

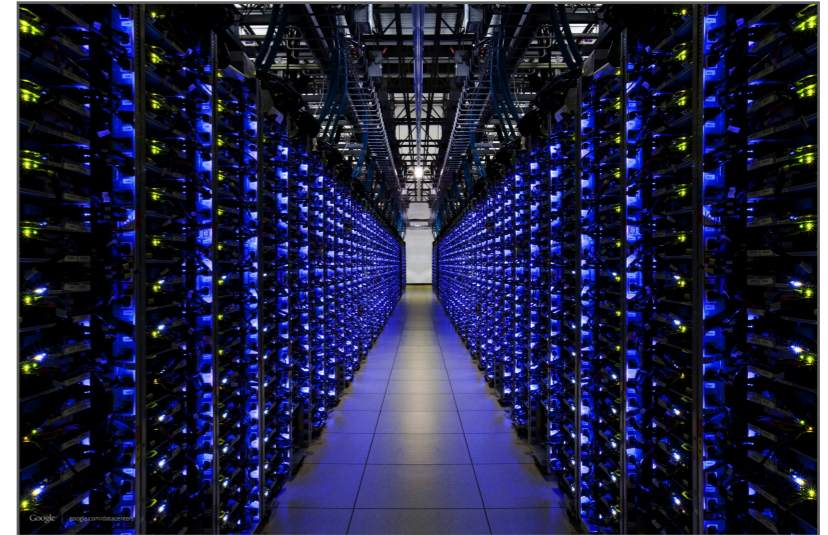
# V.4 MapReduce

- 1. System Architecture**
- 2. Programming Model**
- 3. Hadoop**

**Based on MRS Chapter 4 and RU Chapter 2**

# Why MapReduce?

- Large clusters of **commodity computers** (as opposed to few supercomputers)
- Challenges:
  - **load balancing**
  - **fault tolerance**
  - **ease of programming**
- **MapReduce**
  - system for distributed data processing
  - programming model
- Full details: [Ghemawat et al. '03][Dean and Ghemawat '04]



Jeff Dean



Sanjay Ghemawat

# Why MapReduce?

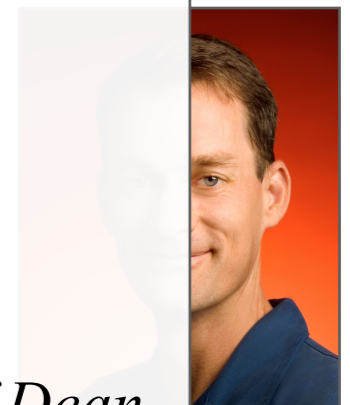
- Large clusters of **commodity computers** (as opposed to few supercomputers)



## Jeff Dean Facts:

- *When Jeff Dean designs software, he first codes the binary and then writes the source as documentation.*
- *load balancing*  
*Compilers don't warn Jeff Dean. Jeff Dean warns compilers.*
- *fault tolerance*  
*Jeff Dean's keyboard has two keys: 1 and 0.*
- *ease of programming*  
*When Graham Bell invented the telephone, he saw a missed call from Jeff Dean.*

- *Source: <http://www.quora.com/Jeff-Dean/What-are-all-the-Jeff-Dean-facts>*



Jeff Dean

- system for distributed data processing
- programming model

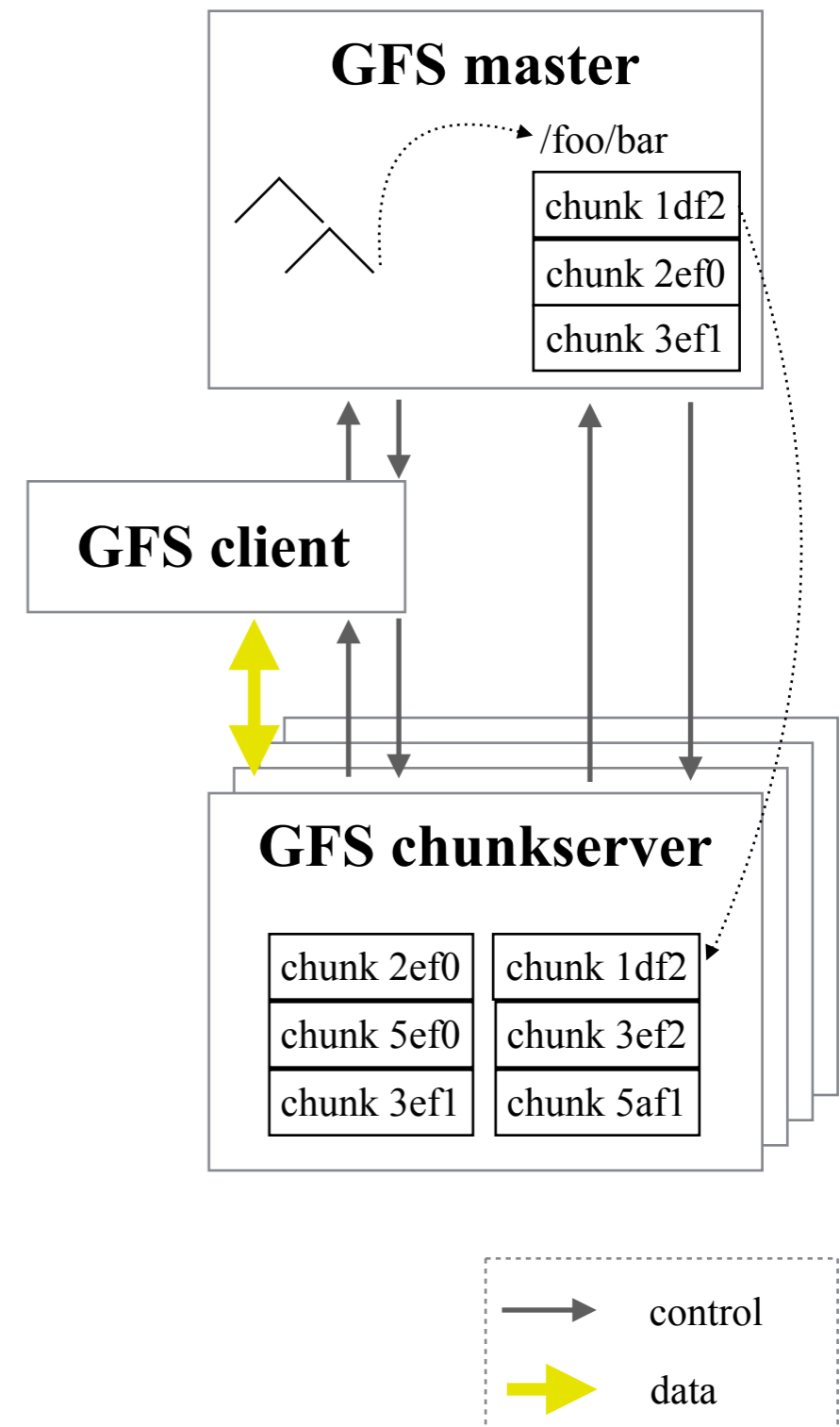


Sanjay Ghemawat

- Full details: [Ghemawat et al. '03][Dean and Ghemawat '04]

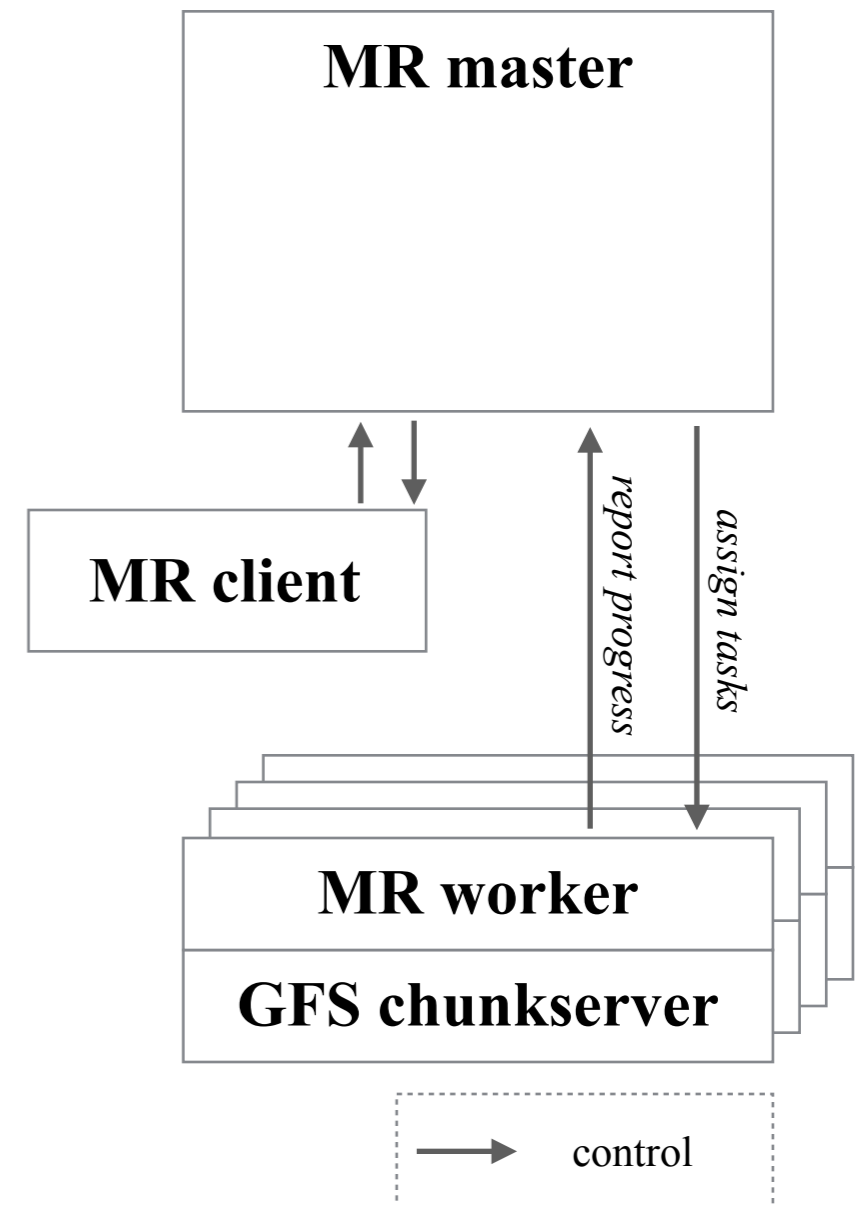
# 1. System Architecture

- **Google File System (GFS)**
  - distributed file system for large clusters
  - tunable replication factor
- **single master**
  - manages namespace (/home/user/data)
  - coordinates replication of data chunks
  - first point of contact for clients
- **many chunkservers**
  - keep data chunks (typically 64 MB)
  - send/receive data chunks to/from clients
- Full details: [Ghemawat et al. '03]



# System Architecture (cont'd)

- **MapReduce (MR)**
  - system for distributed data processing
  - moves computation to the data for locality
  - copes with failure of workers
- **single master**
  - coordinates execution of job
  - (re-)assigns map/reduce tasks to workers
- **many workers**
  - execute assigned map/reduce tasks
- Full details: [Dean and Ghemawat '04]



## 2. Programming Model

- Inspired by **functional programming** (i.e., no side effects)
- Input/output are **key-value pairs**  $(k, v)$  (e.g., string and int)
- Users implement two functions
  - ***map***:  $(k1, v1) \Rightarrow list(k2, v2)$
  - ***reduce***:  $(k2, list(v2)) \Rightarrow list(k3, v3)$  with input sorted by key  $k2$
- Anatomy of a MapReduce job
  - Workers execute ***map***() on their portion of the input data in GFS
  - Intermediate data from ***map***() is **partitioned and sorted**
  - Workers execute ***reduce***() on their partition and write output data to GFS
- Users may implement ***combine***() for local aggregation of intermediate data and ***compare***() to control how data is sorted

# WordCount

- Problem: Count how often every word  $w$  occurs in the document collection (i.e., determine  $cf(w)$ )

```
map(long did, string content) {  
    for(string word : content.split()) {  
        emit(word, 1)  
    }  
}
```

```
reduce(string word, list<int> counts) {  
    int total = 0  
    for(int count : counts) {  
        total += count  
    }  
    emit(word, total)  
}
```

# Execution of WordCount

```
map(long did, string content) {  
    for(string word : content.split()) {  
        emit(word, 1)  
    }  
}
```

```
reduce(string word, list<int> counts) {  
    int total = 0  
    for(int count : counts) {  
        total += count  
    }  
    emit(word, total)  
}
```



# Execution of WordCount

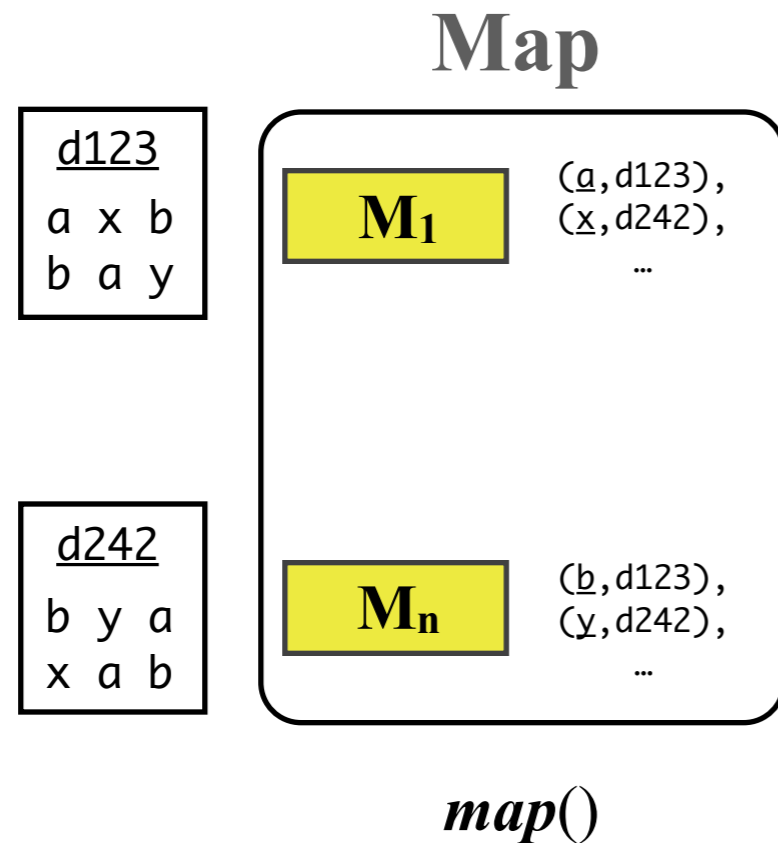
```
d123  
a x b  
b a y
```

```
d242  
b y a  
x a b
```

```
map(long did, string content) {  
    for(string word : content.split()) {  
        emit(word, 1)  
    }  
}
```

```
reduce(string word, list<int> counts) {  
    int total = 0  
    for(int count : counts) {  
        total += count  
    }  
    emit(word, total)  
}
```

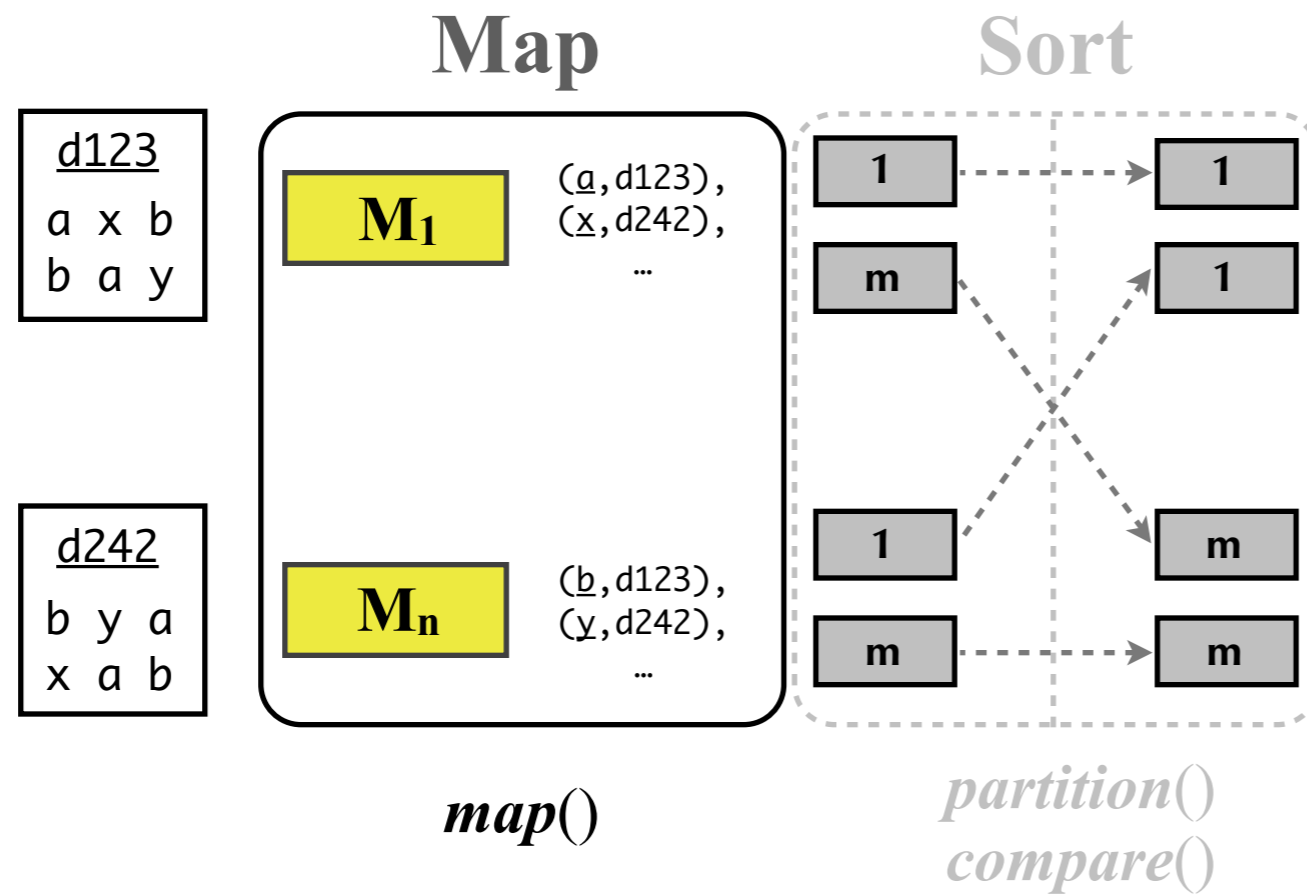
# Execution of WordCount



```
map(long did, string content) {  
    for(string word : content.split()) {  
        emit(word, 1)  
    }  
}
```

```
reduce(string word, list<int> counts) {  
    int total = 0  
    for(int count : counts) {  
        total += count  
    }  
    emit(word, total)  
}
```

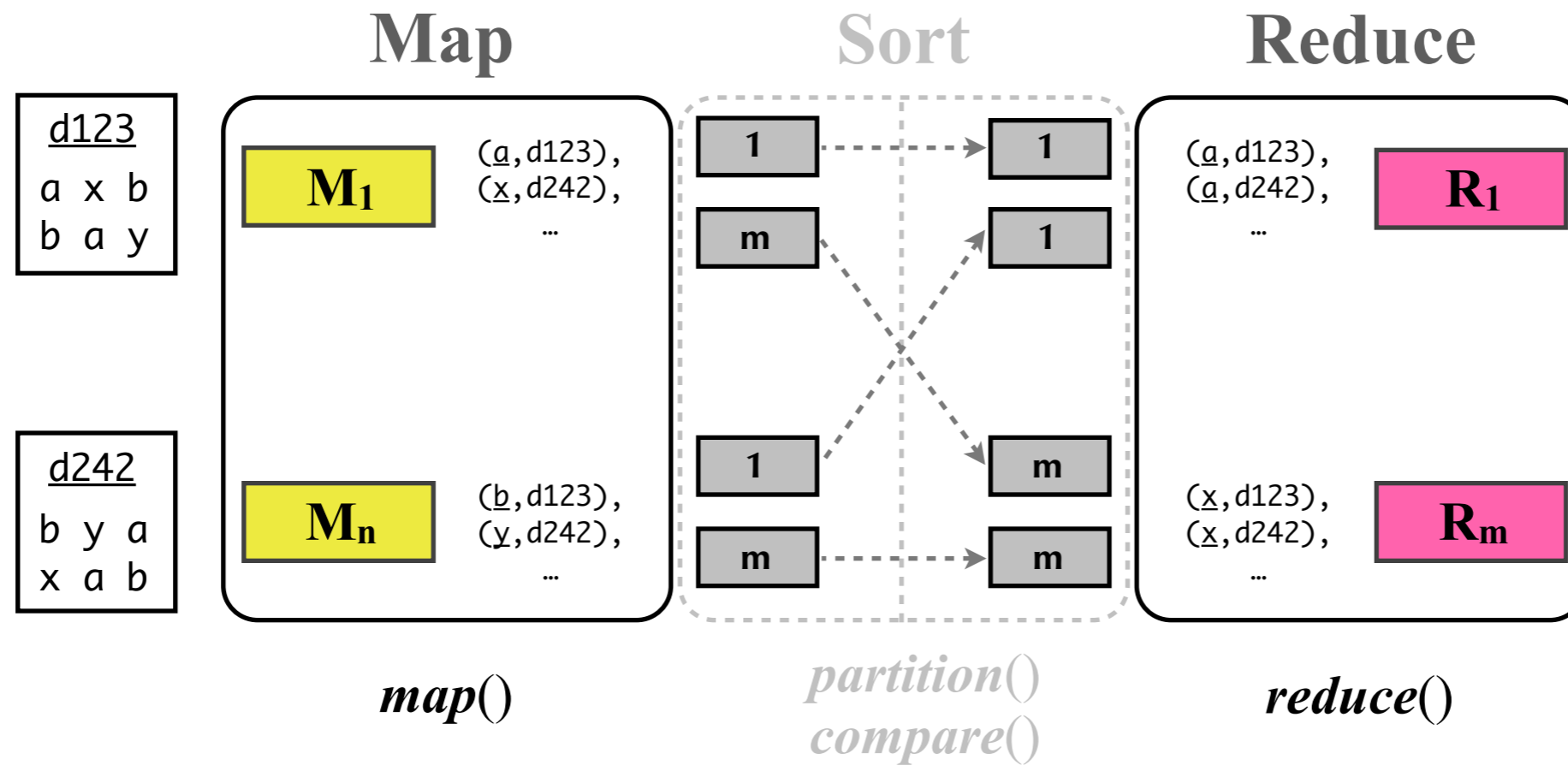
# Execution of WordCount



```
map(long did, string content) {  
    for(string word : content.split()) {  
        emit(word, 1)  
    }  
}
```

```
reduce(string word, list<int> counts) {  
    int total = 0  
    for(int count : counts) {  
        total += count  
    }  
    emit(word, total)  
}
```

# Execution of WordCount



```

map(long did, string content) {
    for(string word : content.split()) {
        emit(word, 1)
    }
}

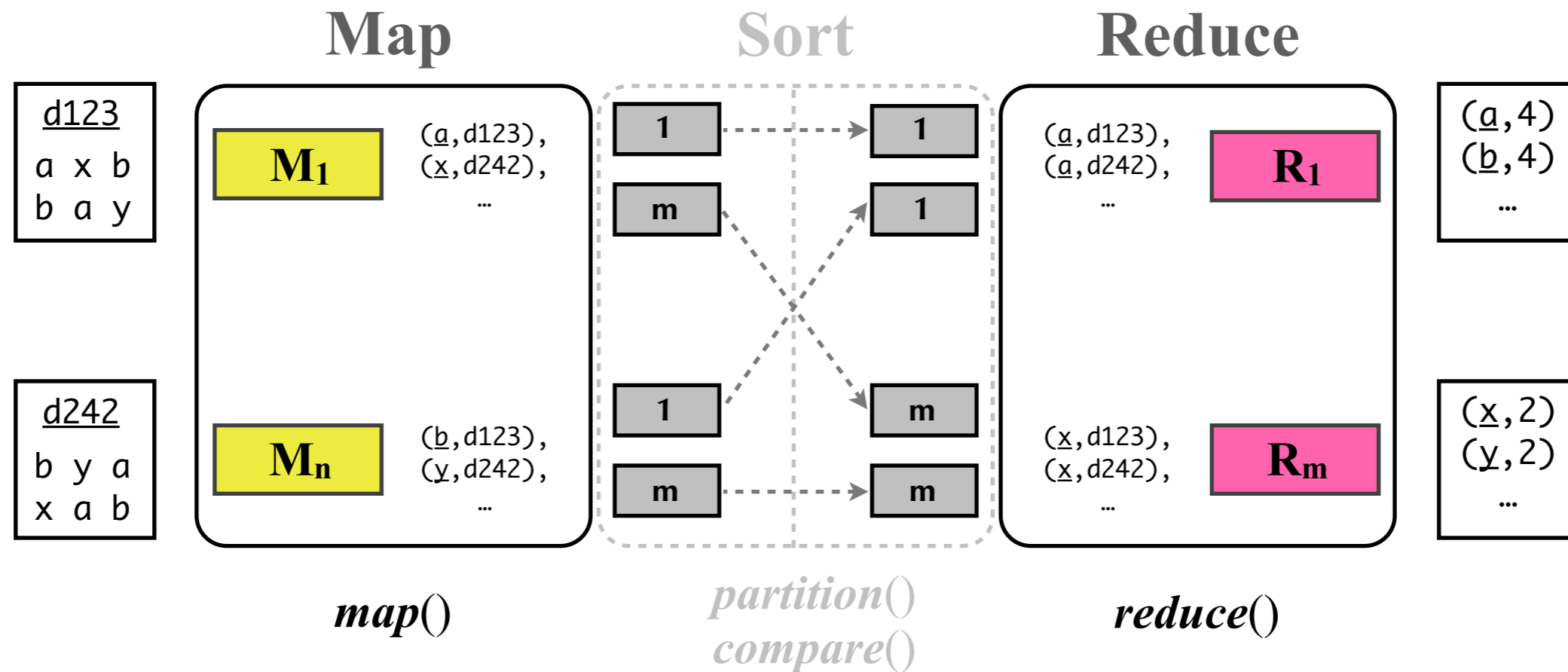
```

```

reduce(string word, list<int> counts) {
    int total = 0
    for(int count : counts) {
        total += count
    }
    emit(word, total)
}

```

# Execution of WordCount



```
map(long did, string content) {
    for(string word : content.split()) {
        emit(word, 1)
    }
}
```

```
reduce(string word, list<int> counts) {
    int total = 0
    for(int count : counts) {
        total += count
    }
    emit(word, total)
}
```

# Inverted Index Construction

- Problem: Construct a positional inverted index with postings containing positions (e.g.,  $\{d_{123}, 3, [1, 9, 20]\}$ )

```
map(long did, string content) {  
    int pos = 0  
    map<string, list<int>> positions = new map<string, list<int>>()  
    for(string word : content.split()) { // tokenize document content  
        positions.get(word).add(pos++) // aggregate word positions  
    }  
    for(string word : map.keys()) {  
        emit(word, new posting(did, positions.get(word))) // emit posting  
    }  
}
```

```
reduce(string word, list<posting> postings) {  
    postings.sort() // sort postings (e.g., by did)  
    emit(word, postings) // emit posting list  
}
```

# 3. Hadoop

- **Open source implementation** of GFS and MapReduce
- **Hadoop File System (HDFS)**
  - name node (master)
  - data node (chunkserver)
- **Hadoop MapReduce**
  - job tracker (master)
  - task tracker (worker)
- Has been successfully deployed on clusters of **10,000s machines**
- **Productive use** at Yahoo!, Facebook, and many more



Doug Cutting

# Jim Gray Benchmark

- **Jim Gray Benchmark:**
  - sort large amount of 100 byte records (10 first bytes are keys)
  - **minute sort:** sort as many records as possible in under a minute
  - **gray sort:** must sort at least 100 TB, must run at least 1 hours
- **November 2008:** Google sorts 1 TB in 68 s and 1 PB in 6:02 h on MapReduce using a cluster of 4,000 computers and 48,000 hard disks  
<http://googleblog.blogspot.com/2008/11/sorting-1pb-with-mapreduce.html>
- **May 2011:** Yahoo! sorts 1 TB in 62 s and 1 PB in 16:15 h on Hadoop using a cluster of approximately 3,800 computers 15,200 hard disks  
[http://developer.yahoo.com/blogs/hadoop/posts/2009/05/hadoop\\_sorts\\_a\\_petabyte\\_in\\_162/](http://developer.yahoo.com/blogs/hadoop/posts/2009/05/hadoop_sorts_a_petabyte_in_162/)



# Summary of V.4

- **MapReduce**  
a system of distributed data processing  
a programming model
- **Hadoop**  
a widely-used open-source implementation of MapReduce

# Additional Literature for V.4

- **Apache Hadoop** (<http://hadoop.apache.org>)
- **J. Dean and S. Ghemawat:** *MapReduce: Simplified Data Processing on Large Clusters*, OSDI 2004
- **J. Dean and S. Ghemawat:** *MapReduce: Simplified Data Processing on Large Clusters*, CACM 51(1):107-113, 2008
- **S. Ghemawat, H. Gobioff, and S.-T. Leung:** *The Google File System*, SOPS 2003
- **J. Lin and C. Dyer:** *Data-Intensive Text Processing with MapReduce*, Morgan & Claypool Publishers, 2010 (<http://lintool.github.io/MapReduceAlgorithms>)

## **V.5 Near-Duplicate Detection**

- 1. Shingling**
- 2. SpotSigs**
- 3. Min-Wise Independent Permutations**
- 4. Locality-Sensitive Hashing**

**Based on MRS Chapter 19 and RU Chapter 3**

# Near-Duplicate Detection

# Near-Duplicate Detection

**SFGate.com**  
SFGATE HOME • NEWS **NEW!** • BUSINESS • SPORTS • ENTERTAINMENT • TRAVEL

SEARCH SFGate Web Search by YAHOO!

**Ap** Associated Press

## Obama Takes on Question of Faith

By NEDRA PICKLER, Associated Press Writer  
Monday, January 21, 2008

PRINTABLE E-MAIL SHARE COMMENTS (0) FONT | SIZE: [ ] [ + ] TOOLS SPONSOR: **verizon**

(01-21) 04:22 PST Columbia, S.C. (AP) --

Barack **Obama** is stepping up his effort to correct the misconception that he's a Muslim now that the presidential campaign has hit the Bible Belt.

At a rally to kick off a weeklong campaign for the South Carolina primary, **Obama** tried to set the record straight from an attack circulating widely on the Internet that is designed to play into prejudices against Muslims and fears of terrorism.

---

HOME PAGE MY TIMES TODAY'S PAPER VIDEO MOST POPULAR TIMES TOPICS Get Home Delivery Log In Register Now

# The New York Times U.S.

U.S. All NYT Search Ameriprise Financial

WORLD U.S. N.Y. / REGION BUSINESS TECHNOLOGY SCIENCE HEALTH SPORTS OPINION ARTS STYLE TRAVEL JOBS REAL ESTATE AUTOS

POLITICS WASHINGTON EDUCATION

## Obama Takes on Question of Faith

By THE ASSOCIATED PRESS  
Published: January 21, 2008

**Filed at 7:16 a.m. ET**

COLUMBIA, S.C. (AP) -- Barack Obama is stepping up his effort to correct the misconception that he's a Muslim now that the presidential campaign has hit the Bible Belt.

At a rally to kick off a weeklong campaign for the South Carolina primary, Obama tried to set the record straight from an attack circulating widely on the Internet that is designed to play into prejudices against Muslims and fears of terrorism.

SIGN IN TO E-MAIL OR SAVE THIS  
PRINT  
ARTICLE TOOLS SPONSORED BY **THE SAVAGES**

### MOST POPULAR

E-MAILED BLOGGED SEARCHED

1. Nicholas D. Kristof: Hillary, Barack, Experience
2. Paul Krugman: Debunking the Reagan Myth
3. Pregnancy Problems Tied to Caffeine
4. Maureen Dowd: Red, White and Blue Tag Sale
5. Roger Cohen: U.S. Soldiers and Shoppers Hit the Wall
6. Stocks Plunge Worldwide on Fears of a U.S. Recession
7. New York Measuring Teachers by Test Scores
8. Op-Ed Contributor: Radical Love Gets a Holiday
9. A Cutting Tradition

# Near-Duplicate Detection

**SFGate.com**  
 SFGATE HOME • NEWS **NEW!** • BUSINESS • SPORTS • ENTERTAINMENT • TRAVEL

HOME PAGE | MY TIMES | TODAY'S PAPER | VIDEO | MOST POPULAR | TIMES TOPICS | Get Home Delivery | Log In | Register Now

**The New York Times** U.S.  Ameriprise Financial

U.S. All NYT Search

TOLOGY | SCIENCE | HEALTH | SPORTS | OPINION | ARTS | STYLE | TRAVEL | JOBS | REAL ESTATE | AUTOS

**BROWN eyed BAKER** *sweet. savory. sinful.*

HOME ABOUT RECIPE INDEX ARCHIVES RESOURCES UNPLUGGED RECIPE E

**Allspice Crumb Muffins**  
 March 5, 2010 60 comments »

**Obama Take**  
 By NEDRA PICKLER  
 Monday, January 21, 2013

PRINTABLE

(01-21) 04:22 PM

**Barack Obama**  
 that the presiden

At a rally to kick  
 set the record str  
 play into prejudi

vendredi 14 juin 2013

about Allspice Crumb Muffins

ccc

allspice-crumb-muffin-main

It's Friday and the weekend is here (cue the trumpets). With the hustle and bustle of typical weekday mornings, many people barely manage a bowl of Cheerios, let alone a homemade treat for breakfast. It seems that pancakes and sausage, fluffy waffles, and jumbo muffins are often reserved for weekend indulgences. Snuggled under your mountain of blankets, convinced that nothing could pry you out of bed after a week of sleep deprivation, you catch a whiff. The bacon sizzling, the sweet maple aroma of sausage, or spiced muffins coming from the oven, and I guarantee you're out of bed in a flash. It seems ingrained in all of us - being roused from sleep with your nose, it's almost inevitable that as soon as your feet hit the floor your stomach is rumbling. And suddenly you can't get to the kitchen fast enough. It's a marvelous way to wake up, and a perfect way to remember to slow down and savor the small things that bring us joy. Whether it be loved ones, the perfect cup of coffee, or a plateful of bacon 😊 No matter your favorite way to enjoy the laziness of a morning where you have nowhere to be, these muffins should be penciled into your weekend-slow-down-and-smell-the-bacon plans.

It's Friday and the weekend is here (cue the trumpets). With the hustle and bustle of typical weekday mornings, many people barely manage a bowl of Cheerios, let alone a homemade treat for breakfast. It seems that pancakes and sausage, fluffy waffles, and jumbo muffins are often reserved for weekend indulgences. Snuggled under your mountain of blankets, convinced that nothing could pry you out of bed after a week of sleep deprivation, you catch a whiff. The bacon sizzling, the sweet maple aroma of sausage, or spiced muffins coming from the oven, and I guarantee you're out of bed in a flash. It seems ingrained in all of us - being roused from sleep with your nose, it's almost inevitable that as soon as your feet hit the floor your stomach is rumbling. And suddenly you can't get to the kitchen fast enough. It's a marvelous way to wake up, and a perfect way to remember to slow down and savor the small things that bring us joy. Whether it be loved ones, the perfect cup of coffee, or a plateful of bacon 😊 No matter your favorite way to enjoy the laziness of a morning where you have nowhere to be, these muffins should be penciled into your weekend-slow-down-and-smell-the-bacon plans.

**Mushroom Growing 4 YOU**

**Chocolate Recipe Guilt Free**  
 PAMELA VINTEN

**Search Ad Roll**

- Vinegar Creamed
- Zucchini Bread
- Homemade Bread Crumbs
- Yummy Caramel Popcorn

Barack, Experience the Reagan Myth Caffeine and Blue Tag Sale and Shoppers Hit the Fears of a U.S. ers by Test Scores Love Gets a Holiday

# Near-Duplicate Detection

The image shows a screenshot of a Stack Overflow question and answer page. The question is titled "Best HashMap initial capacity while indexing a List" and asks for the best initial capacity for a HashMap constructor. The code provided is:

```
List<T> list = myList;
Map<Integer, T> map = new HashMap<Integer, T>(list.size());
for(T item : list) {
    map.put(item.getId(), item);
}
```

The question has 4 votes and 5 answers. The top answer, by Luigi Mendoza, has 4 votes and recommends declaring the variable as Map instead of HashMap and using a profiler. The second answer, by italo, has 2 votes and recommends using a load factor of 1 and a capacity of list.size() + 1.

A near-duplicate page is overlaid on the right side of the image. It shows the same question and code, but with a different answer. The answer by italo has 2 votes and recommends using a load factor of 1 and a capacity of list.size() + 1. The code provided is:

```
HashMap<Integer, T> map = new HashMap<Integer, T>(list.size(), 1.0);
```

The near-duplicate page also shows a snippet of HashMap source code:

```
Here's the relevant piece of HashMap source code:
```

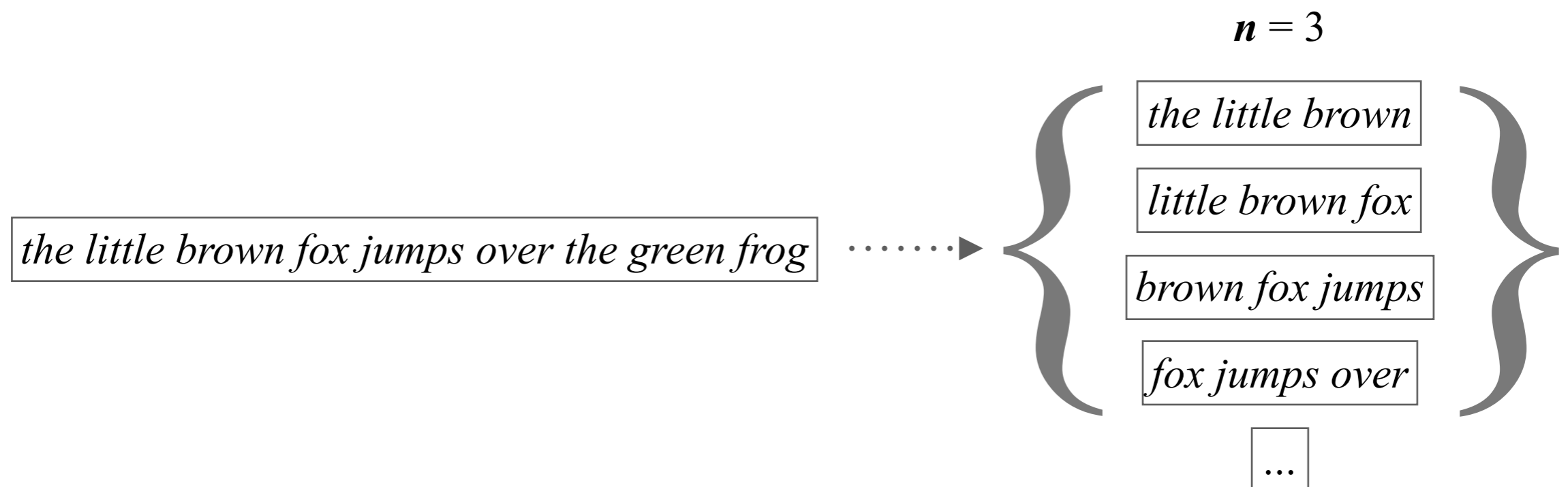
# Near-Duplicate Detection

- Why near-duplicate detection?
  - **smaller indexes** and thus **faster response times**
  - improved **result quality**
- **Building blocks** of a near-duplicate detection method
  - **document representation** (e.g., bag of words, bag of  $n$ -grams, set of links, anchor text of inlinks, set of relevant queries, feature vector)
  - **similarity measure** (e.g., Jaccard coefficient, cosine similarity)
  - **near-duplication detection algorithm**
    - sorting- and indexing-based approaches
    - similarity hashing (e.g., MIPS, LSH)



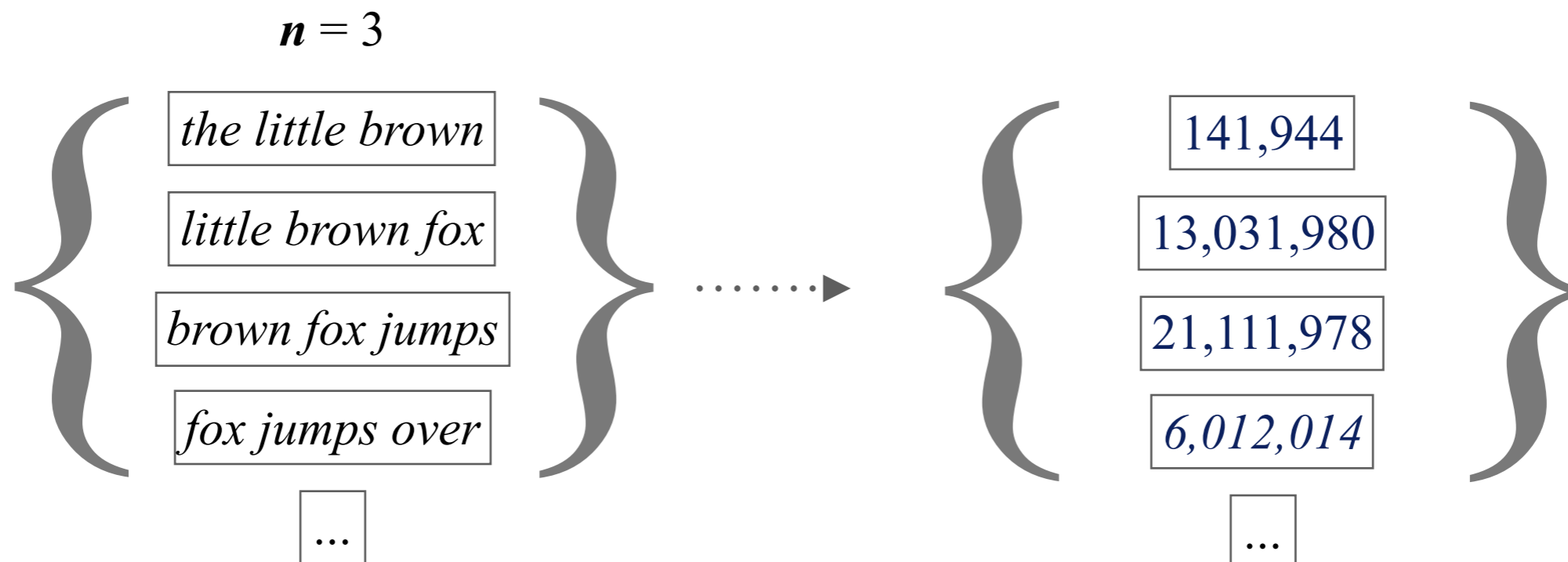
# 1. Shingling

- Observation: Duplicates on the Web are often **slightly perturbed** (e.g., due to different boilerplate, minor rewordings, etc.)
- **Document fingerprinting** (e.g., SHA-1 or MD5) is not effective, since we need to allow for minor differences between documents
- **Shingling** represents document  $d$  as set  $S(d)$  of **word-level  $n$ -grams** (*shingles*) and compares documents based on these sets



# Shingling

- Encode shingles by **hash fingerprints** (e.g., using SHA-1), yielding a set of numbers  $S(d) \subseteq [1, \dots, n]$  (e.g., for  $n = 2^{64}$ )



- Compare suspected near-duplicate documents  $d$  and  $d'$  by

- **Resemblance**  $\frac{|S(d) \cap S(d')|}{|S(d) \cup S(d')|}$  (Jaccard coefficient)

- **Containment**  $\frac{|S(d) \cap S(d')|}{|S(d)|}$  (Relative overlap)

# Shingle-Based Clustering

- Remove near-duplicate document  $d'$  if resemblance or containment is **above a user-specified threshold  $\tau$**
- How to **avoid** comparing all pairs of documents?
  1. Compute **shingle set**  $S(d)$  for each document  $d$
  2. Build **inverted index**: shingle  $\Rightarrow$  list of document identifiers
  3. Compute  $(d, d', c)$  table with common-shingle count  $c$  by considering **all pairs of documents  $(d, d')$  per shingle**
  4. Keep all pairs of documents  $(d, d')$  with **similarity above threshold** and add  $(d, d')$  as edge to a graph
  5. Compute **connected components** of graph (using union-find algorithm) as clusters of near-duplicate documents

# Super Shingles and Complexity

- **Super shingles** (shingles over shingles) can be used to speed up steps 2 and 3 of the algorithm, since documents with many common shingles are likely to have common super shingle
- Algorithm considers **only pairs** of documents that have **at least one shingle in common**, but worst case remains at  $O(n^2)$
- Problem: Shingle sets can become quite large, making the similarity computation expensive
- Full details: [Broder et al. '97]

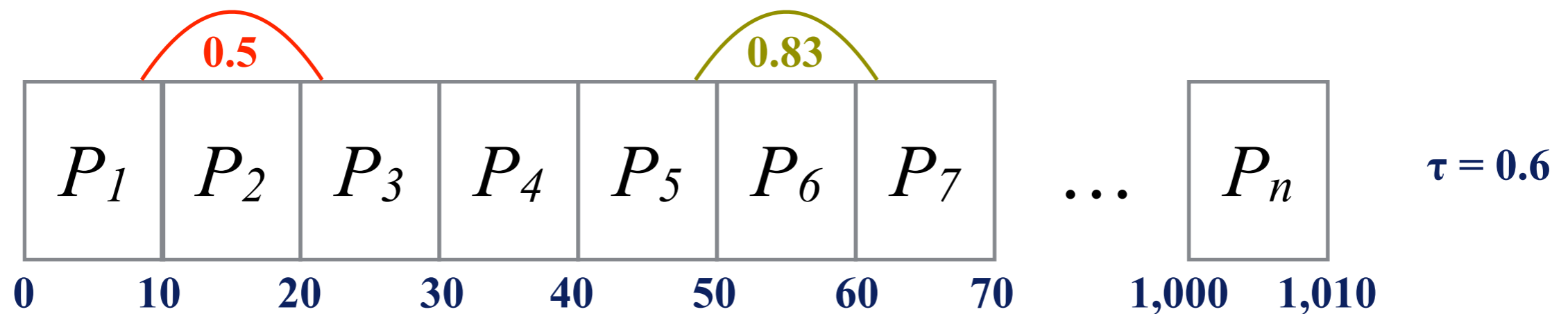
## 2. SpotSigs

- Problem: Near-duplicate detection on the Web fails for web pages with **same core content** but **different navigation, header, etc.**
- Observation: **Stopwords** tend to occur mostly in core content
- **SpotSigs** considers only those shingles that begin with a stopword
- Problem: How can we perform fewer similarity computations?
- **Upper bound** for Jaccard coefficient

$$r(A, B) = \frac{|A \cap B|}{|A \cup B|} \leq \frac{\min(|A|, |B|)}{\max(|A|, |B|)}$$
$$\leq \frac{|A|}{|B|} \quad (\text{assuming } |A| \leq |B| \text{ w.l.o.g.})$$

# SpotSigs

- Do not compare any sets  $|A|$  and  $|B|$  with  $|A| / |B| \leq \tau$
- Given similarity threshold  $\tau$ , **partition the documents** (based on their signature set cardinality) into partitions  $P_1, \dots, P_n$



- Consider document pairs in  $P_i \times P_j$  ( $i \leq j$ ) only if

$$\frac{|\max\{|S(d)| \mid d \in P_i\}|}{|\min\{|S(d)| \mid d \in P_j\}|} > \tau$$

- **Clever partitioning** to compare at most neighboring partitions
- Full details: [Theobald et al. '08]

### 3. Min-Wise Independent Permutations

- **Statistical sketch** to estimate the resemblance of  $S(d)$  and  $S(d')$ 
  - consider  $m$  **independent random permutations** of the two sets, implemented by applying  $m$  **independent hash functions**
  - keep the **minimum value** observed for each of the  $m$  hash functions, yielding a  $m$ -dimensional MIPs vector for each document
  - **estimate resemblance** of  $S(d)$  and  $S(d')$  based on  $MIPs(d)$  and  $MIPs(d')$

$$\hat{r}(d, d') = \frac{|\{1 \leq i \leq m \mid MIPs(d)[i] = MIPs(d')[i]\}|}{m}$$

- Full details: [Broder et al. '00]

# Min-Wise Independent Permutations

Set of shingle fingerprints

$$S(d) = \{3, 8, 12, 17, 21, 24\}$$

$$h_1(x) = 7x + 3 \pmod{51}$$

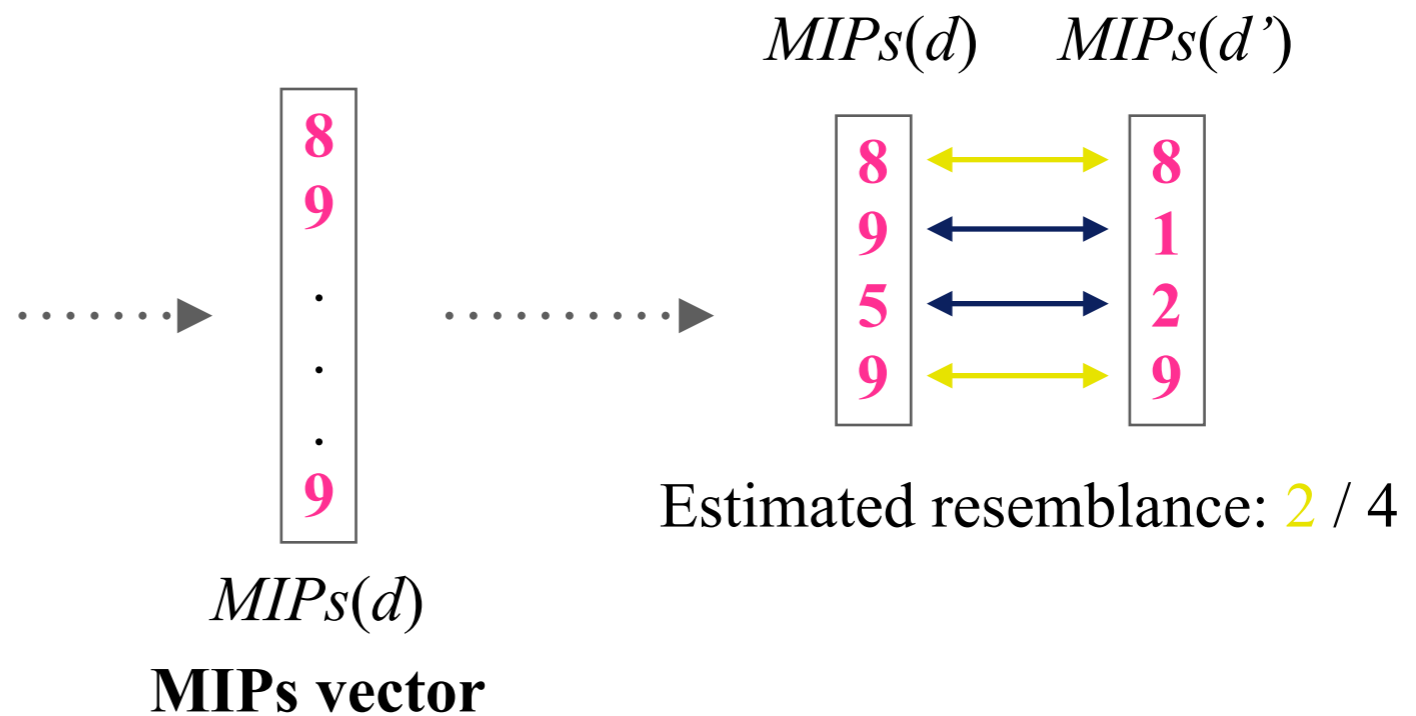
$$\{24, 8, 36, 20, 48, 18\}$$

$$h_2(x) = 5x + 6 \pmod{51}$$

$$\{21, 46, 15, 40, 9, 24\}$$

$$h_m(x) = 3x + 9 \pmod{51}$$

$$\{18, 33, 45, 9, 21, 30\}$$



- MIPs are an **unbiased estimator of resemblance**

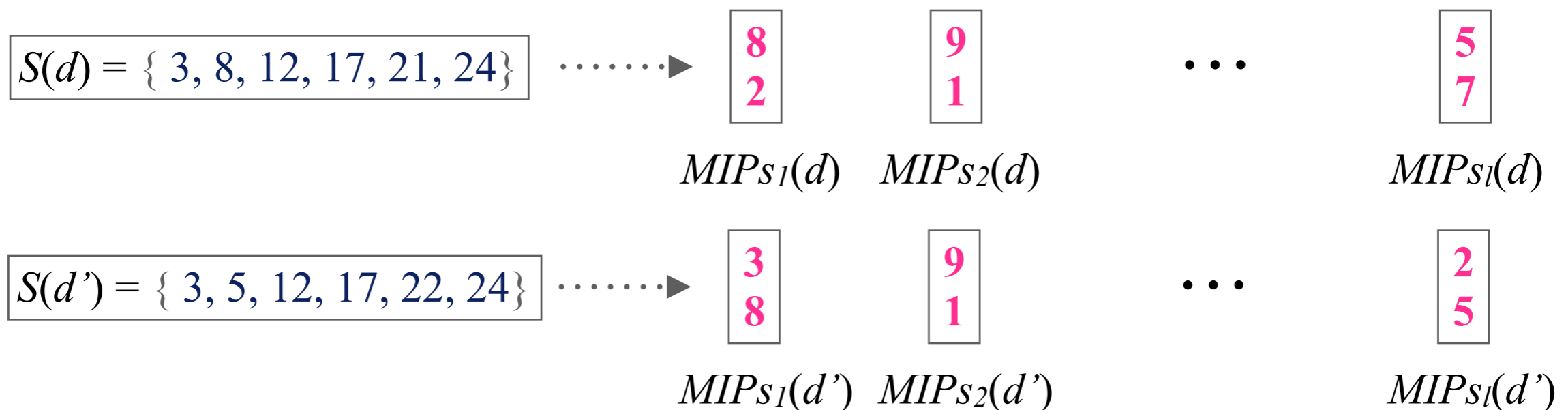
$$P[\min\{h(x)|x \in A\} = \min\{h(y)|y \in B\}] = |A \cap B| / |A \cup B|$$

- MIPs can be seen as **repeated random sampling** of  $x, y$  from  $A, B$



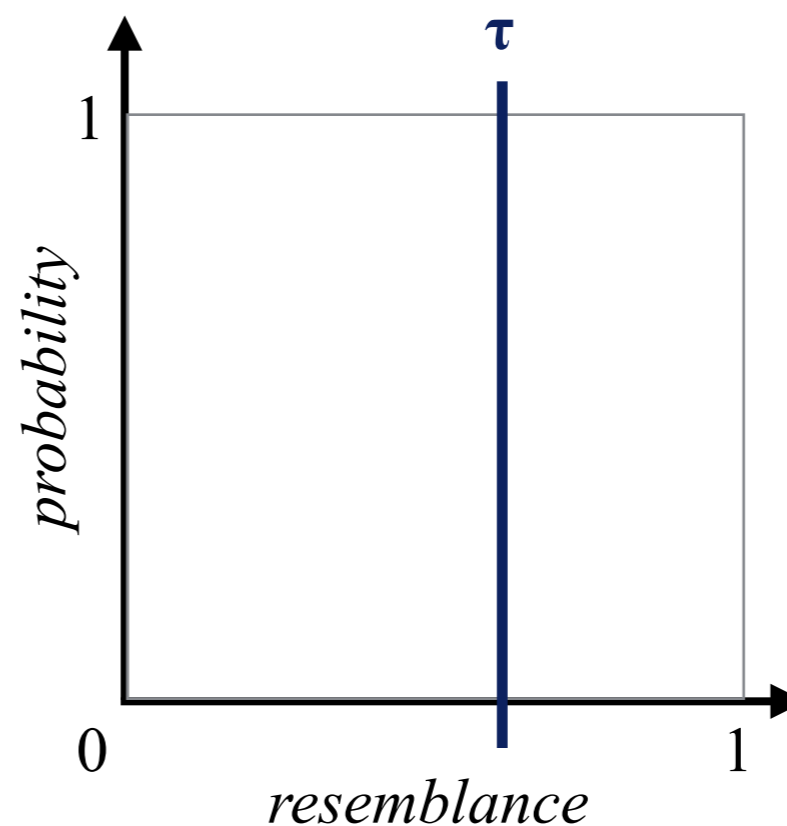
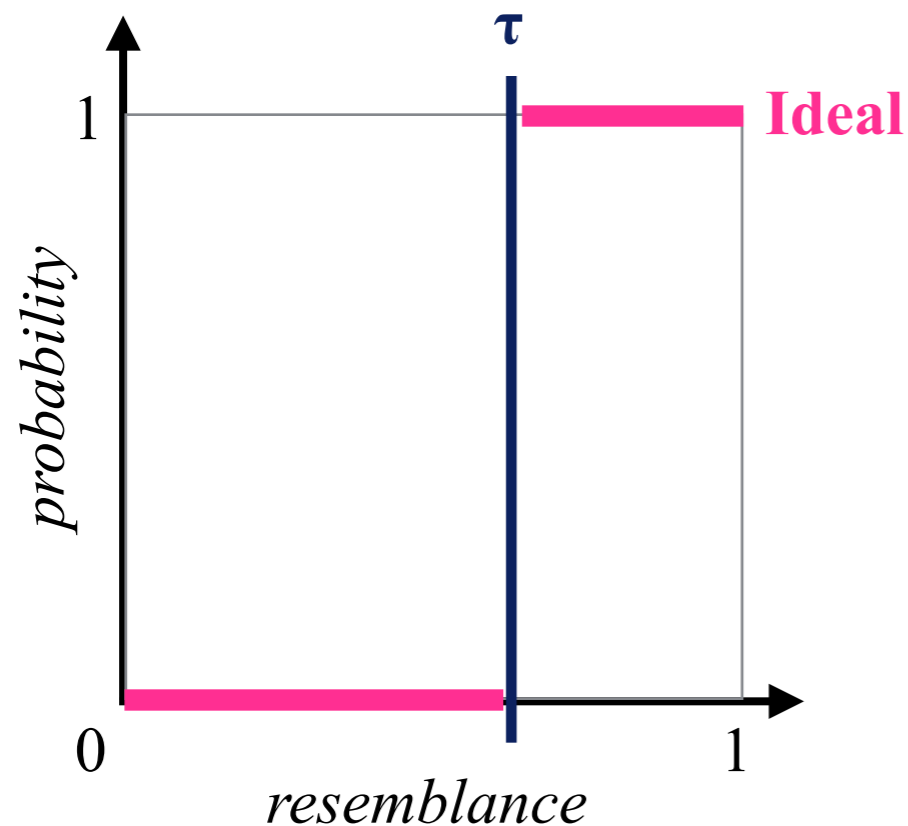
## 4. Locality Sensitive Hashing (for MIPs)

- General idea behind **locality sensitive hashing** (LSH)
  - hash each item  $l$  times so that **similar items map to same bucket**
  - consider pairs of items similar that mapped **at least once to same bucket**
- Locality sensitive hashing with MIPs vectors
  - compute  $l$  **independent MIPs vectors of length  $m$**  for each document
  - consider document pairs with **at least one common MIPs vector**



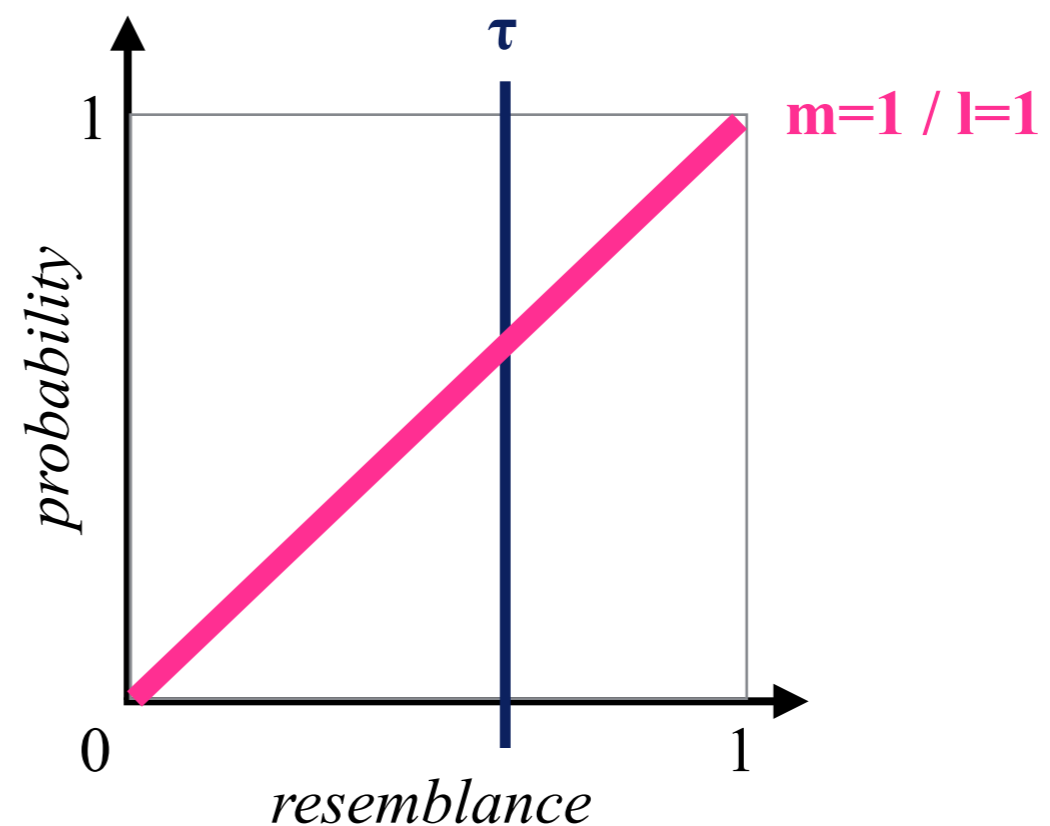
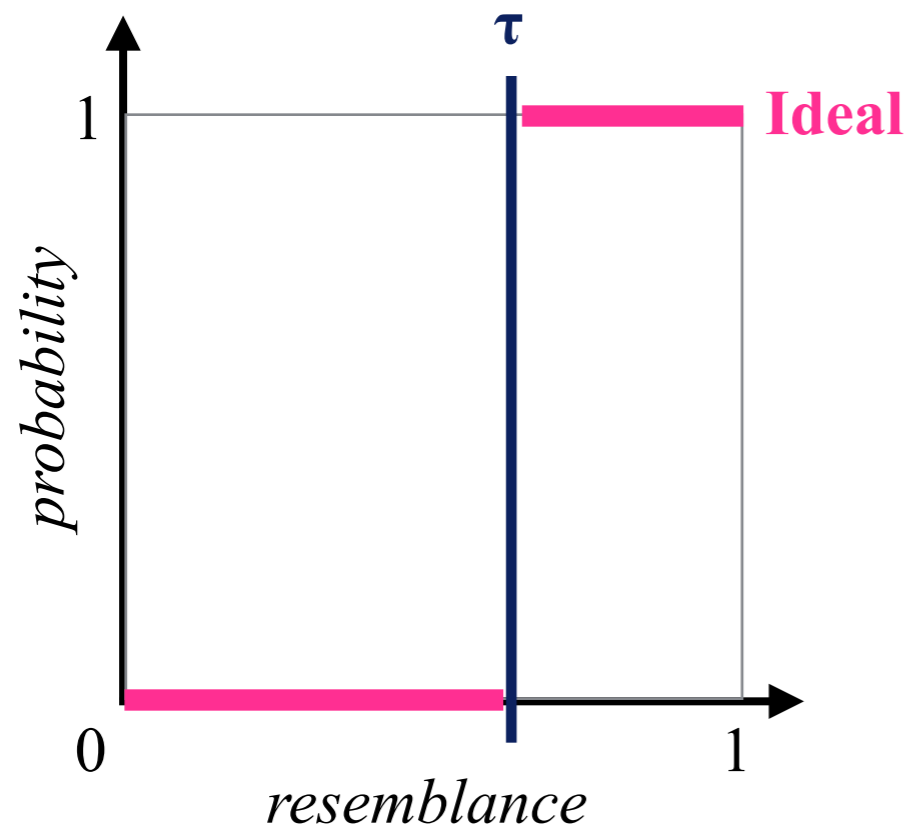
# LSH Analysis

- Let  $r = r(d, d')$  denote the resemblance between  $d$  and  $d'$ 
  - $P[MIP_{S_i}(d) = MIP_{S_i}(d')] = r^m$  : same  $i$ -th MIPs vector
  - $1 - r^m$  : different  $i$ -th MIPs vector
  - $(1 - r^m)^l$  : all MIPs vectors different
  - $1 - (1 - r^m)^l$  : at least one MIPs vector in common



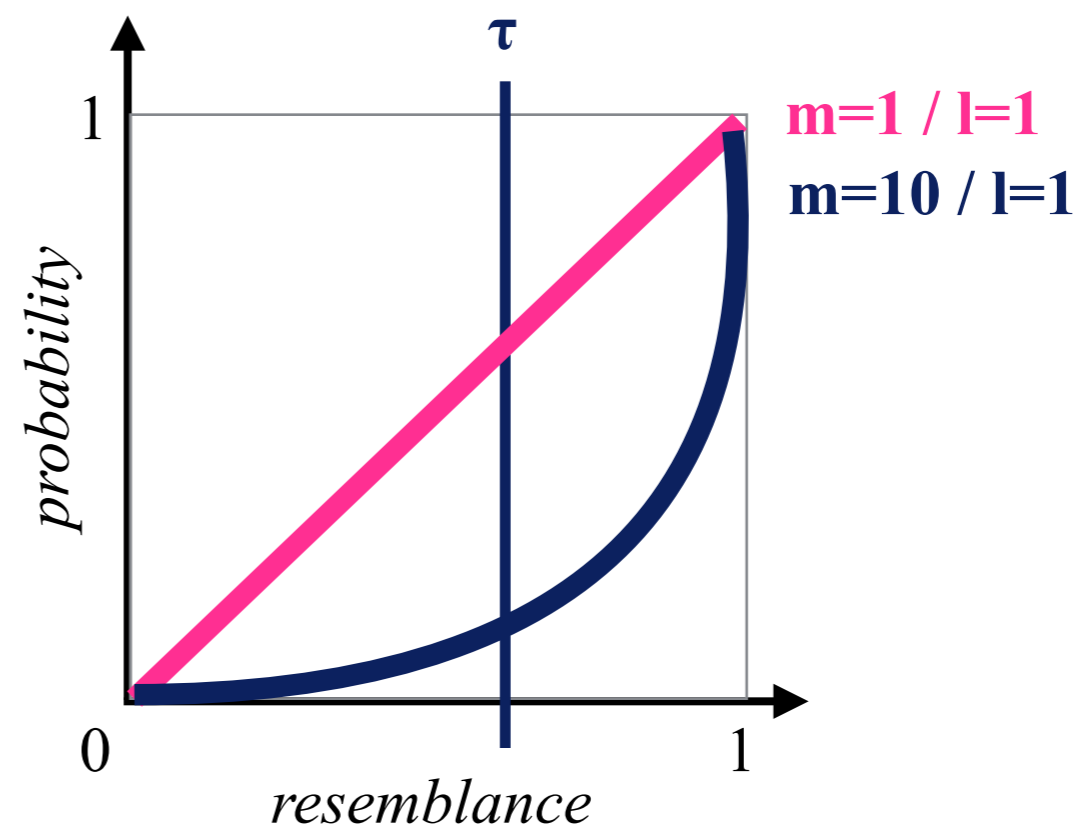
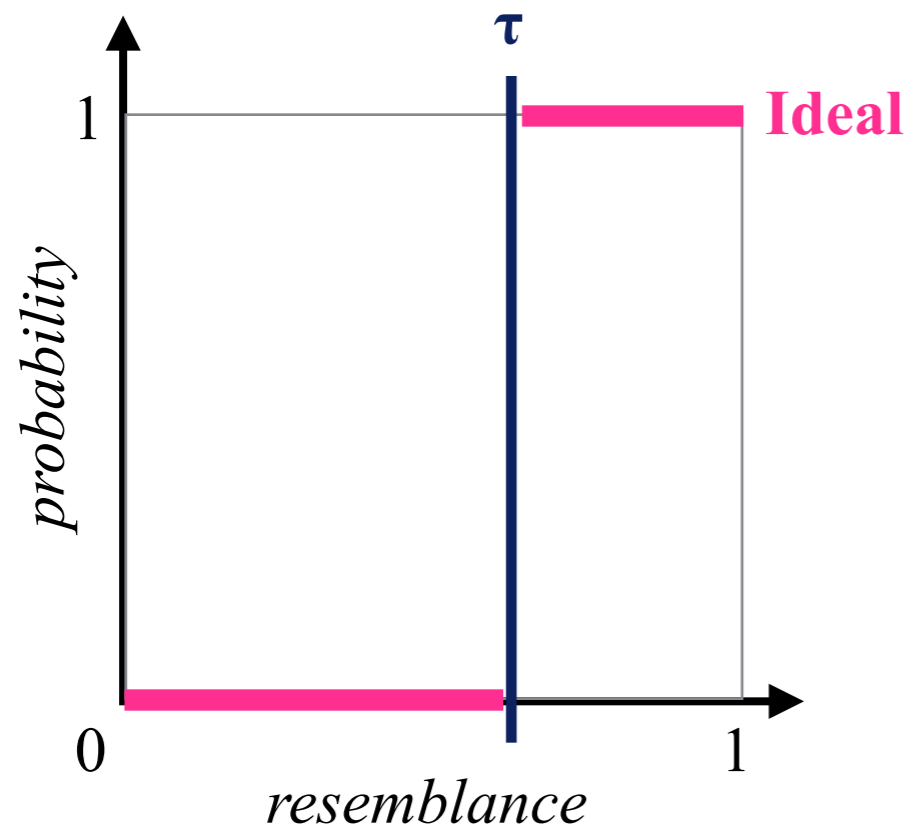
# LSH Analysis

- Let  $r = r(d, d')$  denote the resemblance between  $d$  and  $d'$ 
  - $P[MIP_{S_i}(d) = MIP_{S_i}(d')] = r^m$  : same  $i$ -th MIPs vector
  - $1 - r^m$  : different  $i$ -th MIPs vector
  - $(1 - r^m)^l$  : all MIPs vectors different
  - $1 - (1 - r^m)^l$  : at least one MIPs vector in common



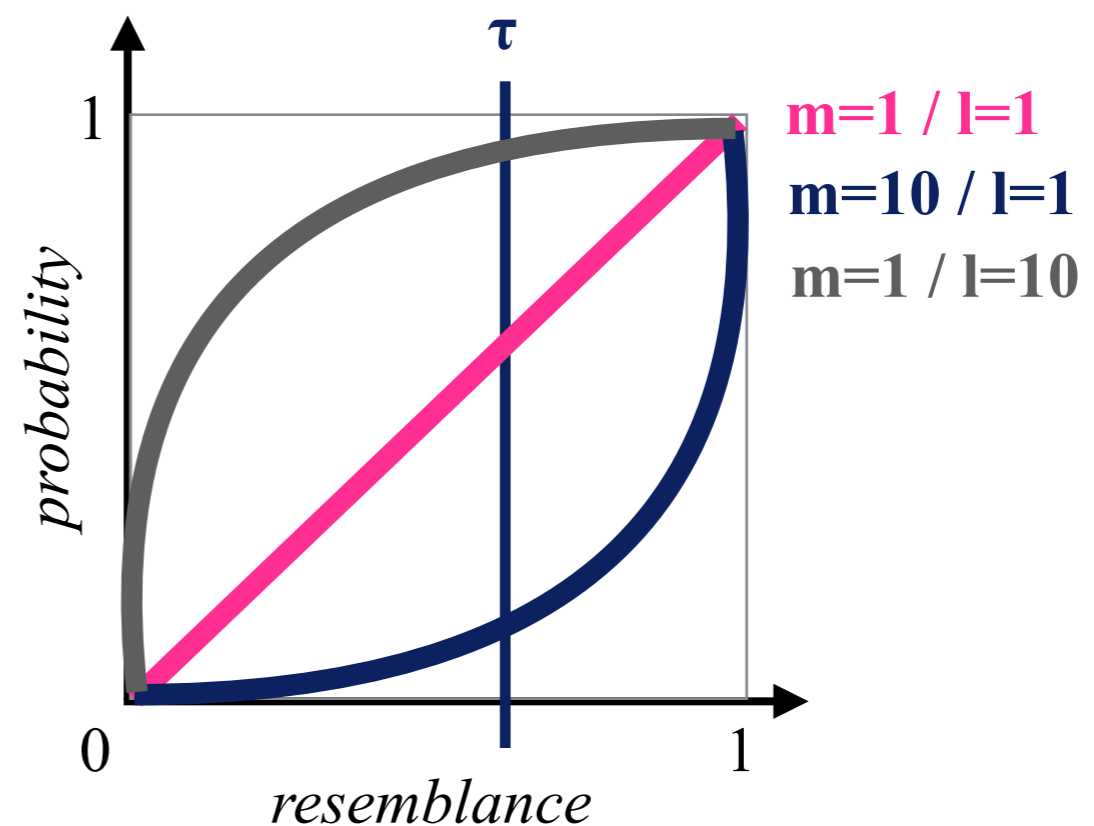
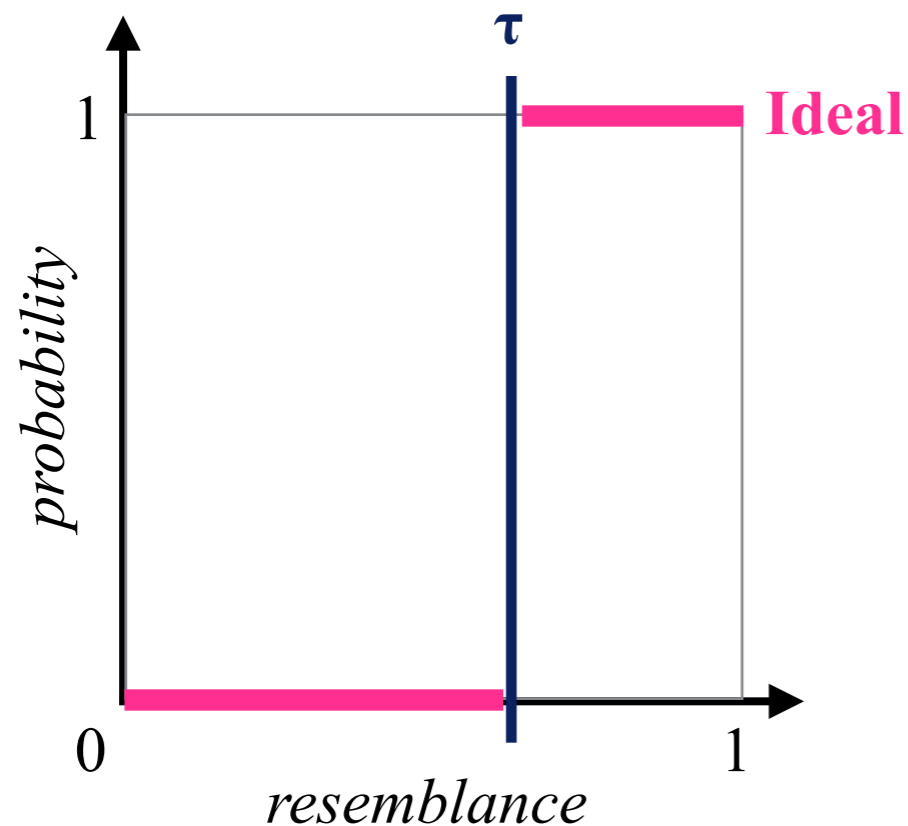
# LSH Analysis

- Let  $r = r(d, d')$  denote the resemblance between  $d$  and  $d'$ 
  - $P[MIP_{S_i}(d) = MIP_{S_i}(d')] = r^m$  : same  $i$ -th MIPs vector
  - $1 - r^m$  : different  $i$ -th MIPs vector
  - $(1 - r^m)^l$  : all MIPs vectors different
  - $1 - (1 - r^m)^l$  : at least one MIPs vector in common



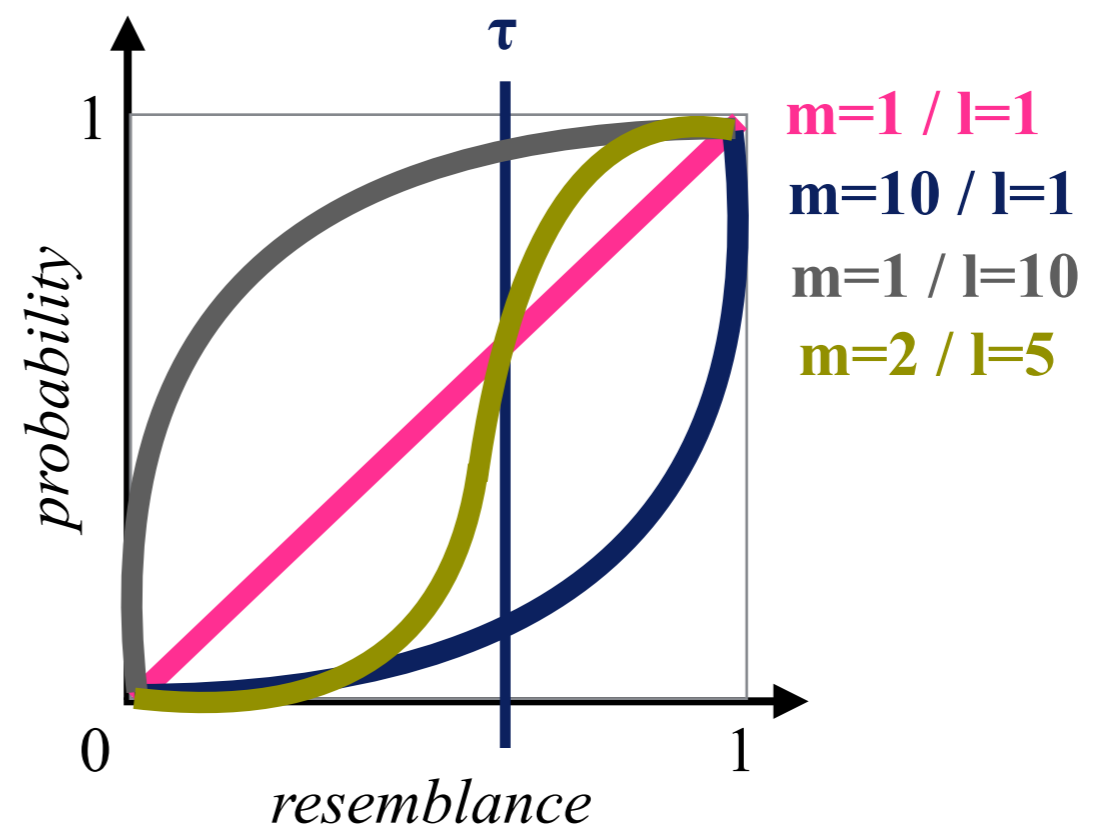
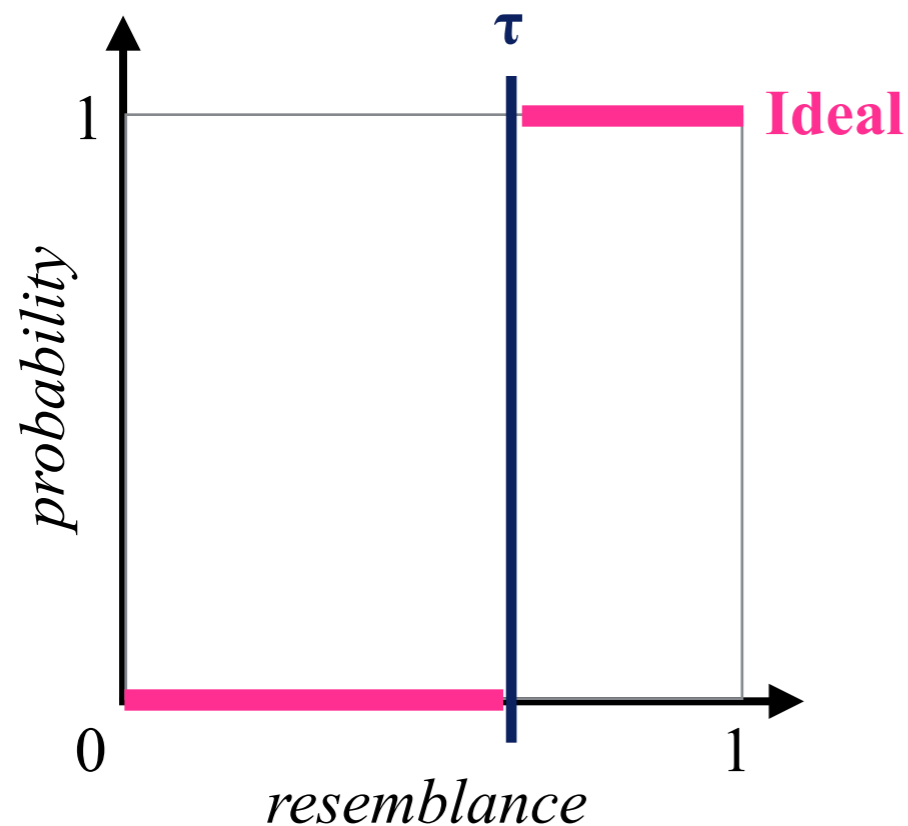
# LSH Analysis

- Let  $r = r(d, d')$  denote the resemblance between  $d$  and  $d'$ 
  - $P[MIP_{S_i}(d) = MIP_{S_i}(d')] = r^m$  : same  $i$ -th MIPs vector
  - $1 - r^m$  : different  $i$ -th MIPs vector
  - $(1 - r^m)^l$  : all MIPs vectors different
  - $1 - (1 - r^m)^l$  : at least one MIPs vector in common



# LSH Analysis

- Let  $r = r(d, d')$  denote the resemblance between  $d$  and  $d'$ 
  - $P[MIP_{S_i}(d) = MIP_{S_i}(d')] = r^m$  : same  $i$ -th MIPs vector
  - $1 - r^m$  : different  $i$ -th MIPs vector
  - $(1 - r^m)^l$  : all MIPs vectors different
  - $1 - (1 - r^m)^l$  : at least one MIPs vector in common



# LSH Analysis

- Example: For a pair of documents  $d$  and  $d'$  with  $r(d, d') = 0.8$ ,  $m = 5$ , and  $l = 20$ , the probability of missing the pair is  $(1 - 0.8^5)^{20} = 3.56 \times 10^{-4}$
- Full details: [Gionis et al. '99]

# Summary of V.5

- **Near-Duplicate Detection**  
essential for smaller indexes and better result quality
- **Shingling**  
to deal with small perturbations in otherwise duplicate documents
- **SpotSigs**  
focuses on shingles beginning with a stopword  
uses smart blocking to compare fewer document pairs
- **Min-Wise Independent Permutations**  
as a statistical sketch to approximate resemblance
- **Locality-Sensitive Hashing**  
as a method to reduce the number of document comparisons



# Additional Literature for V.5

- **A. Broder, S. Glassman, M. Manasse, and G. Zweig:** *Syntactic Clustering of the Web*, WWW 1997
- **A. Broder, M. Charikar, A. Frieze, M. Mitzenmacher:** *Min-Wise Independent Permutations*, JCSS 60(3):630-659, 2000
- **A. Gionis, P. Indyk, and R. Motwani:** *Similarity Search in High Dimensions via Hashing*, VLDB 1999
- **M. Henzinger:** *Finding Near-Duplicate Web Pages: a Large-Scale Evaluation of Algorithms*, SIGIR 2006
- **M. Theobald, J. Siddharth, and A. Paepcke:** *SpotSigs: Robust and efficient near duplicate detection in large web collections*, SIGIR 2008