < m < 100.000$

Background

Approximate Nearest Neighbour Search

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

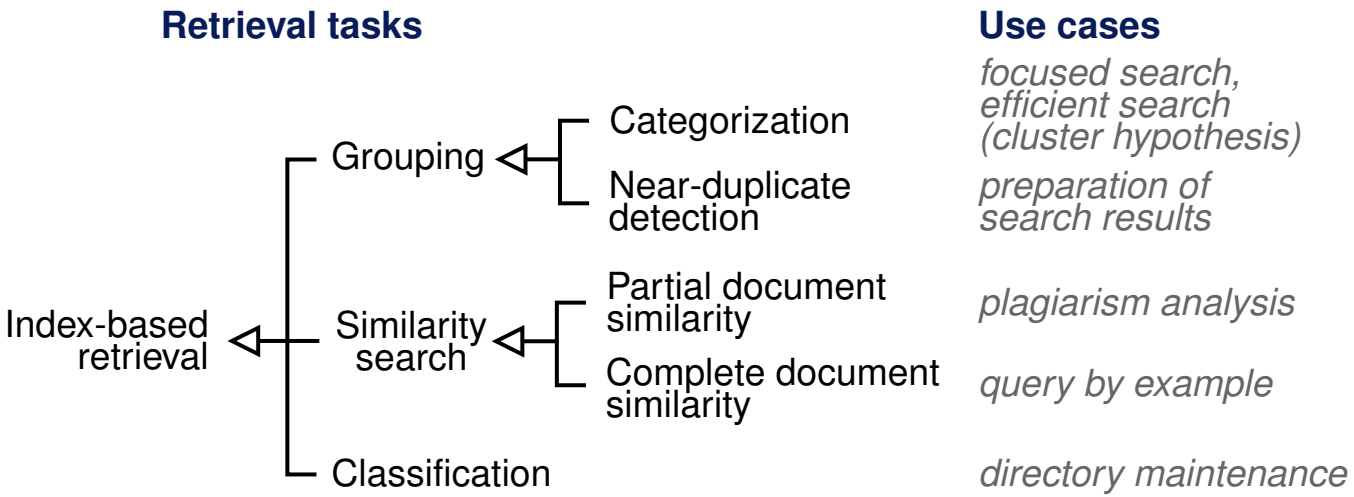


Finding $\mathbf{D}'$ can be done in $\mathcal{O}(1)$ time with high probabilty by means of hashing. [Indyk and Motwani 1998]

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

Text-based Information Retrieval (TIR)

Nearest Neighbour Search



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 φ a hash function

$$h_\varphi : \mathbf{D} \rightarrow U \qquad \text{with } U \subset \mathbf{N}$$

resembles φ if it has the following property [Stein 2005]:

$$h_\varphi(\mathbf{d}) = h_\varphi(\mathbf{d}') \;\Rightarrow\; \varphi(\mathbf{d}, \mathbf{d}') \geq 1 - \varepsilon \qquad \text{with } \mathbf{d}, \mathbf{d}' \in \mathbf{D}, 0 < \varepsilon \ll 1$$

Similarity Hashing

Index Construction

Given a similarity hash function h_φ a hash index

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

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_φ a hash index

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

is constructed using

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

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

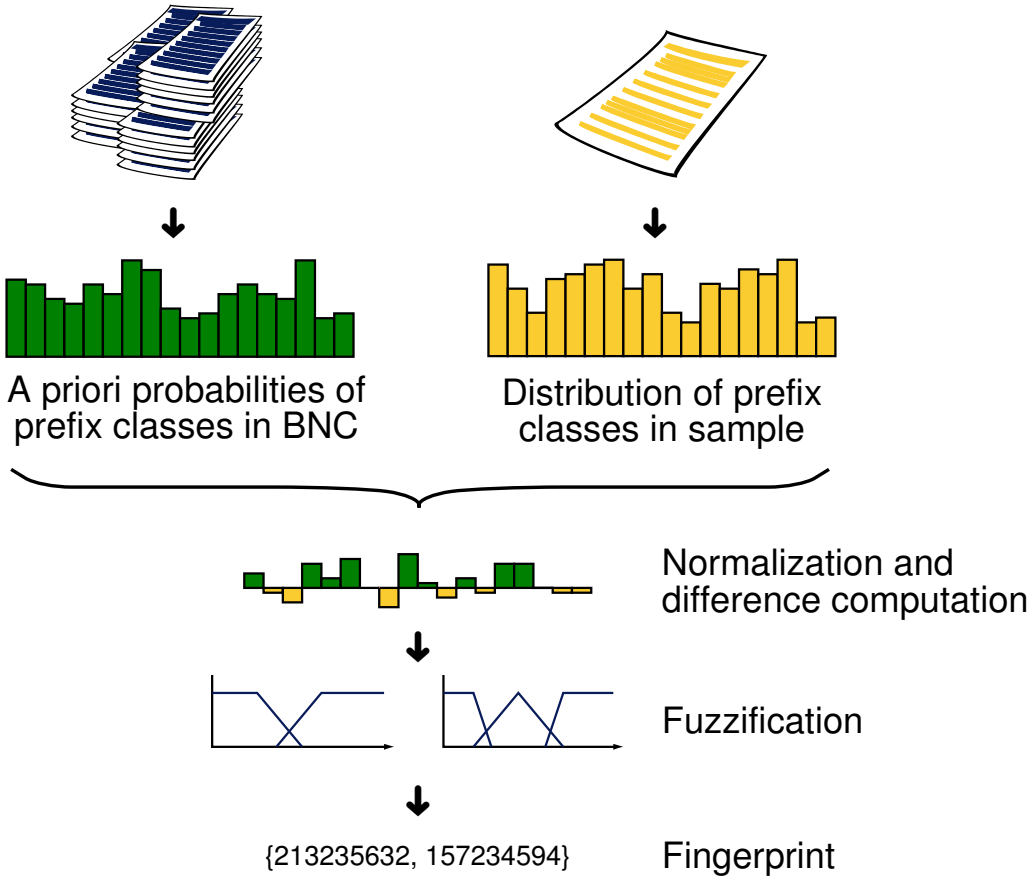
- compute for each $\mathbf{d} \in \mathbf{D}$ its hash value $h_\varphi(\mathbf{d})$
- store a reference to d in $\mathcal{T}$ at storage position $h(h_\varphi(\mathbf{d}))$

To *search* for documents similar to d given its model $\mathbf{d}$,

- return the bucket in $\mathcal{T}$ at storage position $h(h_\varphi(\mathbf{d}))$

Similarity Hash Functions

Fuzzy-Fingerprinting (FF) [Stein 2005]

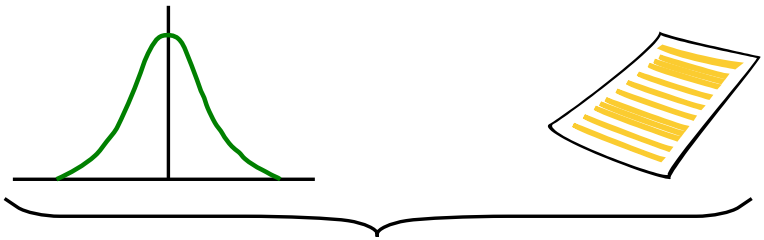


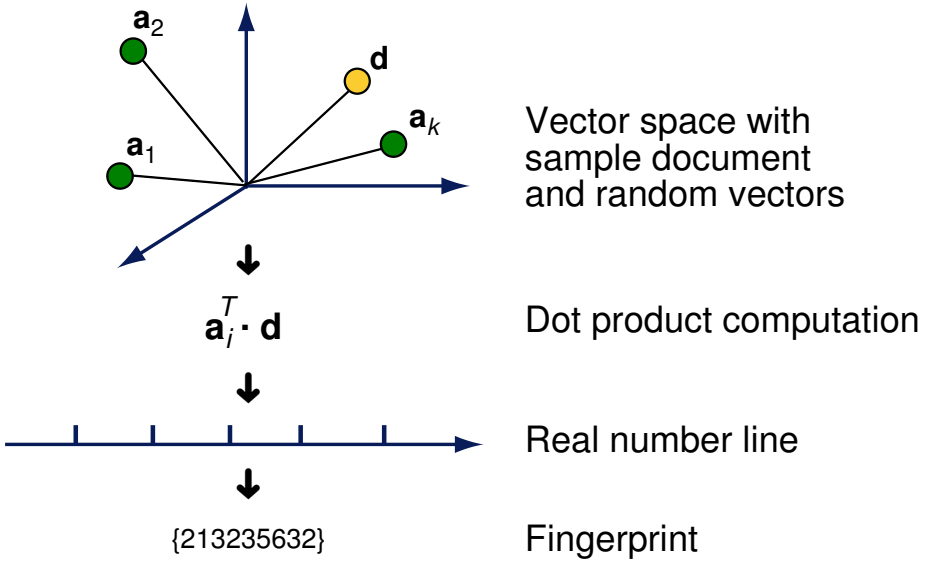
All words having the same prefix belong to the same prefix class.

Similarity Hash Functions

Locality-Sensitive Hashing (LSH)

[Indyk and Motwani 1998, Datar *et. al.* 2004]



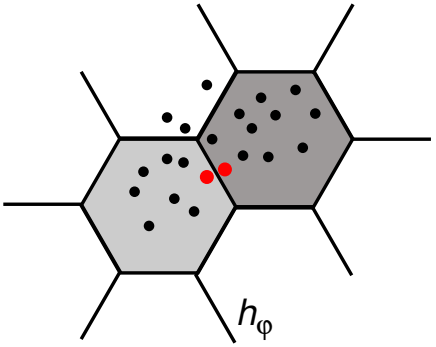


The results of the k dot products are summed.

Similarity Hash Functions

Adjusting Recall and Precision

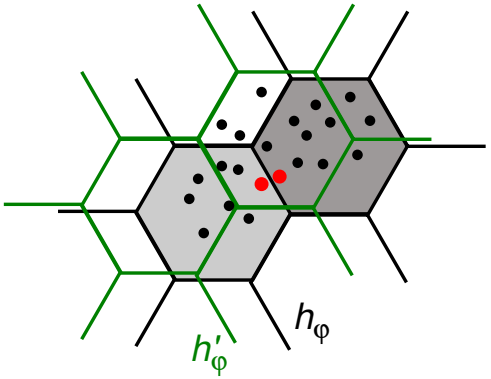
Recall:



Similarity Hash Functions

Adjusting Recall and Precision

Recall:



(FF) # fuzzy schemes.

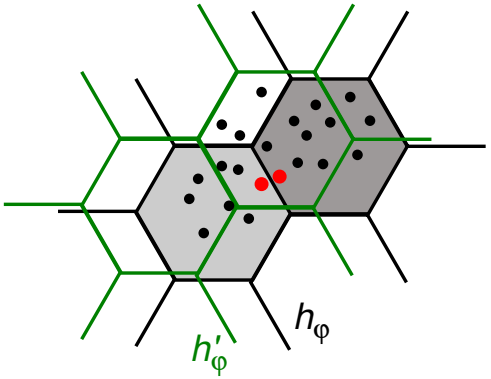
(LSH) # random vector sets.

A set of hash values per document is called fingerprint.

Similarity Hash Functions

Adjusting Recall and Precision

Recall:



(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.

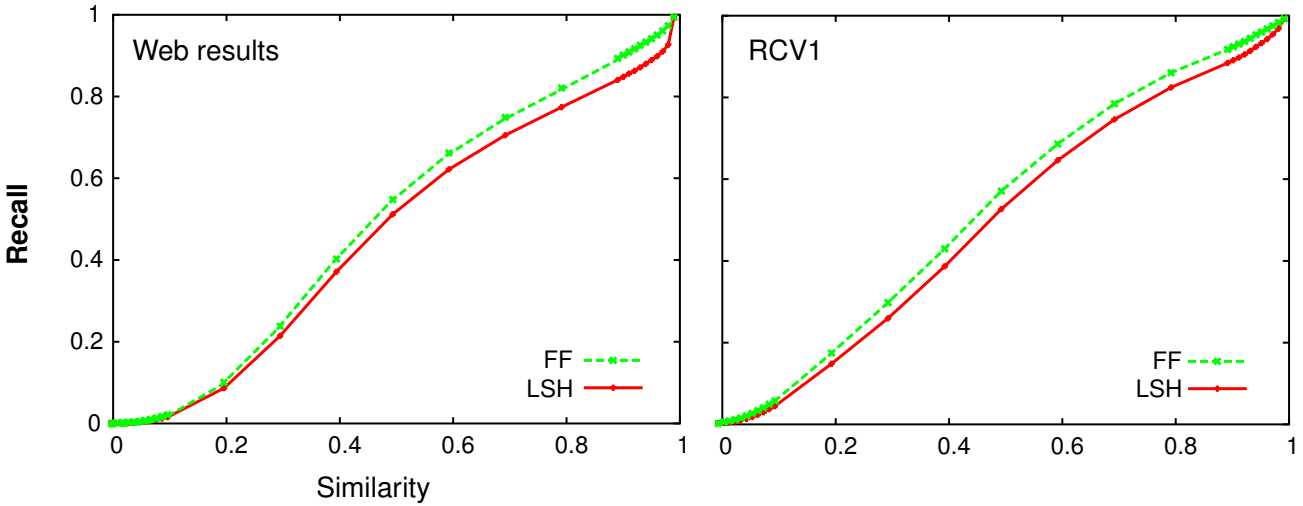
3. Plagiarism corpus: 3.000 documents with high similarity.

➜ Documents as they appear in plagiarism analysis.

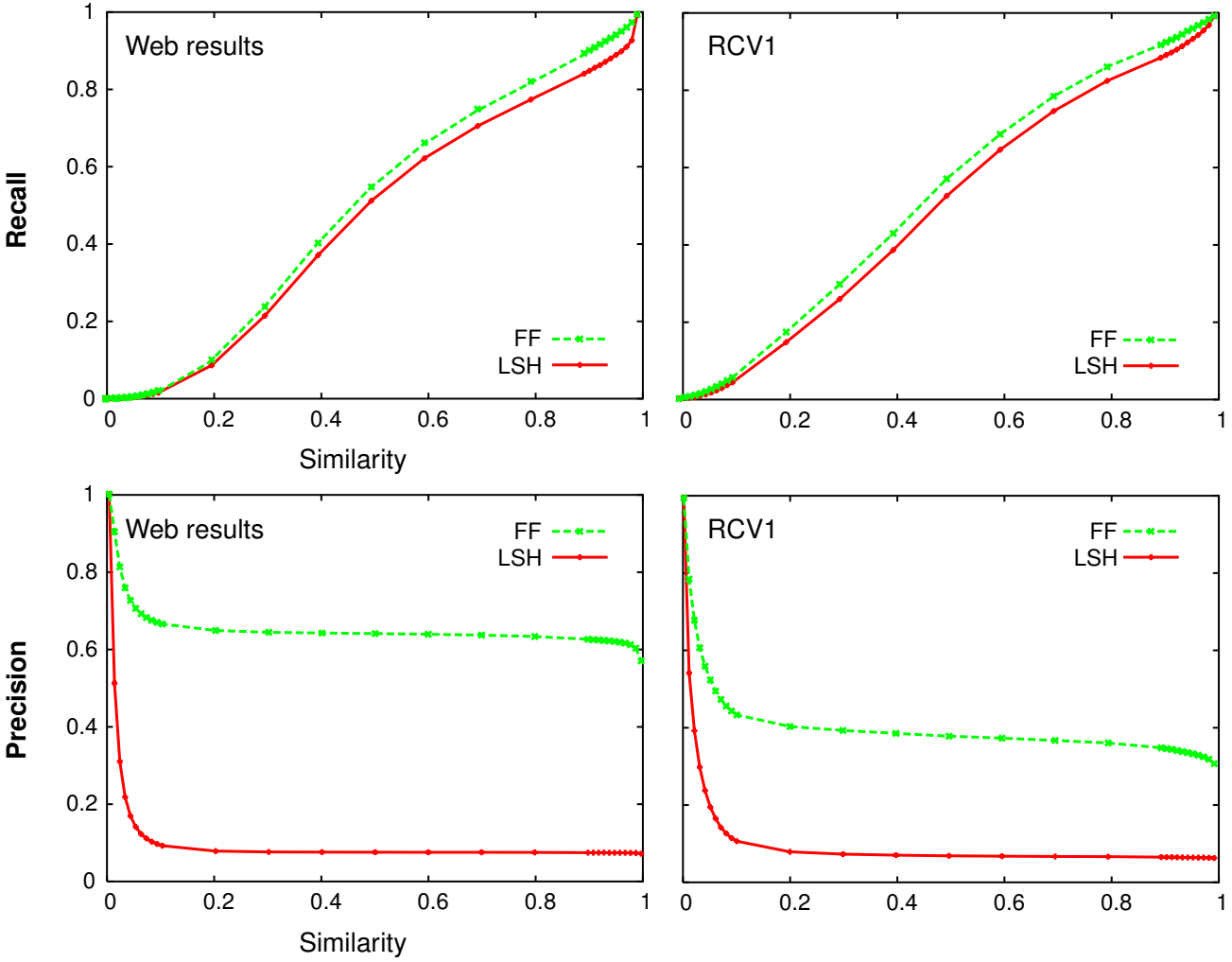
Retrieval tasks: Similarity Search, Near-Duplicate Detection

Precision and Recall were recorded for similarity thresholds ranging from 0 to 1.

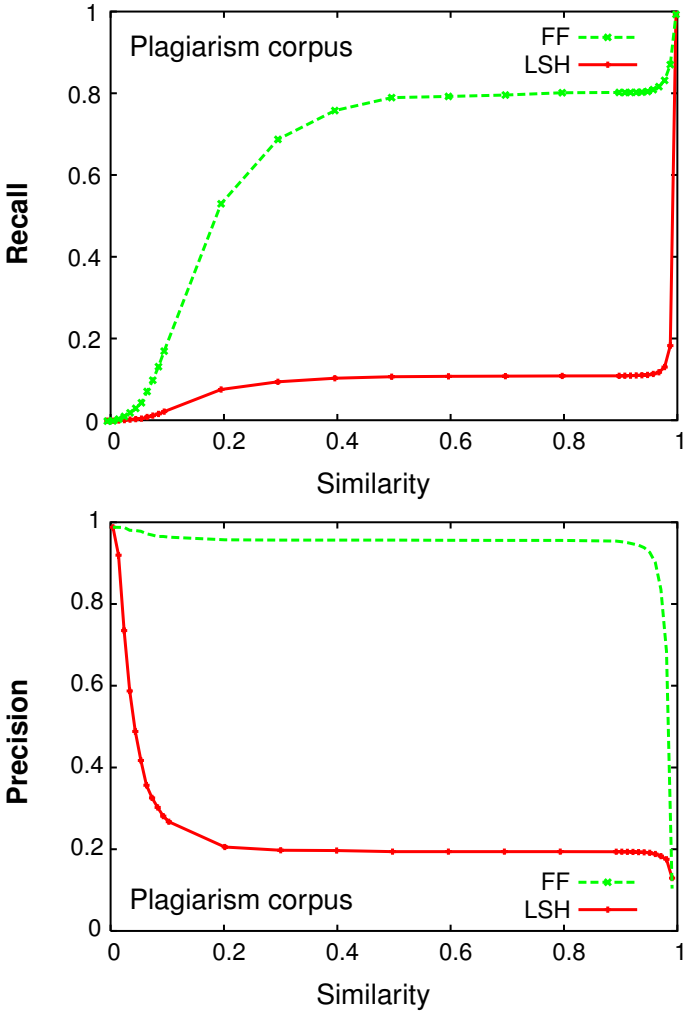
Results



Results



Results



Summary

Near-duplicate detection in plagiarism situation:

- FF outperforms LSH in terms of Precsion 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!